

BdsApi

3.1.4

Generated by Doxygen 1.9.5

1 BdsApi	1
1.1 Introduction	1
1.2 Overview	2
1.3 C++ Examples	2
1.4 Python Examples	5
2 Namespace Index	9
2.1 Namespace List	9
3 Hierarchical Index	11
3.1 Class Hierarchy	11
4 Class Index	15
4.1 Class List	15
5 File Index	19
5.1 File List	19
6 Namespace Documentation	21
6.1 Bds Namespace Reference	21
6.1.1 Typedef Documentation	28
6.1.1.1 DataFormats	28
6.1.2 Enumeration Type Documentation	29
6.1.2.1 Errors	29
6.1.2.2 Priority	29
6.1.2.3 Mode	29
6.1.2.4 DataFlags	30
6.1.2.5 SelectionGroup	30
6.1.2.6 SampleFormat	30
6.1.2.7 AvailType	32
6.1.2.8 DataFormatSet	32
6.1.2.9 LocationSelect	32
6.1.2.10 BdsDataType	33
6.1.2.11 FileHeaderType	33
6.1.2.12 FileSampleType	33
6.1.3 Function Documentation	34
6.1.3.1 bdsLibInit() [1/3]	34
6.1.3.2 bdsLibInit() [2/3]	34
6.1.3.3 bdsLibInit() [3/3]	34
6.1.3.4 bdsDumpPoleZeros()	34
6.1.3.5 bdsChannelGetTypeAux()	34
6.1.3.6 bdsChannelGetName()	35
6.1.3.7 bdsDataInfoSetTimeRange()	35
6.1.3.8 bdsDataInfoFromInfo()	35

6.1.3.9 bdsInfoFromDataInfo()	35
6.1.3.10 bdsDataInfoFlatten()	36
6.1.3.11 bdsDataInfoMergeFlatten()	36
6.1.3.12 bdsUnitsConvert()	36
6.1.3.13 bdsDataInfoFromChannelInfos()	36
6.1.3.14 responseSort()	36
6.1.3.15 bdsMetadataImportFix()	37
6.1.3.16 bdsMetadataExportFix()	37
6.1.3.17 bdsStationAlias()	37
6.1.3.18 bdsDumpSelection()	37
6.1.3.19 bdsDumpSelectionInfo()	37
6.1.3.20 bdsDumpDataInfo()	38
6.1.3.21 bdsDumpChannelInfos()	38
6.1.3.22 bdsDumpData()	38
6.1.3.23 bdsDumpLocation()	38
6.1.3.24 bdsDataChannelInfo()	38
6.1.3.25 bdsDataChannelRef() [1/2]	39
6.1.3.26 bdsDataChannelRef() [2/2]	39
6.1.3.27 bdsDataChannelOverallResponse()	39
6.1.3.28 bdsSelectionChannelInfo()	39
6.1.3.29 bdsPoleZeroGain()	39
6.1.3.30 bdsPoleZeroGainPhase()	40
6.1.3.31 bdsPoleZeroToFap()	40
6.1.3.32 fileNameTime()	40
6.1.3.33 bdsFileNameExpand() [1/3]	40
6.1.3.34 bdsFileNameExpand() [2/3]	40
6.1.3.35 bdsFileNameExpand() [3/3]	41
6.1.3.36 bdsSpecialChannelsSet()	41
6.1.3.37 bdsSpecialChannels()	41
6.1.3.38 bdsSpecialChannelIgnore()	41
6.1.3.39 seedChannelInstrumentCode()	41
6.1.3.40 seedChannelDataType()	42
6.1.3.41 bdsDataTypes()	42
6.1.3.42 bdsUnits()	42
6.1.3.43 bdsUnitCase()	42
6.1.3.44 bdsUnCompressCm8()	42
6.1.3.45 bdsUnCompressSteim1()	43
6.1.3.46 nullString()	43
6.1.3.47 crc()	43
6.1.3.48 crclnit()	43
6.1.3.49 crc64()	43
6.1.3.50 getHexString()	43

6.1.3.51 duplicateDump()	44
6.1.3.52 fixedString()	44
6.1.3.53 dataCalculateDifference()	44
6.1.3.54 dataCalculateUnDifference()	44
6.1.3.55 dataChecksum()	44
6.1.3.56 dataCompressCm6()	44
6.1.3.57 dataDeCompressCm6()	45
6.1.3.58 dataConvert() [1/3]	45
6.1.3.59 unitsCode()	45
6.1.3.60 stringFormat()	45
6.1.3.61 removeCR()	45
6.1.3.62 fixedWidthValue()	45
6.1.3.63 roundDigits()	46
6.1.3.64 bdsDataFileSeedLogWarning()	46
6.1.3.65 bdsDataFileSeedLogError()	46
6.1.3.66 seedTime()	46
6.1.3.67 seedTimeString()	46
6.1.3.68 fromSeedTimeString()	46
6.1.3.69 dataConvert() [2/3]	46
6.1.3.70 dataConvert() [3/3]	47
6.1.3.71 record_handler()	47
6.1.4 Variable Documentation	47
6.1.4.1 apiVersion	47
6.1.4.2 bdsSpecialChannelsList	47
6.1.4.3 seedIcodeToDataTypes	47
6.1.4.4 NetworkNameLen	48
6.1.4.5 StationNameLen	48
6.1.4.6 ChannelTypeLen	48
6.1.4.7 ChannelAuxLen	48
6.1.4.8 SourceLen	48
6.1.4.9 BdsDataFileVersion	48
6.1.4.10 crcVec	49
6.1.4.11 crclnitDone	49
6.1.4.12 cm6Table	49
6.1.4.13 cm6TableRev	49
6.1.4.14 node_types	49
6.1.4.15 Scale	50
6.1.4.16 dataFormatAll	50
7 Class Documentation	51
7.1 Bds::AccessGroup Class Reference	51
7.1.1 Detailed Description	52

7.1.2 Constructor & Destructor Documentation	52
7.1.2.1 AccessGroup()	52
7.1.3 Member Function Documentation	52
7.1.3.1 getType()	52
7.1.3.2 setMembers()	52
7.1.3.3 setMember()	52
7.1.3.4 getMembers()	53
7.1.3.5 getMember()	53
7.1.4 Member Data Documentation	53
7.1.4.1 id	53
7.1.4.2 group	53
7.1.4.3 startTime	53
7.1.4.4 endTime	54
7.1.4.5 network	54
7.1.4.6 station	54
7.2 Bds::AdminAccess Class Reference	54
7.2.1 Detailed Description	59
7.2.2 Constructor & Destructor Documentation	60
7.2.2.1 AdminAccess()	60
7.2.3 Member Function Documentation	60
7.2.3.1 connect()	60
7.2.3.2 validateUser()	60
7.2.3.3 setUser()	61
7.2.3.4 setUserReal()	61
7.2.3.5 getVersion()	61
7.2.3.6 userGetList()	61
7.2.3.7 userUpdate()	62
7.2.3.8 userDelete()	62
7.2.3.9 userGetFromId()	62
7.2.3.10 userGet()	62
7.2.3.11 userSet()	63
7.2.3.12 userGetGroups()	63
7.2.3.13 userGetOptions()	63
7.2.3.14 userSetOptions()	63
7.2.3.15 groupGetList()	63
7.2.3.16 groupUpdate()	64
7.2.3.17 groupDelete()	64
7.2.3.18 accessGroupGetList()	64
7.2.3.19 accessGroupUpdate()	64
7.2.3.20 accessGroupDelete()	64
7.2.3.21 getSelectionInfo()	64
7.2.3.22 getSelections()	65

7.2.3.23 networkGetList()	65
7.2.3.24 networkUpdate()	65
7.2.3.25 networkDelete()	66
7.2.3.26 stationGetList()	66
7.2.3.27 stationUpdate()	66
7.2.3.28 stationDelete()	66
7.2.3.29 locationGetList()	66
7.2.3.30 locationUpdate()	67
7.2.3.31 locationDelete()	67
7.2.3.32 channelGetList()	67
7.2.3.33 channelGet()	68
7.2.3.34 channelUpdate()	68
7.2.3.35 channelDelete()	68
7.2.3.36 sourceGetList()	68
7.2.3.37 sourceUpdate()	68
7.2.3.38 sourceDelete()	69
7.2.3.39 sourcePriorityGetList()	69
7.2.3.40 sourcePriorityUpdate()	69
7.2.3.41 sourcePriorityDelete()	69
7.2.3.42 channelInstrumentGetList()	69
7.2.3.43 channelInstrumentUpdate()	70
7.2.3.44 channelInstrumentDelete()	70
7.2.3.45 digitiserGetList()	70
7.2.3.46 digitiserGet()	70
7.2.3.47 digitiserUpdate()	71
7.2.3.48 digitiserDelete()	71
7.2.3.49 sensorGetList()	71
7.2.3.50 sensorGet()	71
7.2.3.51 sensorUpdate()	72
7.2.3.52 sensorDelete()	72
7.2.3.53 calibrationGetList()	72
7.2.3.54 calibrationUpdate()	72
7.2.3.55 calibrationDelete()	72
7.2.3.56 responseGetList()	72
7.2.3.57 responseUpdate()	73
7.2.3.58 responseDelete()	73
7.2.3.59 eventGetList()	73
7.2.3.60 eventUpdate()	73
7.2.3.61 eventDelete()	74
7.2.3.62 specialChannelGetList()	74
7.2.3.63 specialChannelUpdate()	74
7.2.3.64 specialChannelDelete()	74

7.2.3.65 metadataGetChannelInfo()	75
7.2.3.66 metadataGetFormatted()	75
7.2.3.67 dataFileGetList()	75
7.2.3.68 dataFileUpdate()	76
7.2.3.69 dataFileDelete()	76
7.2.3.70 dataChannelGetList()	76
7.2.3.71 dataChannelUpdate()	77
7.2.3.72 dataChannelDelete()	77
7.2.3.73 dataAvailability()	77
7.2.3.74 dataSearch()	77
7.2.3.75 dataGetChannelInfo()	79
7.2.3.76 dataOpen()	79
7.2.3.77 dataGetInfo()	80
7.2.3.78 dataGetNotes()	81
7.2.3.79 dataGetWarnings()	81
7.2.3.80 dataGetBlock()	81
7.2.3.81 dataSeekBlock()	82
7.2.3.82 dataSetInfo()	82
7.2.3.83 dataPutBlock()	83
7.2.3.84 dataClose()	83
7.2.3.85 dataFormattedRead()	83
7.2.3.86 dataFormattedGetLength()	83
7.2.3.87 dataRealtimeConfig()	84
7.2.3.88 dataRealtimeGet()	84
7.2.3.89 changeGroupStart()	84
7.2.3.90 changeGroupEnd()	85
7.2.3.91 changeGroupGetList()	85
7.2.3.92 changeGroupDelete()	85
7.2.3.93 changeGetListNumber()	85
7.2.3.94 changeGetList()	85
7.2.3.95 changeDelete()	86
7.2.3.96 noteGetList()	86
7.2.3.97 noteUpdate()	86
7.2.3.98 noteDelete()	86
7.2.3.99 noteWriteDocument()	87
7.2.3.100 noteReadDocument()	87
7.2.3.101 logGetList()	87
7.2.3.102 logUpdate()	87
7.2.3.103 logDelete()	88
7.2.3.104 logAppend()	88
7.2.3.105 statisticsGet()	88
7.2.3.106 serverConfigurationGet()	89

7.2.3.107 dataFormatGetList()	89
7.2.3.108 transactionStart()	89
7.2.3.109 transactionEnd()	90
7.2.3.110 modeSet()	90
7.2.3.111 modeSnapshotPause()	90
7.2.3.112 clean()	90
7.2.3.113 databaseBackup()	92
7.2.3.114 databaseRestore()	92
7.2.3.115 sqlQuery()	92
7.2.3.116 extraCall()	93
7.3 Bds::ArrayChannel Class Reference	93
7.3.1 Detailed Description	94
7.3.2 Constructor & Destructor Documentation	94
7.3.2.1 ArrayChannel()	94
7.3.3 Member Data Documentation	94
7.3.3.1 network	94
7.3.3.2 station	95
7.3.3.3 channel	95
7.3.3.4 arrayOffsetEast	95
7.3.3.5 arrayOffsetNorth	95
7.4 Bds::BdsDataBlock Struct Reference	95
7.4.1 Detailed Description	96
7.4.2 Member Data Documentation	96
7.4.2.1 header	96
7.4.2.2 data	96
7.5 Bds::BdsDataBlockHeader Struct Reference	96
7.5.1 Detailed Description	96
7.5.2 Member Data Documentation	97
7.5.2.1 type	97
7.5.2.2 length	97
7.5.2.3 packetOffset	97
7.6 Bds::BdsDataBlockPos Class Reference	97
7.6.1 Detailed Description	98
7.6.2 Constructor & Destructor Documentation	98
7.6.2.1 BdsDataBlockPos()	98
7.6.3 Member Function Documentation	98
7.6.3.1 operator<()	98
7.6.4 Member Data Documentation	98
7.6.4.1 startTime	98
7.6.4.2 endTime	99
7.6.4.3 channel	99
7.6.4.4 numChannels	99

7.6.4.5 segment	99
7.6.4.6 position	99
7.6.4.7 numSamples	99
7.7 Bds::BdsDataPacket Class Reference	99
7.7.1 Detailed Description	100
7.7.2 Constructor & Destructor Documentation	100
7.7.2.1 BdsDataPacket()	100
7.7.2.2 ~BdsDataPacket()	100
7.7.3 Member Function Documentation	100
7.7.3.1 clear()	100
7.7.3.2 reset()	101
7.7.3.3 setChecksumAndLength()	101
7.7.3.4 validateChecksum()	101
7.7.3.5 setHeader()	101
7.7.3.6 getHeader()	101
7.7.3.7 dump()	101
7.8 Bds::BdsDataPacketHeader Struct Reference	101
7.8.1 Detailed Description	102
7.8.2 Member Data Documentation	102
7.8.2.1 type	102
7.8.2.2 length	102
7.8.2.3 streamlet	102
7.8.2.4 sequence	103
7.8.2.5 checksum	103
7.8.2.6 startTime	103
7.8.2.7 endTime	103
7.9 Bds::BdsDataSegment Class Reference	103
7.9.1 Detailed Description	104
7.9.2 Constructor & Destructor Documentation	104
7.9.2.1 BdsDataSegment()	104
7.9.3 Member Function Documentation	104
7.9.3.1 operator<()	104
7.9.4 Member Data Documentation	104
7.9.4.1 startTime	104
7.9.4.2 endTime	104
7.9.4.3 numBlocks	105
7.9.4.4 numSamples	105
7.9.4.5 sampleRate	105
7.9.4.6 blocks	105
7.10 Bds::BdsDataStreamlet Class Reference	105
7.10.1 Detailed Description	106
7.10.2 Constructor & Destructor Documentation	106

7.10.2.1 BdsDataStreamlet()	106
7.10.3 Member Data Documentation	106
7.10.3.1 packetNumber	106
7.10.3.2 position	106
7.10.3.3 channel	106
7.10.3.4 numChannels	106
7.10.3.5 blocks	106
7.10.3.6 segments	107
7.11 Bds::BdsSeedType Class Reference	107
7.11.1 Detailed Description	107
7.11.2 Constructor & Destructor Documentation	107
7.11.2.1 BdsSeedType()	107
7.11.3 Member Function Documentation	108
7.11.3.1 getInt()	108
7.11.3.2 getUInt()	108
7.11.3.3 getDouble()	108
7.11.3.4 getString()	108
7.11.3.5 getStringVariable()	108
7.11.3.6 appendInt()	109
7.11.3.7 appendDouble()	109
7.11.3.8 appendExp()	109
7.11.3.9 appendString()	109
7.11.3.10 appendStringVariable()	109
7.12 Bds::Calibration Class Reference	110
7.12.1 Detailed Description	111
7.12.2 Constructor & Destructor Documentation	111
7.12.2.1 Calibration()	111
7.12.3 Member Function Documentation	112
7.12.3.1 getType()	112
7.12.3.2 setMembers()	112
7.12.3.3 setMember()	112
7.12.3.4 getMembers()	112
7.12.3.5 getMember()	113
7.12.4 Member Data Documentation	113
7.12.4.1 id	113
7.12.4.2 startTime	113
7.12.4.3 endTime	113
7.12.4.4 network	113
7.12.4.5 station	114
7.12.4.6 channel	114
7.12.4.7 source	114
7.12.4.8 name	114

7.12.4.9 samplingFrequency	114
7.12.4.10 calibrationFrequency	114
7.12.4.11 calibrationFactor	115
7.12.4.12 calibrationUnits	115
7.12.4.13 calibrationUnitsDesc	115
7.12.4.14 rawCalibrationFrequency	115
7.12.4.15 rawCalibrationFactor	115
7.12.4.16 rawCalibrationUnits	115
7.12.4.17 depth	116
7.12.4.18 waterLevel	116
7.12.4.19 horizontalAngle	116
7.12.4.20 verticalAngle	116
7.13 Bds::CdChannel_1v0 Struct Reference	116
7.13.1 Detailed Description	117
7.13.2 Member Data Documentation	117
7.13.2.1 auth	117
7.13.2.2 compress	117
7.13.2.3 spare0	117
7.13.2.4 spare1	117
7.13.2.5 calibrationFactor	117
7.13.2.6 calibrationPeriod	117
7.13.2.7 name	118
7.13.2.8 stationName	118
7.13.2.9 channelName	118
7.13.2.10 channel	118
7.14 Bds::CdDataChannel Class Reference	118
7.14.1 Detailed Description	119
7.14.2 Member Data Documentation	119
7.14.2.1 station	119
7.14.2.2 channel	119
7.14.2.3 mode	119
7.14.2.4 status	119
7.14.2.5 startTime	119
7.14.2.6 period	119
7.14.2.7 numSamples	120
7.14.2.8 dataSize	120
7.14.2.9 data	120
7.15 Bds::CdDataFormatFrame_1v0 Struct Reference	120
7.15.1 Detailed Description	120
7.15.2 Member Data Documentation	120
7.15.2.1 frameType	121
7.15.2.2 frameLength	121

7.15.2.3 maxFrameLength	121
7.15.2.4 numChannels	121
7.15.2.5 period	121
7.15.2.6 channels	121
7.16 Bds::CdFlag Class Reference	121
7.16.1 Detailed Description	122
7.16.2 Constructor & Destructor Documentation	122
7.16.2.1 CdFlag()	122
7.16.3 Member Data Documentation	122
7.16.3.1 dead	122
7.16.3.2 zeroed	122
7.17 Bds::CdPacketData Class Reference	122
7.17.1 Detailed Description	123
7.17.2 Member Data Documentation	123
7.17.2.1 frameType	123
7.17.2.2 trailerOffset	123
7.17.2.3 creator	123
7.17.2.4 destination	123
7.17.2.5 sequenceNum	124
7.17.2.6 series	124
7.17.2.7 numChannels	124
7.17.2.8 period	124
7.17.2.9 startTime	124
7.17.2.10 channels	124
7.17.2.11 authKey	124
7.17.2.12 authSize	124
7.17.2.13 auth	125
7.17.2.14 crc	125
7.18 Bds::Change Class Reference	125
7.18.1 Detailed Description	126
7.18.2 Constructor & Destructor Documentation	126
7.18.2.1 Change()	126
7.18.3 Member Function Documentation	126
7.18.3.1 getType()	126
7.18.3.2 setMembers()	126
7.18.3.3 setMember()	126
7.18.3.4 getMembers()	127
7.18.3.5 getMember()	127
7.18.4 Member Data Documentation	127
7.18.4.1 id	127
7.18.4.2 changeGroupId	127
7.18.4.3 time	127

7.18.4.4 type	128
7.18.4.5 table	128
7.18.4.6 rowId	128
7.19 Bds::ChangeGroup Class Reference	128
7.19.1 Detailed Description	129
7.19.2 Constructor & Destructor Documentation	129
7.19.2.1 ChangeGroup()	129
7.19.3 Member Function Documentation	129
7.19.3.1 getType()	129
7.19.3.2 setMembers()	130
7.19.3.3 setMember()	130
7.19.3.4 getMembers()	130
7.19.3.5 getMember()	130
7.19.4 Member Data Documentation	130
7.19.4.1 id	130
7.19.4.2 time	131
7.19.4.3 type	131
7.19.4.4 user	131
7.19.4.5 title	131
7.19.4.6 description	131
7.20 Bds::Channel Class Reference	132
7.20.1 Detailed Description	133
7.20.2 Constructor & Destructor Documentation	133
7.20.2.1 Channel()	133
7.20.3 Member Function Documentation	133
7.20.3.1 getType()	133
7.20.3.2 setMembers()	133
7.20.3.3 setMember()	134
7.20.3.4 getMembers()	134
7.20.3.5 getMember()	134
7.20.4 Member Data Documentation	134
7.20.4.1 id	134
7.20.4.2 startTime	134
7.20.4.3 endTime	135
7.20.4.4 network	135
7.20.4.5 station	135
7.20.4.6 channel	135
7.20.4.7 channelType	135
7.20.4.8 channelAux	135
7.20.4.9 dataType	136
7.20.4.10 description	136
7.21 Bds::ChannelInfo Class Reference	136

7.21.1 Detailed Description	137
7.21.2 Constructor & Destructor Documentation	137
7.21.2.1 ChannelInfo()	137
7.21.3 Member Data Documentation	137
7.21.3.1 startTime	137
7.21.3.2 endTime	137
7.21.3.3 station	138
7.21.3.4 stationLocation	138
7.21.3.5 channel	138
7.21.3.6 channelLocation	138
7.21.3.7 source	138
7.21.3.8 digitiser	138
7.21.3.9 sensor	139
7.21.3.10 calibration	139
7.21.3.11 responses	139
7.22 Bds::ChannelInfos Class Reference	139
7.22.1 Detailed Description	140
7.22.2 Constructor & Destructor Documentation	140
7.22.2.1 ChannelInfos()	140
7.22.3 Member Data Documentation	140
7.22.3.1 array	140
7.22.3.2 channels	140
7.23 Bds::ChannelInstrument Class Reference	141
7.23.1 Detailed Description	141
7.23.2 Constructor & Destructor Documentation	142
7.23.2.1 ChannelInstrument()	142
7.23.3 Member Function Documentation	142
7.23.3.1 getType()	142
7.23.3.2 setMembers()	142
7.23.3.3 setMember()	142
7.23.3.4 getMembers()	143
7.23.3.5 getMember()	143
7.23.4 Member Data Documentation	143
7.23.4.1 id	143
7.23.4.2 startTime	143
7.23.4.3 endTime	143
7.23.4.4 channelId	144
7.23.4.5 source	144
7.23.4.6 digitiserId	144
7.23.4.7 sensorId	144
7.24 Bds::ChannelName Class Reference	144
7.24.1 Detailed Description	145

7.24.2 Constructor & Destructor Documentation	145
7.24.2.1 ChannelName()	145
7.24.3 Member Data Documentation	145
7.24.3.1 network	145
7.24.3.2 station	145
7.24.3.3 channel	146
7.24.3.4 source	146
7.25 Bds::CleanOptions Class Reference	146
7.25.1 Detailed Description	146
7.25.2 Constructor & Destructor Documentation	146
7.25.2.1 CleanOptions()	147
7.25.3 Member Data Documentation	147
7.25.3.1 logs	147
7.25.3.2 changes	147
7.25.3.3 deletedFiles	147
7.26 Bds::CompressSteim1 Class Reference	147
7.26.1 Detailed Description	148
7.26.2 Constructor & Destructor Documentation	148
7.26.2.1 CompressSteim1()	148
7.26.3 Member Function Documentation	148
7.26.3.1 setByteOrder()	148
7.26.3.2 clear()	148
7.26.3.3 unCompress()	148
7.27 Bds::DataAccess Class Reference	149
7.27.1 Detailed Description	152
7.27.2 Constructor & Destructor Documentation	152
7.27.2.1 DataAccess()	152
7.27.3 Member Function Documentation	152
7.27.3.1 connect()	152
7.27.3.2 validateUser()	153
7.27.3.3 setUser()	153
7.27.3.4 setUserReal()	153
7.27.3.5 getVersion()	153
7.27.3.6 userGetFromId()	154
7.27.3.7 userGet()	154
7.27.3.8 userSet()	154
7.27.3.9 userGetGroups()	154
7.27.3.10 userGetOptions()	154
7.27.3.11 userSetOptions()	155
7.27.3.12 groupGetList()	155
7.27.3.13 networkGetList()	155
7.27.3.14 stationGetList()	155

7.27.3.15 channelGetList()	155
7.27.3.16 sourceGetList()	156
7.27.3.17 sourcePriorityGetList()	156
7.27.3.18 dataFileGetList()	156
7.27.3.19 dataChannelGetList()	157
7.27.3.20 channelInstrumentGetList()	157
7.27.3.21 digitiserGetList()	157
7.27.3.22 digitiserGet()	158
7.27.3.23 sensorGetList()	158
7.27.3.24 sensorGet()	158
7.27.3.25 calibrationGetList()	158
7.27.3.26 responseGetList()	159
7.27.3.27 locationGetList()	159
7.27.3.28 eventGetList()	159
7.27.3.29 specialChannelGetList()	160
7.27.3.30 metadataGetChannelInfo()	160
7.27.3.31 metadataGetFormatted()	160
7.27.3.32 getSelectionInfo()	161
7.27.3.33 getSelections()	161
7.27.3.34 dataAvailability()	162
7.27.3.35 dataSearch()	162
7.27.3.36 dataGetChannelInfo()	163
7.27.3.37 dataOpen()	163
7.27.3.38 dataGetInfo()	164
7.27.3.39 dataGetNotes()	164
7.27.3.40 dataGetWarnings()	165
7.27.3.41 dataSeekBlock()	165
7.27.3.42 dataGetBlock()	166
7.27.3.43 dataClose()	166
7.27.3.44 dataFormattedRead()	166
7.27.3.45 dataFormattedGetLength()	167
7.27.3.46 dataRealtimeConfig()	167
7.27.3.47 dataRealtimeGet()	167
7.27.3.48 noteGetList()	168
7.27.3.49 noteUpdate()	168
7.27.3.50 noteWriteDocument()	168
7.27.3.51 noteReadDocument()	169
7.27.3.52 logUpdate()	169
7.27.3.53 logAppend()	169
7.27.3.54 modeSet()	169
7.27.3.55 modeSnapshotPause()	170
7.27.3.56 clean()	170

7.27.3.57 databaseBackup()	170
7.27.3.58 statisticsGet()	171
7.27.3.59 serverConfigurationGet()	171
7.27.3.60 dataFormatGetList()	171
7.28 Bds::DataAddAccess Class Reference	172
7.28.1 Detailed Description	175
7.28.2 Constructor & Destructor Documentation	175
7.28.2.1 DataAddAccess()	175
7.28.3 Member Function Documentation	176
7.28.3.1 connect()	176
7.28.3.2 validateUser()	176
7.28.3.3 setUser()	176
7.28.3.4 setUserReal()	177
7.28.3.5 getVersion()	177
7.28.3.6 userGetFromId()	177
7.28.3.7 userGet()	177
7.28.3.8 userSet()	177
7.28.3.9 userGetGroups()	178
7.28.3.10 userGetOptions()	178
7.28.3.11 userSetOptions()	178
7.28.3.12 groupGetList()	178
7.28.3.13 networkGetList()	178
7.28.3.14 stationGetList()	178
7.28.3.15 channelGetList()	179
7.28.3.16 sourceGetList()	179
7.28.3.17 sourcePriorityGetList()	179
7.28.3.18 dataFileGetList()	179
7.28.3.19 dataChannelGetList()	180
7.28.3.20 channelInstrumentGetList()	180
7.28.3.21 digitiserGetList()	180
7.28.3.22 digitiserGet()	181
7.28.3.23 sensorGetList()	181
7.28.3.24 sensorGet()	181
7.28.3.25 calibrationGetList()	182
7.28.3.26 responseGetList()	182
7.28.3.27 locationGetList()	182
7.28.3.28 eventGetList()	183
7.28.3.29 eventUpdate()	183
7.28.3.30 eventDelete()	183
7.28.3.31 specialChannelGetList()	183
7.28.3.32 metadataGetChannelInfo()	184
7.28.3.33 metadataGetFormatted()	184

7.28.3.34	getSelectionInfo()	185
7.28.3.35	getSelections()	185
7.28.3.36	dataAvailability()	185
7.28.3.37	dataSearch()	186
7.28.3.38	dataGetChannelInfo()	186
7.28.3.39	dataOpen()	187
7.28.3.40	dataGetInfo()	188
7.28.3.41	dataGetNotes()	188
7.28.3.42	dataGetWarnings()	188
7.28.3.43	dataSeekBlock()	189
7.28.3.44	dataGetBlock()	189
7.28.3.45	dataSetInfo()	190
7.28.3.46	dataPutBlock()	190
7.28.3.47	dataClose()	190
7.28.3.48	dataFormattedRead()	190
7.28.3.49	dataFormattedGetLength()	191
7.28.3.50	dataRealtimeConfig()	191
7.28.3.51	dataRealtimeGet()	191
7.28.3.52	noteGetList()	192
7.28.3.53	noteUpdate()	192
7.28.3.54	noteWriteDocument()	192
7.28.3.55	noteReadDocument()	193
7.28.3.56	logUpdate()	193
7.28.3.57	logAppend()	193
7.28.3.58	modeSet()	193
7.28.3.59	modeSnapshotPause()	194
7.28.3.60	clean()	194
7.28.3.61	databaseBackup()	194
7.28.3.62	statisticsGet()	195
7.28.3.63	serverConfigurationGet()	195
7.28.3.64	dataFormatGetList()	195
7.29	Bds::DataAvail Class Reference	196
7.29.1	Detailed Description	196
7.29.2	Constructor & Destructor Documentation	196
7.29.2.1	DataAvail()	197
7.29.3	Member Data Documentation	197
7.29.3.1	startTime	197
7.29.3.2	endTime	197
7.29.3.3	availType	197
7.30	Bds::DataAvailChan Class Reference	197
7.30.1	Detailed Description	198
7.30.2	Constructor & Destructor Documentation	198

7.30.2.1 DataAvailChan()	198
7.30.3 Member Data Documentation	199
7.30.3.1 startTime	199
7.30.3.2 endTime	199
7.30.3.3 network	199
7.30.3.4 station	199
7.30.3.5 channel	199
7.30.3.6 source	200
7.30.3.7 segments	200
7.31 Bds::DataBlock Class Reference	200
7.31.1 Detailed Description	201
7.31.2 Constructor & Destructor Documentation	201
7.31.2.1 DataBlock()	201
7.31.3 Member Data Documentation	201
7.31.3.1 startTime	201
7.31.3.2 endTime	201
7.31.3.3 channelNumber	202
7.31.3.4 segmentNumber	202
7.31.3.5 channelData	202
7.31.3.6 info	202
7.32 Bds::DataBlockChannel Class Reference	202
7.32.1 Detailed Description	203
7.32.2 Constructor & Destructor Documentation	203
7.32.2.1 DataBlockChannel()	203
7.32.3 Member Data Documentation	203
7.32.3.1 network	203
7.32.3.2 station	203
7.32.3.3 channel	204
7.32.3.4 source	204
7.33 Bds::DataBlockPos Class Reference	204
7.33.1 Detailed Description	204
7.33.2 Constructor & Destructor Documentation	204
7.33.2.1 DataBlockPos()	205
7.33.3 Member Function Documentation	205
7.33.3.1 operator<()	205
7.33.4 Member Data Documentation	205
7.33.4.1 startTime	205
7.33.4.2 endTime	205
7.33.4.3 position	205
7.33.4.4 order	205
7.33.4.5 ref	206
7.33.4.6 numSamples	206

7.34 Bds::DataChannel Class Reference	206
7.34.1 Detailed Description	207
7.34.2 Constructor & Destructor Documentation	207
7.34.2.1 DataChannel()	208
7.34.3 Member Function Documentation	208
7.34.3.1 getType()	208
7.34.3.2 setMembers()	208
7.34.3.3 setMember()	208
7.34.3.4 getMembers()	209
7.34.3.5 getMember()	209
7.34.4 Member Data Documentation	209
7.34.4.1 id	209
7.34.4.2 startTime	209
7.34.4.3 endTime	209
7.34.4.4 network	210
7.34.4.5 station	210
7.34.4.6 channel	210
7.34.4.7 source	210
7.34.4.8 numBlocks	210
7.34.4.9 numSamples	210
7.34.4.10 sampleRate	211
7.34.4.11 sampleFormat	211
7.34.4.12 dataFileId	211
7.34.4.13 dataFileChannel	211
7.34.4.14 importFormat	211
7.34.4.15 importFilename	211
7.34.4.16 importStartTime	212
7.34.4.17 info	212
7.35 Bds::DataCollate Class Reference	212
7.35.1 Detailed Description	212
7.35.2 Constructor & Destructor Documentation	212
7.35.2.1 DataCollate()	212
7.35.2.2 ~DataCollate()	213
7.35.3 Member Function Documentation	213
7.35.3.1 addSource()	213
7.35.3.2 readData()	213
7.36 Bds::DataError Class Reference	213
7.36.1 Detailed Description	215
7.36.2 Constructor & Destructor Documentation	215
7.36.2.1 DataError() [1/2]	215
7.36.2.2 DataError() [2/2]	215
7.36.3 Member Function Documentation	215

7.36.3.1 set()	215
7.36.3.2 mergeDataInfo()	216
7.36.3.3 getErrorNumber()	216
7.36.3.4 getTitle()	216
7.36.3.5 setString()	216
7.36.3.6 setStringUser()	216
7.36.3.7 getString()	216
7.36.3.8 num()	217
7.36.3.9 str()	217
7.36.3.10 operator int()	217
7.36.4 Member Data Documentation	217
7.36.4.1 oerrorNumber	217
7.36.4.2 otitle	217
7.36.4.3 odescription	217
7.36.4.4 ofilename	218
7.36.4.5 ostartTime	218
7.36.4.6 oendTime	218
7.36.4.7 onetwork	218
7.36.4.8 ostation	218
7.36.4.9 ochannel	218
7.36.4.10 osource	219
7.36.4.11 ouser	219
7.37 Bds::DataFile Class Reference	219
7.37.1 Detailed Description	221
7.37.2 Member Enumeration Documentation	221
7.37.2.1 DataOrder	221
7.37.2.2 Features	222
7.37.2.3 WriteOptionsList	222
7.37.2.4 ReadOptionsList	222
7.37.3 Constructor & Destructor Documentation	222
7.37.3.1 DataFile()	223
7.37.3.2 ~DataFile()	223
7.37.4 Member Function Documentation	223
7.37.4.1 init()	223
7.37.4.2 open()	223
7.37.4.3 close()	223
7.37.4.4 setFormat()	224
7.37.4.5 getFileName()	224
7.37.4.6 getDataOrder()	224
7.37.4.7 getFeatures()	224
7.37.4.8 getFixesInfo()	225
7.37.4.9 setInfo()	225

7.37.4.10 start()	225
7.37.4.11 writeData()	226
7.37.4.12 end()	226
7.37.4.13 flush()	226
7.37.4.14 fileNameProcess()	227
7.37.4.15 getFormat()	227
7.37.4.16 getInfo()	227
7.37.4.17 seekBlock()	228
7.37.4.18 readData()	228
7.37.4.19 getMetaData()	229
7.37.4.20 dataErrorFixup()	229
7.37.4.21 timeCompare()	229
7.37.4.22 duplicateCheck()	230
7.37.4.23 getFilePosition()	230
7.37.4.24 getFormats()	230
7.37.5 Member Data Documentation	230
7.37.5.1 ofileName	230
7.37.5.2 omode	230
7.37.5.3 ofileNameTime	230
7.37.5.4 ofile	231
7.37.5.5 oformat	231
7.38 Bds::DataFileAd22 Class Reference	231
7.38.1 Detailed Description	232
7.38.2 Constructor & Destructor Documentation	232
7.38.2.1 DataFileAd22()	232
7.38.3 Member Function Documentation	232
7.38.3.1 getFeatures()	232
7.38.3.2 getDataOrder()	232
7.38.3.3 getFixesInfo()	232
7.38.3.4 getInfo()	232
7.38.3.5 readData()	233
7.38.3.6 getFormats()	233
7.39 Bds::DataFileAscii Class Reference	234
7.39.1 Detailed Description	234
7.39.2 Constructor & Destructor Documentation	234
7.39.2.1 DataFileAscii()	235
7.39.3 Member Function Documentation	235
7.39.3.1 open()	235
7.39.3.2 getDataOrder()	235
7.39.3.3 getFeatures()	235
7.39.3.4 setFormat()	235
7.39.3.5 setInfo()	236

7.39.3.6 start()	236
7.39.3.7 writeData()	237
7.39.3.8 end()	237
7.39.3.9 getFormats()	237
7.40 Bds::DataFileBdrs Class Reference	237
7.40.1 Detailed Description	238
7.40.2 Constructor & Destructor Documentation	238
7.40.2.1 DataFileBdrs()	238
7.40.3 Member Function Documentation	238
7.40.3.1 getFeatures()	238
7.40.3.2 getDataOrder()	239
7.40.3.3 getFixesInfo()	239
7.40.3.4 getInfo()	239
7.40.3.5 readData()	240
7.40.3.6 getFormats()	240
7.41 Bds::DataFileBds Class Reference	240
7.41.1 Detailed Description	241
7.41.2 Member Enumeration Documentation	242
7.41.2.1 anonymous enum	242
7.41.2.2 anonymous enum	242
7.41.2.3 PackFormat	242
7.41.3 Constructor & Destructor Documentation	242
7.41.3.1 DataFileBds()	242
7.41.3.2 ~DataFileBds()	243
7.41.4 Member Function Documentation	243
7.41.4.1 open()	243
7.41.4.2 flush()	243
7.41.4.3 close()	243
7.41.4.4 setFormat()	243
7.41.4.5 setInfo()	244
7.41.4.6 writeData()	244
7.41.4.7 getDataOrder()	244
7.41.4.8 getInfo()	244
7.41.4.9 seekBlock()	245
7.41.4.10 readData()	245
7.41.4.11 getFormats()	245
7.41.4.12 setDiskBlockSize()	245
7.41.4.13 getDiskBlockSize()	245
7.41.4.14 streamletToChannel()	246
7.41.4.15 setWritePositionForAppend()	246
7.41.4.16 setReadPositionToStart()	246
7.41.4.17 packetRead()	246

7.41.4.18 packetWrite()	246
7.42 Bds::DataFileBknas Class Reference	247
7.42.1 Detailed Description	247
7.42.2 Constructor & Destructor Documentation	247
7.42.2.1 DataFileBknas()	247
7.42.3 Member Function Documentation	247
7.42.3.1 open()	247
7.42.3.2 setInfo()	248
7.42.3.3 writeData()	248
7.42.3.4 getFormats()	249
7.43 Bds::DataFileCd Class Reference	249
7.43.1 Detailed Description	249
7.43.2 Constructor & Destructor Documentation	250
7.43.2.1 DataFileCd()	250
7.43.3 Member Function Documentation	250
7.43.3.1 getFeatures()	250
7.43.3.2 getDataOrder()	250
7.43.3.3 getFixesInfo()	250
7.43.3.4 getInfo()	250
7.43.3.5 readData()	251
7.43.3.6 getFormats()	251
7.44 Bds::DataFileCss Class Reference	252
7.44.1 Detailed Description	252
7.44.2 Constructor & Destructor Documentation	252
7.44.2.1 DataFileCss()	252
7.44.3 Member Function Documentation	253
7.44.3.1 getFeatures()	253
7.44.3.2 getDataOrder()	253
7.44.3.3 getInfo()	253
7.44.3.4 readData()	254
7.44.3.5 getFormats()	254
7.45 Bds::DataFileCssData Class Reference	254
7.45.1 Detailed Description	255
7.45.2 Constructor & Destructor Documentation	255
7.45.2.1 DataFileCssData()	255
7.45.2.2 ~DataFileCssData()	256
7.45.3 Member Function Documentation	256
7.45.3.1 set()	256
7.45.4 Member Data Documentation	256
7.45.4.1 sta	256
7.45.4.2 chan	256
7.45.4.3 startTime	256

7.45.4.4 wfid	256
7.45.4.5 chanid	257
7.45.4.6 jdate	257
7.45.4.7 endTime	257
7.45.4.8 nsamp	257
7.45.4.9 sampleRate	257
7.45.4.10 calibrationFactor	257
7.45.4.11 calibrationFreq	257
7.45.4.12 instType	257
7.45.4.13 segtype	258
7.45.4.14 datatype	258
7.45.4.15 clip	258
7.45.4.16 dirName	258
7.45.4.17 fileName	258
7.45.4.18 fileOffset	258
7.45.4.19 commId	258
7.45.4.20 loadDate	258
7.45.4.21 file	259
7.45.4.22 sampleFormat	259
7.45.4.23 sampleSize	259
7.45.4.24 sampleBigEndian	259
7.46 Bds::DataFileGcf Class Reference	259
7.46.1 Detailed Description	260
7.46.2 Constructor & Destructor Documentation	260
7.46.2.1 DataFileGcf()	260
7.46.3 Member Function Documentation	260
7.46.3.1 getFeatures()	260
7.46.3.2 getDataOrder()	261
7.46.3.3 getFixesInfo()	261
7.46.3.4 getInfo()	261
7.46.3.5 readData()	262
7.46.3.6 getFormats()	262
7.47 Bds::DataFileIdc Class Reference	262
7.47.1 Detailed Description	263
7.47.2 Constructor & Destructor Documentation	263
7.47.2.1 DataFileIdc()	263
7.47.3 Member Function Documentation	263
7.47.3.1 getFeatures()	263
7.47.3.2 getMetaData()	264
7.47.3.3 setInfo()	264
7.47.3.4 getFormats()	264
7.48 Bds::DataFileIms Class Reference	265

7.48.1 Detailed Description	265
7.48.2 Constructor & Destructor Documentation	266
7.48.2.1 DataFileIms()	266
7.48.3 Member Function Documentation	266
7.48.3.1 open()	266
7.48.3.2 close()	266
7.48.3.3 getDataOrder()	266
7.48.3.4 getFeatures()	267
7.48.3.5 setInfo()	267
7.48.3.6 start()	267
7.48.3.7 writeData()	268
7.48.3.8 end()	268
7.48.3.9 getMetaData()	268
7.48.3.10 getFormats()	269
7.49 Bds::DataFileInfo Class Reference	269
7.49.1 Detailed Description	270
7.49.2 Constructor & Destructor Documentation	270
7.49.2.1 DataFileInfo()	270
7.49.3 Member Function Documentation	271
7.49.3.1 getType()	271
7.49.3.2 setMembers()	271
7.49.3.3 setMember()	271
7.49.3.4 getMembers()	271
7.49.3.5 getMember()	271
7.49.4 Member Data Documentation	272
7.49.4.1 id	272
7.49.4.2 startTime	272
7.49.4.3 endTime	272
7.49.4.4 location	272
7.49.4.5 format	272
7.49.4.6 url	273
7.49.4.7 stream	273
7.49.4.8 comment	273
7.49.4.9 importUserId	273
7.49.4.10 importTime	273
7.49.4.11 state	273
7.50 Bds::DataFileLac Class Reference	274
7.50.1 Detailed Description	274
7.50.2 Constructor & Destructor Documentation	274
7.50.2.1 DataFileLac()	274
7.50.3 Member Function Documentation	275
7.50.3.1 getFeatures()	275

7.50.3.2	getDataOrder()	275
7.50.3.3	getFixesInfo()	275
7.50.3.4	getInfo()	275
7.50.3.5	readData()	276
7.50.3.6	getFormats()	276
7.51	Bds::DataFileLog Class Reference	276
7.51.1	Detailed Description	277
7.51.2	Constructor & Destructor Documentation	277
7.51.2.1	DataFileLog()	277
7.51.3	Member Function Documentation	277
7.51.3.1	open()	277
7.51.3.2	getDataOrder()	278
7.51.3.3	getFeatures()	278
7.51.3.4	getInfo()	278
7.51.3.5	readData()	279
7.51.3.6	setFormat()	279
7.51.3.7	setInfo()	279
7.51.3.8	start()	280
7.51.3.9	writeData()	280
7.51.3.10	end()	281
7.51.3.11	getFormats()	281
7.52	Bds::DataFileOptions Class Reference	281
7.52.1	Detailed Description	281
7.52.2	Constructor & Destructor Documentation	282
7.52.2.1	DataFileOptions()	282
7.52.3	Member Function Documentation	282
7.52.3.1	operator int()	282
7.52.3.2	operator" =()	282
7.52.4	Member Data Documentation	282
7.52.4.1	ooptionList	282
7.52.4.2	oignoreBlockList	282
7.53	Bds::DataFileResponse Class Reference	283
7.53.1	Detailed Description	283
7.53.2	Constructor & Destructor Documentation	283
7.53.2.1	DataFileResponse()	283
7.53.3	Member Function Documentation	283
7.53.3.1	getFeatures()	284
7.53.3.2	getMetaData()	284
7.53.3.3	setInfo()	284
7.53.3.4	getFormats()	284
7.54	Bds::DataFileSac Class Reference	285
7.54.1	Detailed Description	285

7.54.2 Constructor & Destructor Documentation	285
7.54.2.1 DataFileSac()	285
7.54.3 Member Function Documentation	285
7.54.3.1 getFeatures()	286
7.54.3.2 getMetaData()	286
7.54.3.3 setInfo()	286
7.54.3.4 getFormats()	286
7.55 Bds::DataFileSeed Class Reference	287
7.55.1 Detailed Description	288
7.55.2 Constructor & Destructor Documentation	288
7.55.2.1 DataFileSeed()	288
7.55.2.2 ~DataFileSeed()	288
7.55.3 Member Function Documentation	288
7.55.3.1 close()	288
7.55.3.2 getDataOrder()	289
7.55.3.3 getFeatures()	289
7.55.3.4 getFixesInfo()	289
7.55.3.5 setFormat()	289
7.55.3.6 getInfo()	289
7.55.3.7 readData()	290
7.55.3.8 getMetaData()	290
7.55.3.9 setInfo()	291
7.55.3.10 start()	291
7.55.3.11 writeData()	292
7.55.3.12 end()	292
7.55.3.13 msrFileWrite()	292
7.55.3.14 getFormats()	292
7.55.4 Member Data Documentation	292
7.55.4.1 omsrErr	293
7.55.4.2 onoLock	293
7.56 Bds::DataFileStationXml Class Reference	293
7.56.1 Detailed Description	294
7.56.2 Constructor & Destructor Documentation	294
7.56.2.1 DataFileStationXml()	294
7.56.3 Member Function Documentation	294
7.56.3.1 getFeatures()	294
7.56.3.2 setInfo()	294
7.56.3.3 getMetaData()	295
7.56.3.4 getFormats()	295
7.57 Bds::DataFileTapeDigitiser Class Reference	295
7.57.1 Detailed Description	296
7.57.2 Constructor & Destructor Documentation	296

7.57.2.1 DataFileTypeDigitiser()	296
7.57.3 Member Function Documentation	296
7.57.3.1 open()	296
7.57.3.2 getInfo()	296
7.57.3.3 readData()	297
7.57.3.4 getFormats()	297
7.58 Bds::DataFileWra Class Reference	297
7.58.1 Detailed Description	298
7.58.2 Constructor & Destructor Documentation	298
7.58.2.1 DataFileWra()	298
7.58.3 Member Function Documentation	298
7.58.3.1 setFormat()	298
7.58.3.2 getFeatures()	299
7.58.3.3 getDataOrder()	299
7.58.3.4 getFixesInfo()	299
7.58.3.5 getInfo()	299
7.58.3.6 readData()	300
7.58.3.7 getFormats()	300
7.59 Bds::DataFileWraAgso Class Reference	301
7.59.1 Detailed Description	301
7.59.2 Constructor & Destructor Documentation	301
7.59.2.1 DataFileWraAgso()	301
7.59.3 Member Function Documentation	302
7.59.3.1 getFeatures()	302
7.59.3.2 getDataOrder()	302
7.59.3.3 getInfo()	302
7.59.3.4 readData()	303
7.59.3.5 getFormats()	303
7.60 Bds::DataFormat Class Reference	303
7.60.1 Detailed Description	304
7.60.2 Constructor & Destructor Documentation	304
7.60.2.1 DataFormat()	304
7.60.3 Member Data Documentation	304
7.60.3.1 names	304
7.60.3.2 dataRead	305
7.60.3.3 dataWrite	305
7.60.3.4 metadataRead	305
7.60.3.5 metadataWrite	305
7.60.3.6 extension	305
7.60.3.7 description	305
7.61 Bds::DataFormatAll Class Reference	306
7.61.1 Detailed Description	306

7.61.2 Constructor & Destructor Documentation	306
7.61.2.1 DataFormatAll()	306
7.61.2.2 ~DataFormatAll()	306
7.61.3 Member Function Documentation	306
7.61.3.1 formatList()	307
7.61.3.2 formatGet()	307
7.61.3.3 formatGetExtension()	307
7.61.3.4 findFormat()	307
7.62 Bds::DataHandle Class Reference	307
7.62.1 Detailed Description	308
7.62.2 Constructor & Destructor Documentation	308
7.62.2.1 DataHandle()	308
7.62.3 Member Data Documentation	308
7.62.3.1 handle	308
7.62.3.2 dataFileId	308
7.63 Bds::DataInfo Class Reference	309
7.63.1 Detailed Description	309
7.63.2 Constructor & Destructor Documentation	309
7.63.2.1 DataInfo()	310
7.63.3 Member Data Documentation	310
7.63.3.1 startTime	310
7.63.3.2 endTime	310
7.63.3.3 array	310
7.63.3.4 description	310
7.63.3.5 synchronous	311
7.63.3.6 channels	311
7.63.3.7 info	311
7.63.3.8 infoExtra	311
7.63.3.9 warnings	311
7.64 Bds::Digitiser Class Reference	312
7.64.1 Detailed Description	313
7.64.2 Constructor & Destructor Documentation	313
7.64.2.1 Digitiser()	313
7.64.3 Member Function Documentation	313
7.64.3.1 getType()	313
7.64.3.2 setMembers()	313
7.64.3.3 setMember()	314
7.64.3.4 getMembers()	314
7.64.3.5 getMember()	314
7.64.4 Member Data Documentation	314
7.64.4.1 id	314
7.64.4.2 startTime	314

7.64.4.3 endTime	315
7.64.4.4 name	315
7.64.4.5 type	315
7.64.4.6 serialNumber	315
7.64.4.7 numberChannels	315
7.64.4.8 baseSamplingFrequency	315
7.64.4.9 initialSamplingFrequency	316
7.64.4.10 gain	316
7.64.4.11 shared	316
7.65 Bds::Event Class Reference	316
7.65.1 Detailed Description	317
7.65.2 Constructor & Destructor Documentation	318
7.65.2.1 Event()	318
7.65.3 Member Data Documentation	318
7.65.3.1 id	318
7.65.3.2 userId	318
7.65.3.3 type	319
7.65.3.4 title	319
7.65.3.5 network	319
7.65.3.6 source	319
7.65.3.7 startTime	319
7.65.3.8 endTime	319
7.65.3.9 eventTime	320
7.65.3.10 longitude	320
7.65.3.11 latitude	320
7.65.3.12 elevation	320
7.65.3.13 waterDepth	320
7.65.3.14 magnitude	320
7.65.3.15 magnitudeUnits	321
7.65.3.16 description	321
7.65.3.17 notes	321
7.65.3.18 extra	321
7.65.3.19 dataChannels	321
7.66 Bds::Fap Class Reference	321
7.66.1 Detailed Description	322
7.66.2 Constructor & Destructor Documentation	322
7.66.2.1 Fap()	322
7.66.3 Member Data Documentation	322
7.66.3.1 frequency	322
7.66.3.2 amplitude	322
7.66.3.3 phase	323
7.67 Bds::Fir Class Reference	323

7.67.1 Detailed Description	323
7.67.2 Constructor & Destructor Documentation	323
7.67.2.1 Fir()	323
7.67.3 Member Data Documentation	324
7.67.3.1 b	324
7.67.3.2 a	324
7.68 Bds::FirEntry Class Reference	324
7.68.1 Detailed Description	324
7.68.2 Constructor & Destructor Documentation	325
7.68.2.1 FirEntry()	325
7.68.3 Member Data Documentation	325
7.68.3.1 coefficient	325
7.68.3.2 error	325
7.69 Bds::GcfChannel Struct Reference	325
7.69.1 Detailed Description	326
7.69.2 Member Data Documentation	326
7.69.2.1 systemId	326
7.69.2.2 streamId	326
7.69.2.3 type	326
7.69.2.4 sampleRate	326
7.69.2.5 format	326
7.69.2.6 channel	326
7.70 Bds::Group Class Reference	327
7.70.1 Detailed Description	327
7.70.2 Constructor & Destructor Documentation	327
7.70.2.1 Group()	327
7.70.3 Member Function Documentation	328
7.70.3.1 getType()	328
7.70.3.2 setMembers()	328
7.70.3.3 setMember()	328
7.70.3.4 getMembers()	328
7.70.3.5 getMember()	328
7.70.4 Member Data Documentation	329
7.70.4.1 id	329
7.70.4.2 group	329
7.70.4.3 description	329
7.71 Bds::ListRange Class Reference	329
7.71.1 Detailed Description	330
7.71.2 Constructor & Destructor Documentation	330
7.71.2.1 ListRange()	330
7.71.3 Member Function Documentation	330
7.71.3.1 getType()	330

7.71.3.2 setMembers()	331
7.71.3.3 setMember()	331
7.71.3.4 getMembers()	331
7.71.3.5 getMember()	331
7.71.4 Member Data Documentation	331
7.71.4.1 start	331
7.71.4.2 number	332
7.71.4.3 reverse	332
7.72 Bds::Location Class Reference	332
7.72.1 Detailed Description	333
7.72.2 Constructor & Destructor Documentation	333
7.72.2.1 Location()	334
7.72.3 Member Function Documentation	334
7.72.3.1 getType()	334
7.72.3.2 setMembers()	334
7.72.3.3 setMember()	334
7.72.3.4 getMembers()	334
7.72.3.5 getMember()	335
7.72.4 Member Data Documentation	335
7.72.4.1 id	335
7.72.4.2 startTime	335
7.72.4.3 endTime	335
7.72.4.4 network	335
7.72.4.5 station	336
7.72.4.6 channel	336
7.72.4.7 datum	336
7.72.4.8 longitude	336
7.72.4.9 latitude	336
7.72.4.10 elevation	336
7.72.4.11 arrayOffsetEast	337
7.72.4.12 arrayOffsetNorth	337
7.73 Bds::Log Class Reference	337
7.73.1 Detailed Description	338
7.73.2 Constructor & Destructor Documentation	338
7.73.2.1 Log()	338
7.73.3 Member Function Documentation	338
7.73.3.1 getType()	338
7.73.3.2 setMembers()	339
7.73.3.3 setMember()	339
7.73.3.4 getMembers()	339
7.73.3.5 getMember()	339
7.73.4 Member Data Documentation	339

7.73.4.1 id	339
7.73.4.2 time	340
7.73.4.3 type	340
7.73.4.4 priority	340
7.73.4.5 subSystem	340
7.73.4.6 title	340
7.73.4.7 description	340
7.74 Bds::LogSelect Class Reference	341
7.74.1 Detailed Description	341
7.74.2 Constructor & Destructor Documentation	341
7.74.2.1 LogSelect()	341
7.74.3 Member Data Documentation	341
7.74.3.1 startTime	342
7.74.3.2 type	342
7.74.3.3 priority	342
7.74.3.4 subSystem	342
7.75 Bds::Network Class Reference	342
7.75.1 Detailed Description	343
7.75.2 Constructor & Destructor Documentation	343
7.75.2.1 Network()	343
7.75.3 Member Function Documentation	343
7.75.3.1 getType()	343
7.75.3.2 setMembers()	344
7.75.3.3 setMember()	344
7.75.3.4 getMembers()	344
7.75.3.5 getMember()	344
7.75.4 Member Data Documentation	344
7.75.4.1 id	344
7.75.4.2 network	345
7.75.4.3 description	345
7.75.4.4 stations	345
7.76 Bds::Note Class Reference	345
7.76.1 Detailed Description	346
7.76.2 Constructor & Destructor Documentation	346
7.76.2.1 Note()	347
7.76.3 Member Function Documentation	347
7.76.3.1 getType()	347
7.76.3.2 setMembers()	347
7.76.3.3 setMember()	347
7.76.3.4 getMembers()	348
7.76.3.5 getMember()	348
7.76.4 Member Data Documentation	348

7.76.4.1 id	348
7.76.4.2 startTime	348
7.76.4.3 endTime	348
7.76.4.4 network	349
7.76.4.5 station	349
7.76.4.6 channel	349
7.76.4.7 source	349
7.76.4.8 type	349
7.76.4.9 user	349
7.76.4.10 timeAdded	350
7.76.4.11 errorNumber	350
7.76.4.12 title	350
7.76.4.13 description	350
7.76.4.14 docFormat	350
7.76.4.15 docUrl	350
7.76.4.16 dataFileId	351
7.76.4.17 importFilename	351
7.76.4.18 eventId	351
7.77 Bds::Point Class Reference	351
7.77.1 Detailed Description	351
7.77.2 Constructor & Destructor Documentation	352
7.77.2.1 Point()	352
7.77.3 Member Data Documentation	352
7.77.3.1 x	352
7.77.3.2 y	352
7.78 Bds::PoleZero Class Reference	352
7.78.1 Detailed Description	353
7.78.2 Constructor & Destructor Documentation	353
7.78.2.1 PoleZero()	353
7.78.3 Member Data Documentation	353
7.78.3.1 poles	353
7.78.3.2 zeros	353
7.79 Bds::Polynomial Class Reference	353
7.79.1 Detailed Description	354
7.79.2 Constructor & Destructor Documentation	354
7.79.2.1 Polynomial()	354
7.79.3 Member Data Documentation	355
7.79.3.1 transferType	355
7.79.3.2 approximationType	355
7.79.3.3 validFrequencyUnits	355
7.79.3.4 frequencyLowerBound	355
7.79.3.5 frequencyUpperBound	355

7.79.3.6 approximationLowerBound	356
7.79.3.7 approximationUpperBound	356
7.79.3.8 maximumError	356
7.79.3.9 coefficients	356
7.80 Bds::PolynomialEntry Class Reference	356
7.80.1 Detailed Description	357
7.80.2 Constructor & Destructor Documentation	357
7.80.2.1 PolynomialEntry()	357
7.80.3 Member Data Documentation	357
7.80.3.1 coefficient	357
7.80.3.2 plusError	357
7.80.3.3 minusError	358
7.80.3.4 measurementMethod	358
7.81 Bds::Response Class Reference	358
7.81.1 Detailed Description	360
7.81.2 Constructor & Destructor Documentation	360
7.81.2.1 Response()	360
7.81.3 Member Data Documentation	361
7.81.3.1 id	361
7.81.3.2 startTime	361
7.81.3.3 endTime	361
7.81.3.4 network	361
7.81.3.5 station	362
7.81.3.6 channel	362
7.81.3.7 source	362
7.81.3.8 stage	362
7.81.3.9 name	362
7.81.3.10 type	362
7.81.3.11 poleZeros	363
7.81.3.12 faps	363
7.81.3.13 fir	363
7.81.3.14 polynomial	363
7.81.3.15 gain	363
7.81.3.16 gainFrequency	363
7.81.3.17 stageType	364
7.81.3.18 decimation	364
7.81.3.19 decimationOffset	364
7.81.3.20 decimationDelay	364
7.81.3.21 decimationCorr	364
7.81.3.22 symmetry	364
7.81.3.23 description	365
7.81.3.24 measured	365

7.81.3.25 sampleRate	365
7.81.3.26 inputUnits	365
7.81.3.27 inputUnitsDesc	365
7.81.3.28 outputUnits	365
7.81.3.29 outputUnitsDesc	366
7.82 Bds::ResponseObj Class Reference	366
7.82.1 Detailed Description	366
7.82.2 Constructor & Destructor Documentation	366
7.82.2.1 ResponseObj()	366
7.82.2.2 ~ResponseObj()	367
7.82.3 Member Function Documentation	367
7.82.3.1 getString()	367
7.82.3.2 setString()	367
7.83 Bds::Selection Class Reference	367
7.83.1 Detailed Description	368
7.83.2 Constructor & Destructor Documentation	368
7.83.2.1 Selection()	369
7.83.3 Member Data Documentation	369
7.83.3.1 id	369
7.83.3.2 range	369
7.83.3.3 startTime	369
7.83.3.4 endTime	370
7.83.3.5 channels	370
7.83.3.6 channelId	370
7.83.3.7 digitiserId	370
7.83.3.8 sensorId	370
7.83.3.9 sensorOldId	370
7.83.3.10 completeSegments	371
7.83.3.11 calibrationName	371
7.83.3.12 array	371
7.83.3.13 eventId	371
7.83.3.14 name	371
7.83.3.15 locationSelect	371
7.83.3.16 dataTypes	372
7.83.3.17 excludeChannels	372
7.84 Bds::SelectionChannel Class Reference	372
7.84.1 Detailed Description	372
7.84.2 Constructor & Destructor Documentation	372
7.84.2.1 SelectionChannel()	373
7.84.3 Member Data Documentation	373
7.84.3.1 network	373
7.84.3.2 station	373

7.84.3.3 channel	373
7.84.3.4 source	373
7.85 Bds::SelectionInfo Class Reference	373
7.85.1 Detailed Description	374
7.85.2 Constructor & Destructor Documentation	374
7.85.2.1 SelectionInfo()	374
7.85.3 Member Data Documentation	375
7.85.3.1 startTime	375
7.85.3.2 endTime	375
7.85.3.3 networks	375
7.85.3.4 arrays	375
7.85.3.5 stations	375
7.85.3.6 arraysAndStations	376
7.85.3.7 channels	376
7.85.3.8 sources	376
7.85.3.9 numDataChannels	376
7.86 Bds::Sensor Class Reference	376
7.86.1 Detailed Description	377
7.86.2 Constructor & Destructor Documentation	377
7.86.2.1 Sensor()	378
7.86.3 Member Function Documentation	378
7.86.3.1 getType()	378
7.86.3.2 setMembers()	378
7.86.3.3 setMember()	378
7.86.3.4 getMembers()	378
7.86.3.5 getMember()	379
7.86.4 Member Data Documentation	379
7.86.4.1 id	379
7.86.4.2 startTime	379
7.86.4.3 endTime	379
7.86.4.4 name	379
7.86.4.5 type	380
7.86.4.6 serialNumber	380
7.86.4.7 numberChannels	380
7.86.4.8 gainUnits	380
7.86.4.9 gain	380
7.86.4.10 oldId	380
7.86.4.11 shared	381
7.87 Bds::Source Class Reference	381
7.87.1 Detailed Description	382
7.87.2 Constructor & Destructor Documentation	382
7.87.2.1 Source()	382

7.87.3 Member Function Documentation	382
7.87.3.1 getType()	382
7.87.3.2 setMembers()	382
7.87.3.3 setMember()	382
7.87.3.4 getMembers()	383
7.87.3.5 getMember()	383
7.87.4 Member Data Documentation	383
7.87.4.1 id	383
7.87.4.2 source	383
7.87.4.3 sourceMeta	383
7.87.4.4 alias	384
7.87.4.5 description	384
7.88 Bds::SourcePriority Class Reference	384
7.88.1 Detailed Description	385
7.88.2 Constructor & Destructor Documentation	385
7.88.2.1 SourcePriority()	385
7.88.3 Member Function Documentation	385
7.88.3.1 getType()	385
7.88.3.2 setMembers()	386
7.88.3.3 setMember()	386
7.88.3.4 getMembers()	386
7.88.3.5 getMember()	386
7.88.4 Member Data Documentation	386
7.88.4.1 id	386
7.88.4.2 startTime	387
7.88.4.3 endTime	387
7.88.4.4 source	387
7.88.4.5 priority	387
7.89 Bds::SpecialChannel Class Reference	387
7.89.1 Detailed Description	388
7.89.2 Constructor & Destructor Documentation	388
7.89.2.1 SpecialChannel()	388
7.89.3 Member Function Documentation	389
7.89.3.1 getType()	389
7.89.3.2 setMembers()	389
7.89.3.3 setMember()	389
7.89.3.4 getMembers()	389
7.89.3.5 getMember()	389
7.89.4 Member Data Documentation	390
7.89.4.1 id	390
7.89.4.2 startTime	390
7.89.4.3 endTime	390

7.89.4.4 network	390
7.89.4.5 station	390
7.89.4.6 channel	391
7.89.4.7 dataType	391
7.89.4.8 description	391
7.90 Bds::Station Class Reference	391
7.90.1 Detailed Description	392
7.90.2 Constructor & Destructor Documentation	392
7.90.2.1 Station()	392
7.90.3 Member Data Documentation	392
7.90.3.1 id	392
7.90.3.2 network	392
7.90.3.3 name	392
7.90.3.4 alias	393
7.90.3.5 type	393
7.90.3.6 description	393
7.90.3.7 channels	393
7.91 Bds::TimePeriod Class Reference	393
7.91.1 Detailed Description	394
7.91.2 Constructor & Destructor Documentation	394
7.91.2.1 TimePeriod()	394
7.91.3 Member Function Documentation	394
7.91.3.1 getType()	394
7.91.3.2 setMembers()	395
7.91.3.3 setMember()	395
7.91.3.4 getMembers()	395
7.91.3.5 getMember()	395
7.91.4 Member Data Documentation	395
7.91.4.1 startTime	395
7.91.4.2 endTime	396
7.92 Bds::User Class Reference	396
7.92.1 Detailed Description	397
7.92.2 Constructor & Destructor Documentation	397
7.92.2.1 User()	397
7.92.3 Member Function Documentation	397
7.92.3.1 getType()	397
7.92.3.2 setMembers()	397
7.92.3.3 setMember()	397
7.92.3.4 getMembers()	398
7.92.3.5 getMember()	398
7.92.4 Member Data Documentation	398
7.92.4.1 id	398

7.92.4.2 user	398
7.92.4.3 password	398
7.92.4.4 name	399
7.92.4.5 email	399
7.92.4.6 telephone	399
7.92.4.7 address	399
7.92.4.8 enabled	399
7.92.4.9 groups	399
8 File Documentation	401
8.1 /src/blacknest/bds/bds/bdsDataLib/BdsCompress.cpp File Reference	401
8.2 /src/blacknest/bds/bds/bdsDataLib/BdsCompress.d File Reference	401
8.3 /src/blacknest/bds/bds/bdsDataLib/BdsCompress.h File Reference	401
8.4 BdsCompress.h	402
8.5 /src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.cpp File Reference	402
8.6 /src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.d File Reference	403
8.7 /src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.h File Reference	403
8.8 BdsDataCollate.h	403
8.9 /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.cpp File Reference	403
8.10 /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.d File Reference	404
8.11 /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h File Reference	404
8.12 BdsDataFile.h	404
8.13 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.cpp File Reference	405
8.13.1 Macro Definition Documentation	406
8.13.1.1 DEBUG_VELATRACK	406
8.14 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.d File Reference	406
8.15 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.h File Reference	406
8.16 BdsDataFileAd22.h	407
8.17 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.cpp File Reference	407
8.18 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.d File Reference	408
8.19 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.h File Reference	408
8.20 BdsDataFileAscii.h	408
8.21 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.cpp File Reference	409
8.22 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.d File Reference	409
8.23 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.h File Reference	409
8.24 BdsDataFileBdrs.h	409
8.25 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.cpp File Reference	410
8.25.1 Macro Definition Documentation	411
8.25.1.1 LDEBUG	411
8.25.1.2 LDEBUG2	411
8.25.1.3 LDEBUG3	411
8.25.1.4 dlprintf	411

8.25.1.5 dl2printf	411
8.25.1.6 dl3printf	411
8.25.1.7 ALLOW_TIMESTAMP_JITTER	412
8.25.1.8 TIMESTAMP_JITTER	412
8.26 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.d File Reference	412
8.27 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h File Reference	412
8.28 BdsDataFileBds.h	413
8.29 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.cpp File Reference	415
8.29.1 Function Documentation	416
8.29.1.1 clip()	416
8.30 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.d File Reference	416
8.31 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.h File Reference	416
8.32 BdsDataFileBknas.h	416
8.33 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.cpp File Reference	417
8.33.1 Macro Definition Documentation	418
8.33.1.1 LDEBUG	418
8.33.1.2 dprintf	418
8.33.1.3 INCLUDE_CHANNEL_AUTH	418
8.33.1.4 ALLOW_TIMESTAMP_JITTER	418
8.33.1.5 TIMESTAMP_JITTER	418
8.33.1.6 MULTIPLE_SEGMENT	418
8.33.1.7 SEGMENT_GAP	418
8.33.1.8 ntohs	419
8.33.1.9 htons	419
8.33.2 Variable Documentation	419
8.33.2.1 ErrorFormatNoDataFormat	419
8.34 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.d File Reference	419
8.35 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h File Reference	419
8.36 BdsDataFileCd.h	420
8.37 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.cpp File Reference	421
8.38 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.d File Reference	421
8.39 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.h File Reference	421
8.40 BdsDataFileCss.h	422
8.41 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.cpp File Reference	423
8.41.1 Macro Definition Documentation	423
8.41.1.1 DEBUG	423
8.41.1.2 TEST_REORDER	423
8.42 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.d File Reference	423
8.43 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.h File Reference	423
8.44 BdsDataFileGcf.h	424
8.45 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.cpp File Reference	424
8.45.1 Macro Definition Documentation	425

8.45.1.1 LDEBUG	425
8.45.1.2 dprintf	425
8.46 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.d File Reference	425
8.47 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.h File Reference	425
8.48 BdsDataFileIdc.h	426
8.49 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIm.s.cpp File Reference	426
8.50 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIm.s.d File Reference	427
8.51 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIm.s.h File Reference	427
8.52 BdsDataFileIm.s.h	427
8.53 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.cpp File Reference	428
8.54 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.d File Reference	428
8.55 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.h File Reference	428
8.56 BdsDataFileLac.h	429
8.57 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.cpp File Reference	429
8.58 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.d File Reference	430
8.59 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.h File Reference	430
8.60 BdsDataFileLog.h	430
8.61 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.cpp File Reference	431
8.61.1 Macro Definition Documentation	431
8.61.1.1 LDEBUG	431
8.61.1.2 dprintf	431
8.62 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.d File Reference	431
8.63 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.h File Reference	431
8.64 BdsDataFileResponse.h	432
8.65 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.cpp File Reference	432
8.65.1 Macro Definition Documentation	433
8.65.1.1 BDEBUGL1	433
8.66 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.d File Reference	433
8.67 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.h File Reference	433
8.68 BdsDataFileSac.h	434
8.69 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.cpp File Reference	434
8.69.1 Macro Definition Documentation	435
8.69.1.1 BDEBUGL1	435
8.69.1.2 BDEBUGL2	435
8.70 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.d File Reference	435
8.71 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.h File Reference	435
8.72 BdsDataFileStationXml.h	436
8.73 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.cpp File Reference	436
8.74 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.d File Reference	437
8.75 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.h File Reference	437
8.76 BdsDataFileTapeDigitiser.h	437
8.77 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.cpp File Reference	438

8.78 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.d File Reference	438
8.79 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.h File Reference	438
8.80 BdsDataFileWra.h	439
8.81 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.cpp File Reference	439
8.81.1 Function Documentation	439
8.81.1.1 parseStringFixedFields()	440
8.82 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.d File Reference	440
8.83 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.h File Reference	440
8.84 BdsDataFileWraAgso.h	440
8.85 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.cpp File Reference	441
8.86 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.d File Reference	441
8.87 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.h File Reference	441
8.88 BdsDataLib.h	442
8.89 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.cpp File Reference	442
8.89.1 Macro Definition Documentation	443
8.89.1.1 BDEBUGL1	443
8.89.1.2 BDEBUGL2	443
8.89.1.3 BDEBUGL3	443
8.89.1.4 DEBUG	444
8.89.1.5 DEBUG_BLOCKETTE	444
8.89.1.6 DEBUG_BLOCKS	444
8.89.1.7 FILL_BLOCKS	444
8.89.1.8 ROUND_TIMESTAMPS_US	444
8.90 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.d File Reference	444
8.91 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.h File Reference	444
8.91.1 Typedef Documentation	445
8.91.1.1 MSRecord	445
8.92 BdsDataFileSeed.h	445
8.93 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.cpp File Reference	447
8.94 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.d File Reference	447
8.95 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.h File Reference	447
8.96 BdsSeedType.h	447
8.97 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.cpp File Reference	448
8.98 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.d File Reference	448
8.99 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.idl File Reference	448
8.100 /src/blacknest/bds/bds/bdsDataLib/canada_compress.d File Reference	448
8.101 /src/blacknest/bds/bds/bdsDataLib/canada_compress.h File Reference	448
8.101.1 Macro Definition Documentation	448
8.101.1.1 CANCOMP_ERR	449
8.101.1.2 CANCOMP_SUCCESS	449
8.101.1.3 CANCOMP_NOT_20	449
8.101.1.4 CANCOMP_CORRUPT	449

8.101.1.5 CANCOMP_EXCEED	449
8.101.2 Function Documentation	449
8.101.2.1 canada_uncompress()	449
8.101.2.2 canada_compress()	450
8.102 canada_compress.h	450
8.103 BdsC.cc File Reference	450
8.104 BdsC.d File Reference	451
8.105 BdsC.h File Reference	451
8.106 BdsC.h	451
8.107 BdsD.cc File Reference	455
8.108 BdsD.d File Reference	455
8.109 BdsD.h File Reference	455
8.109.1 Detailed Description	458
8.110 BdsD.h	459
8.111 BdsLib.cpp File Reference	469
8.112 BdsLib.d File Reference	471
8.113 BdsLib.dox File Reference	471
8.114 BdsLib.h File Reference	471
8.114.1 Detailed Description	474
8.115 BdsLib.h	474
8.116 BdsS.cc File Reference	475
8.117 BdsS.d File Reference	476
8.118 BdsT.cc File Reference	476
8.119 /src/blacknest/bds/bds/doc/bdsApi-extra.dox File Reference	476
8.120 /src/blacknest/bds/bds/doc/bdsApiOverview.dox File Reference	476
8.121 /src/blacknest/bds/bds/doc/BdsPython.dox File Reference	476
Index	477

Chapter 1

BdsApi

Author

Dr Terry Barnaby

Version

3.1.0

Date

2022-11-21

1.1 Introduction

This documentation provides detailed reference information for the BEAM BdsApi software API of the Blacknest Data System (BDS). The API provides the ability to store and access seismic sensor data and metadata as well as administer the BDS system. The API is an object orientated API implemented in 'C++' with a number of object classes. It also has bindings for other languages which include Python and PHP.

The API operates over a network interface using an RPC type mechanism implemented by BEAM's BOAP RPC system. The BdsApi API makes use of the BEAM Beamlib 'C++' class library for lower level and system independent functionality. The BEAM Beamlib 'C++' class library provides a small set of low level 'C++' classes for strings, lists and system interface functions. It also implements the BOAP RPC mechanism used to implement the BdsApi. There is some brief information on the BEAM class library later on in this page and the API description is available in the Beamlib documentation.

The BDS Python API is built on top of the standard BDS 'C++' API using the SWIG API generator. Thus all of the standard BDS C++ API documentation applies however there are some differences due to the language facility and syntax differences. The core difference is when returning data from functions. With C++ you can return data by passing references or pointers to objects. In Python this is not generally possible and so objects are returned at the left hand side of functions instead.

This is the reference documentation for the BdsApi. An overall API description and programming manual is provided separately in: [BdsDevelopment.pdf](#)

1.2 Overview

The BdsApi has been developed using the BOAP (BEAM Object Access Protocol). This provides a simple but powerful Object Orientated RPC mechanism using an efficient binary protocol. The BdsApi is written in a high level interface definition language (IDL). The Beamlib bidl tool generates the client and server side 'C++' interface and implementation files for the API. These are then provided as a set of 'C++' header files and a binary library file for the clients and server programs to link to. The BOAP system employs a simple BOAP name server process that provides a translation between object names and network IPAddress/Socket numbers. The BOAP name server runs on the main BDS Server host. More information on the BOAP system can be found in the Beamlib documentation.

The object orientated BDS API implements a number of data storage classes and three BdsServer interface objects. The interface objects are:

1. **Bds::DataAccess** BDS Data API: This provides read only access to the data and meta data. It is used by the AutoDRM email and Web systems as well as for user and general program access to the data.
2. **Bds::DataAddAccess** BDS DataAdd API: This provides read and restricted write access to enable the adding of data to the system. It will not allow deletions of data to be performed. It is designed to be used by manual and automatic data adding programs.
3. **Bds::AdminAccess** BDS Admin API: This provides full read/write access to the data and meta data as well as administrative configuration information.

These access API's are related in that the DataAddAccess API is a subset of the AdminAccess API and the Data↔Access API is a subset of the DataAddAccess API. These API access objects should be consulted to view the functionality provided by the BDS system API's.

This documentation is automatically generated from the BDS source code and describes the classes and their member functions. For higher level information on how to use these please consult the [BdsDevelopment.↔pdf](#) document..

1.3 C++ Examples

There are some examples of client applications using the BdsApi in the **bdsExamples** directory of the source code. Some simple Data Access client examples are listed below:

```

/*****
 * BdsDataClient1.cpp BDS API example code for a Data Client
 * T.Barnaby, BEAM Ltd, 2008-09-02
 *****/

 * This is a very basic example of using the BdsApi from a data access
 * perspective. It is designed to give an overview of using the API.
 * This program gets data in the BKNAS format.
 */
#include <iostream>
#include <stdio.h>
#include <BdsD.h>
#include <BdsC.h>
using namespace Bds;
using namespace std;
// Function to read some data
BError bdsTest(DataAccess& bds){
    BError          err;
    Selection       selection;
    DataInfo        dataInfo;
    DataHandle      dataHandle;
    BArray<BUInt8>  data;

    // Set up selection
#ifdef ZAP
    selection.startTime.setString("2002-01-01T00:00:00.000000");
    selection.endTime.setString("2002-01-01T00:01:00.000000");
#else
    selection.startTime.setString("2002-01-01T23:59:00.000000");

```



```

        selection.endTime.setString("2002-01-02T00:01:00.000000");
    #endif
    selection.channels.append(SelectionChannel("BN", "EKA", "", ""));
    // Get list of all data available for the selection
    if(err = bds.dataSearch(selection, dataInfo)){
        return err.set(1, BString("Error: Searching for data: ") + err.getString());
    }
    // We should now choose which set of data we would like from the list, here we just
    // choose the first entry and get the data in appropriate format.
    if(!dataInfo.channels.size())
        return err.set(1, "No data found");

    if(err = bds.dataOpen(dataInfo, "r", "IMS", 0, dataHandle)){
        return err;
    }

    while(1){
        if(err = bds.dataFormattedRead(dataHandle, 1024, data)){
            return err;
        }
        if(data.size() == 0)
            break;

        fwrite(data.data(), 1, data.size(), stdout);
    }

    return err;
}

int main(int argc, char** argv){
    BError      err;
    BString      hostName;
    DataAccess   bds;
    hostName = getenv("BDS_HOST");
    if(hostName == "")
        hostName = "localhost";

    if(argc == 2)
        hostName = argv[1];
    // Connect to the DataAccess service
    if(err = bds.connectService(BString("/") + hostName + "/bdsDataAccess")){
        cerr << "Error: " << err.getString() << "\n";
        return 1;
    }
    // Connect to service
    if(!err)
        err = bds.connect("test", "beam00");

    // Run a normal data gathering as a normal data access client would.
    if(!err)
        err = bdsTest(bds);

    if(err){
        cerr << "Error: " << err.getString() << "\n";
        return 1;
    }

    return 0;
}

/*****
 * BdsDataClient2.cpp BDS API example code for a Data Client
 * T.Barnaby, BEAM Ltd, 2008-09-02
 *****/
*
* This is a very basic example of using the BdsApi from a data access
* perspective. It is designed to give an overview of using the API.
* This program gets data in raw format and outputs it in ASCII.
*/
#include <iostream>
#include <iomanip>
#include <stdio.h>
#include <BdsD.h>
#include <BdsC.h>
using namespace Bds;
using namespace std;
// Function to read some data
BError bdsTest(DataAccess& bds){
    BError      err;
    Selection    selection;
    DataInfo     dataInfo;
    DataHandle    dataHandle;
    DataBlock     data;
    BUInt32      blockNumber = 0;
    BUInt        c;
    BUInt        s;

    // Set up selection
    selection.startTime.setString("2008-01-03T00:00:00.000000");

```

```

selection.endTime.setString("2008-01-03T00:01:00.000000");
selection.channels.append(SelectionChannel("BN", "EKA", "", ""));
// Get list of all data available for the selection
if(err = bds.dataSearch(selection, dataInfo)){
    return err.set(1, BString("Error: Searching for data: ") + err.getString());
}
// We should now choose which set of data we would like from the list, here we just
// choose the first entry and get the data in appropriate format.
if(!dataInfo.channels.size())
    return err.set(1, "No data found");

if(err = bds.dataOpen(dataInfo, "r", "API", 0, dataHandle)){
    return err;
}

while(1){
    if(err = bds.dataGetBlock(dataHandle, 0, 0, blockNumber, data)){
        return err;
    }

    if(data.startTime >= dataInfo.endTime)
        break;
    for(s = 0; s < data.channelData[0].size(); s++){
        for(c = 0; c < data.channelData.size(); c++){
            if(c != 0)
                std::cout << ", ";
            std::cout << setw(8) << data.channelData[c][s];
        }
        std::cout << "\n";
    }
    blockNumber++;
}

return err;
}

int main(int argc, char** argv){
    BError          err;
    BString          hostName;
    DataAccess       bds;
    hostName = getenv("BDS_HOST");
    if(hostName == "")
        hostName = "localhost";

    if(argc == 2)
        hostName = argv[1];
    // Connect to the DataAccess service
    if(err = bds.connectService(BString("//") + hostName + "/bdsDataAccess")){
        cerr << "Error: " << err.getString() << "\n";
        return 1;
    }

    // Connect to service
    if(!err)
        err = bds.connect("test", "beam00");

    // Run a normal data gathering as a normal data access client would.
    if(!err)
        err = bdsTest(bds);

    if(err){
        cerr << "Error: " << err.getString() << "\n";
        return 1;
    }

    return 0;
}

/*****
 * BdsMetaDatal.cpp   BDS API example code for a Meta Data Client
 *                   T.Barnaby, BEAM Ltd, 2009-07-01
 *****/

*
* This is a very basic example of using the BdsApi from a meta data access
* perspective. It is designed to give an overview of using the API.
* This program gets information on the data channels.
*/
#include <iostream>
#include <iomanip>
#include <stdio.h>
#include <BdsD.h>
#include <BdsC.h>
using namespace Bds;
using namespace std;
// Function to read some data
BError bdsTest1(DataAccess& bds){
    BError          err;
    Selection        selection;
    Biter            i;

```

```

BUInt      n;
BList<Station> stations;

// Set up selection
selection.startTime.setString("2008-01-03T00:00:00.000000");
selection.endTime.setString("2008-01-03T00:01:00.000000");
selection.channels.append(SelectionChannel("BN", "EKA", "", ""));
// Get list of stations available
if(err = bds.stationGetList(selection, stations)){
    return err.set(1, BString("Error: Getting stations: ") + err.getString());
}
// This displays some of the information available
for(stations.start(i), n = 0; !stations.isEnd(i); stations.next(i), n++){
    Station& c = stations[i];

    cout << n << ": Station: " << c.name << " Type: " << c.type << "\n";
    cout << "      " << "Description: " << c.description
         << " Number of stations " << c.channels.number() << "\n";
}

return err;
}
int main(int argc, char** argv){
    BError      err;
    BString      hostName;
    DataAccess   bds;
    hostName = getenv("BDS_HOST");
    if(hostName == "")
        hostName = "localhost";

    if(argc == 2)
        hostName = argv[1];
    // Connect to the DataAccess service
    if(err = bds.connectService(BString("/") + hostName + "/bdsDataAccess")){
        cerr << "Error: " << err.getString() << "\n";
        return 1;
    }

    // Connect to service
    if(!err)
        err = bds.connect("test", "beam00");

    // Run a normal data gathering as a normal data access client would.
    if(!err)
        err = bdsTest1(bds);

    if(err){
        cerr << "Error: " << err.getString() << "\n";
        return 1;
    }

    return 0;
}

```

1.4 Python Examples

There are some examples of client applications using the BdsApi in the **bdsExamples** directory of the source code. Some simple Data Access client examples are listed below:

```

#!/usr/bin/python
import sys
import getopt
from bdslib import *
# Function to read some data
def bdsTest(bds):
    err = BError();
    selection = Selection();

    # Set up selection
    selection.startTime.setString("2008-01-01T00:00:00.000000");
    selection.endTime.setString("2008-01-01T00:01:00.000000");
    print("Selection: StartTime:", selection.startTime.getString(), "EndTime:",
          selection.endTime.getString());

    selection.channels.append(SelectionChannel("TT", "TSB01", "", ""));
    # bdsDumpSelection(selection);
    # Get list of all data available for the selection
    (err, dataInfo) = bds.dataSearch(selection);
    if(err):
        return err.set(1, BString("Error: Searching for data: ") + err.getString());
    # We should now choose which set of data we would like from the list, here we just

```

```

# choose all of the data.
s = dataInfo.channels.size();
if(s == 0):
    return err.set(1, "No data found");

# bdsDumpDataInfo(dataInfo);

# Open the data file for reading in IMS format
(err, dataHandle) = bds.dataOpen(dataInfo, "r", "IMS", 0);
if(err):
    return err;

# Read the formatted data
while(1):
    # print "Loop";
    (err, data) = bds.dataFormattedRead(dataHandle, 1024);
    if(err):
        return err;
    if(data.number() == 0):
        break;
    s = "".join(chr(x) for x in data);
    print(s);
    return err;

def main():
    hostName = "localhost";
    bds = DataAccess();
    # Connect to the DataAccess service
    err = bds.connectService("//" + hostName + "/bdsDataAccess");
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;
    # Connect to service
    err = bds.connect("test", "beam00");
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;

    (err, version, name) = bds.getVersion();
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;
    print("Version:" , version, "Name:", name);
    err = bdsTest(bds);
    if(err):
        print("Error:", err.getErrorNo(), err.getString());
        return 1;

    return 0;

if __name__ == "__main__":
    main();
#!/usr/bin/python
import sys
import getopt
from bdslib import *
# Function to read some data
def bdsTest(bds):
    err = BError();
    selection = Selection();
    dataInfo = DataInfo();

    # Set up selection
    selection.startTime.setString("2008-01-01T00:00:00.000000");
    selection.endTime.setString("2008-01-01T00:01:00.000000");
    selection.channels.append(SelectionChannel("TT", "TSB01", "", ""));
    # bdsDumpSelection(selection);
    # Get list of all data available for the selection
    (err, dataInfo) = bds.dataSearch(selection);
    if(err):
        return err.set(1, BString("Error: Searching for data: ") + err.getString());
    # We should now choose which set of data we would like from the list, here we just
    # choose the first entry and get the data in appropriate format.
    s = dataInfo.channels.size();
    if(s == 0):
        return err.set(1, "No data found");

    # bdsDumpDataInfo(dataInfo);

    (err, dataHandle) = bds.dataOpen(dataInfo, "r", "API", 0);
    if(err):
        return err;

    blockNumber = 0;
    while(1):
        # print "Loop";
        (err, data) = bds.dataGetBlock(dataHandle, 0, 1, blockNumber);
        if(err):
            return err;

```

```

        # print("DataChannels:", data.channelData.size());

        print("Data0:", data.channelData[0][0]);
        blockNumber += 1;
    return err;
def main():
    hostName = "localhost";
    bds = DataAccess();
    # Connect to the DataAccess service
    err = bds.connectService("//" + hostName + "/bdsDataAccess");
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;
    # Connect to service
    err = bds.connect("test", "beam00");
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;

    (err, version, name) = bds.getVersion();
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;
    print("Version:" , version, "Name:", name);
    err = bdsTest(bds);
    if(err):
        print("Error:", err.getErrorNo(), err.getString());
        return 1;

    return 0;
if __name__ == "__main__":
    main();
#!/usr/bin/python
import sys
import getopt
from bdslib import *
# Function to read display info on Station
def bdsTest(bds):
    err = BError();
    selection = Selection();

    # Set up selection
    selection.startTime.setString("2008-01-01T00:00:00.000000");
    selection.endTime.setString("2008-01-01T00:01:00.000000");
    selection.channels.append(SelectionChannel("TT", "TSA", "", ""));
# bdsDumpSelection(selection);
# Get list of stations available
(err, stations) = bds.stationGetList(selection);
if(err):
    return err.set(1, "Error: Getting stations: " + err.getString());
# This displays some of the information available
for s in stations:
    print("Station: " + s.name + " Type: " + s.type);
    print("      " + "Description: " + s.description + " Number of station/channels " +
          str(s.channels.number()));
    return err;
def main():
    hostName = "localhost";

    # Create DataAccess object to connect to BDS Server
    bds = DataAccess();
    # Connect to the DataAccess service
    err = bds.connectService("//" + hostName + "/bdsDataAccess");
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;
    # Connect to service
    err = bds.connect("test", "beam00");
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;

    (err, version, name) = bds.getVersion();
    if(err):
        print("Error: " + str(err) + "\n");
        return 1;
    print("Version:" , version, "Name:", name);
    err = bdsTest(bds);
    if(err):
        print("Error:", err.getErrorNo(), err.getString());
        return 1;

    return 0;
if __name__ == "__main__":
    main();

```


Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Bds	21
---------------------	-------	--------------------

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

BBuffer[external]	
BBufferStore[external]	
Bds::BdsDataPacket	99
Bds::ArrayChannel	93
Bds::BdsDataBlock	95
Bds::BdsDataBlockHeader	96
Bds::BdsDataBlockPos	97
Bds::BdsDataPacketHeader	101
Bds::BdsDataSegment	103
Bds::BdsDataStreamlet	105
Bds::BdsSeedType	107
Bds::DataFileCssData	254
Bds::CdChannel_1v0	116
Bds::CdDataChannel	118
Bds::CdDataFormatFrame_1v0	120
Bds::CdFlag	121
Bds::CdPacketData	122
Bds::ChannelInfo	136
Bds::ChannelInfos	139
Bds::ChannelName	144
Bds::CleanOptions	146
Bds::CompressSteim1	147
Bds::DataAvail	196
Bds::DataAvailChan	197
Bds::DataBlock	200
Bds::DataBlockChannel	202
Bds::DataBlockPos	204
Bds::DataCollate	212
Bds::DataError	213
Bds::DataFile	219
Bds::DataFileAd22	231
Bds::DataFileAscii	234
Bds::DataFileBdrs	237
Bds::DataFileBds	240
Bds::DataFileBknas	247

Bds::DataFileCd	249
Bds::DataFileCss	252
Bds::DataFileGcf	259
Bds::DataFileIdc	262
Bds::DataFileImms	265
Bds::DataFileLac	274
Bds::DataFileLog	276
Bds::DataFileResponse	283
Bds::DataFileSac	285
Bds::DataFileSeed	287
Bds::DataFileStationXml	293
Bds::DataFileTapeDigitiser	295
Bds::DataFileWra	297
Bds::DataFileWraAgso	301
Bds::DataFileOptions	281
Bds::DataFormat	303
Bds::DataFormatAll	306
Bds::DataHandle	307
Bds::DataInfo	309
Bds::Event	316
Bds::Fap	321
Bds::Fir	323
Bds::FirEntry	324
Bds::GcfChannel	325
Bds::LogSelect	341
Bds::Point	351
Bds::PoleZero	352
Bds::Polynomial	353
Bds::PolynomialEntry	356
Bds::Response	358
Bds::ResponseObj	366
Bds::Selection	367
Bds::SelectionChannel	372
Bds::SelectionInfo	373
Bds::Station	391
BObj[external]	
Bds::AccessGroup	51
Bds::Calibration	110
Bds::Change	125
Bds::ChangeGroup	128
Bds::Channel	132
Bds::ChannelInstrument	141
Bds::DataChannel	206
Bds::DataFileInfo	269
Bds::Digitiser	312
Bds::Group	327
Bds::ListRange	329
Bds::Location	332
Bds::Log	337
Bds::Network	342
Bds::Note	345
Bds::Sensor	376
Bds::Source	381
Bds::SourcePriority	384
Bds::SpecialChannel	387
Bds::TimePeriod	393
Bds::User	396
BSocket[external]	

BoapClientObject[external]	
Bds::AdminAccess	54
Bds::DataAccess	149
Bds::DataAddAccess	172
BoapClientObject[external]	

Chapter 4

Class Index

4.1 Class List

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

Bds::AccessGroup	This holds information on data access groups	51
Bds::AdminAccess	This is the AdminAccess Access API interface	54
Bds::ArrayChannel	This class defines an array's channel	93
Bds::BdsDataBlock	BdsDataFileBds: internal fixed size BDS Data Block	95
Bds::BdsDataBlockHeader	BdsDataFileBds: internal fixed size BDS Data Block header	96
Bds::BdsDataBlockPos	BdsDataFileBds: internal file storage data block position	97
Bds::BdsDataPacket	BdsDataFileBds: internal file storage packet	99
Bds::BdsDataPacketHeader	BdsDataFileBds internal file storage packet header	101
Bds::BdsDataSegment	BdsDataFileBds: internal file storage data segment	103
Bds::BdsDataStreamlet	BdsDataFileBds: internal file storage data streamlet	105
Bds::BdsSeedType	BdsDataFileSeed internal parent for all SEED types	107
Bds::Calibration	This class defines a calibration setting	110
Bds::CdChannel_1v0	BdsDataFile: Internal CD1.0 channel information	116
Bds::CdDataChannel	BdsDataFile: Internal CD channel information	118
Bds::CdDataFormatFrame_1v0	BdsDataFile: Internal CD1.0 frame information	120
Bds::CdFlag	BdsDataFile: Internal CD flag	121
Bds::CdPacketData	BdsDataFile: Internal CD data packet	122
Bds::Change	This holds information on a Medatdata or Sensor data database change	125

Bds::ChangeGroup	
This holds information on a set of Changes	128
Bds::Channel	
This class defines a seismic data Channel	132
Bds::ChannelInfo	
This class provides full Metadata information on a channel	136
Bds::ChannelInfos	
This class provides Metadata information on a set of channels	139
Bds::ChannelInstrument	
This class defines a Channel 's instrument	141
Bds::ChannelName	
This class defines a full channel name	144
Bds::CleanOptions	
This defines the set of clean options used in the clean() function	146
Bds::CompressSteim1	
Steim1 un-compress class	147
Bds::DataAccess	
This is the Data Access API interface to the BDS system	149
Bds::DataAddAccess	
This is the DataAdd Access API interface	172
Bds::DataAvail	
This class provides availability information on a particular period of data	196
Bds::DataAvailChan	
This class defines availability information on a set of data	197
Bds::DataBlock	
This class provides the actual Sensor data values contained within a single data block	200
Bds::DataBlockChannel	
This class provides the actual seismic data values contained within a single data block along with the network:station:channel:source information	202
Bds::DataBlockPos	
This defines the position of a data block in a file. It is used by the BDS data converters to order blocks by time	204
Bds::DataChannel	
This class defines information on a single channel's set of data stored in a file	206
Bds::DataCollate	
Not sure if this is used or what it does	212
Bds::DataError	
This stores a data error. It includes an error number and a string as well as information on what seismic channel it is for	213
Bds::DataFile	
This class defines the interface for generic data file access that all of the BDS data converters share	219
Bds::DataFileAd22	
Data file convertor for AD22 format files	231
Bds::DataFileAscii	
Data file convertor for ASCII format files	234
Bds::DataFileBdrs	
Data file convertor for BDRS format files	237
Bds::DataFileBds	
This class implements the BDS Data File/Stream access system	240
Bds::DataFileBknas	
Data file convertor for BKNAS format files	247
Bds::DataFileCd	
Data file convertor for CD1.0 and CD1.1 file formats	249
Bds::DataFileCss	
Data file convertor for CSS format files	252
Bds::DataFileCssData	
DataFileCss internal CSS data type	254

Bds::DataFileGcf	
Data file convertor for GCF format files	259
Bds::DataFileIdc	
This class defines the interface for IDC response data file access	262
Bds::DataFileIms	
Data file convertor for IMS format files	265
Bds::DataFileInfo	
This class defines information on a Sensor data file	269
Bds::DataFileLac	
Data file convertor for LAC format files	274
Bds::DataFileLog	
Data file convertor for LOG format files	276
Bds::DataFileOptions	
This defines a list of BDS data converter options	281
Bds::DataFileResponse	
This class defines the interface for generic response data file access	283
Bds::DataFileSac	
Data file convertor for SAC format files	285
Bds::DataFileSeed	
Data file convertor for SEED file formats	287
Bds::DataFileStationXml	
This class defines the interface for generic response data file access	293
Bds::DataFileTapeDigitiser	
This class implements the TapeDigitiser's file output conversion and storing system	295
Bds::DataFileWra	
Data file convertor for WRA format files	297
Bds::DataFileWraAgso	
Data file convertor for WRA AGSO format files	301
Bds::DataFormat	
This holds information on a Sensor data format	303
Bds::DataFormatAll	
This class defines the interface for generic data file access	306
Bds::DataHandle	
This defines a handle to a sensor data stream/file when opened for read or write	307
Bds::DataInfo	
This class defines information on a set of data	309
Bds::Digitiser	
This class defines a seismic Digitiser	312
Bds::Event	
This class defines a seismic event	316
Bds::Fap	
This class defines an entry in an Amplitude/Phase Response table	321
Bds::Fir	
This class defines an FIR response table	323
Bds::FirEntry	
This class defines an entry in a FIR coefficient table	324
Bds::GcfChannel	
DataFileGcf internal GCF channel information	325
Bds::Group	
This holds information on a User security group	327
Bds::ListRange	
This class defines an integer based range	329
Bds::Location	
This class defines the physical location of a Station	332
Bds::Log	
This holds information on a Log entry	337
Bds::LogSelect	
This defines the selection criteria when requesting a set of log entries	341

Bds::Network	This class defines a seismic Network organisation	342
Bds::Note	This holds information on a Note for general information	345
Bds::Point	This class defines an X,Y location	351
Bds::PoleZero	This class defines a Pole/Zero Response	352
Bds::Polynomial	This class defines an Polynomial response table	353
Bds::PolynomialEntry	This class defines an entry in a Polynomial coefficient table	356
Bds::Response	This class defines a seismic Response characteristic	358
Bds::ResponseObj	Response object adding string conversion	366
Bds::Selection	This class defines a generic Metadata or Sensor data selection	367
Bds::SelectionChannel	This class defines an individual channel for selection	372
Bds::SelectionInfo	This class defines the set of Metadata or Siesmic sensor data to be selected when getSelection↔ Info() is use	373
Bds::Sensor	This class defines a seismic Sensor	376
Bds::Source	This class defines a seismic data Source	381
Bds::SourcePriority	This class defines a Source Priority entry	384
Bds::SpecialChannel	A Special channel identifier	387
Bds::Station	This class defines a seismic station	391
Bds::TimePeriod	This class defines a TimePeriod	393
Bds::User	This holds information on a user	396

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

/src/blacknest/bds/bds/bdsDataLib/BdsCompress.cpp	401
/src/blacknest/bds/bds/bdsDataLib/BdsCompress.d	401
/src/blacknest/bds/bds/bdsDataLib/BdsCompress.h	401
/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.cpp	402
/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.d	403
/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.h	403
/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.cpp	403
/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.d	404
/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h	404
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.cpp	405
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.d	406
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.h	406
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.cpp	407
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.d	408
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.h	408
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.cpp	409
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.d	409
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.h	409
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.cpp	410
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.d	412
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h	412
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.cpp	415
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.d	416
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.h	416
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.cpp	417
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.d	419
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h	419
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.cpp	421
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.d	421
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.h	421
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.cpp	423
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.d	423
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.h	423
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.cpp	424
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.d	425

/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.h	425
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIms.cpp	426
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIms.d	427
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIms.h	427
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.cpp	428
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.d	428
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.h	428
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.cpp	429
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.d	430
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.h	430
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.cpp	431
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.d	431
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.h	431
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.cpp	432
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.d	433
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.h	433
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.cpp	434
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.d	435
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.h	435
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.cpp	436
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.d	437
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.h	437
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.cpp	438
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.d	438
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.h	438
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.cpp	439
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.d	440
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.h	440
/src/blacknest/bds/bds/bdsDataLib/BdsDataLib.cpp	441
/src/blacknest/bds/bds/bdsDataLib/BdsDataLib.d	441
/src/blacknest/bds/bds/bdsDataLib/BdsDataLib.h	441
/src/blacknest/bds/bds/bdsDataLib/canada_compress.d	448
/src/blacknest/bds/bds/bdsDataLib/canada_compress.h	448
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.cpp	442
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.d	444
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.h	444
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.cpp	447
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.d	447
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.h	447
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.cpp	448
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.d	448
/src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.idl	448
BdsC.cc	450
BdsC.d	451
BdsC.h	451
BdsD.cc	455
BdsD.d	455
BdsD.h	
BOAP data class definitions for: Bds	455
BdsLib.cpp	469
BdsLib.d	471
BdsLib.h	
General BdsLib API functions	471
BdsS.cc	475
BdsS.d	476
BdsT.cc	476

Chapter 6

Namespace Documentation

6.1 Bds Namespace Reference

Classes

- class [AccessGroup](#)
This holds information on data access groups.
- class [AdminAccess](#)
This is the [AdminAccess](#) Access API interface.
- class [ArrayChannel](#)
This class defines an array's channel.
- struct [BdsDataBlock](#)
BdsDataFileBds: internal fixed size BDS Data Block.
- struct [BdsDataBlockHeader](#)
BdsDataFileBds: internal fixed size BDS Data Block header.
- class [BdsDataBlockPos](#)
BdsDataFileBds: internal file storage data block position.
- class [BdsDataPacket](#)
BdsDataFileBds: internal file storage packet.
- struct [BdsDataPacketHeader](#)
BdsDataFileBds internal file storage packet header.
- class [BdsDataSegment](#)
BdsDataFileBds: internal file storage data segment.
- class [BdsDataStreamlet](#)
BdsDataFileBds: internal file storage data streamlet.
- class [BdsSeedType](#)
BdsDataFileSeed internal parent for all SEED types.
- class [Calibration](#)
This class defines a calibration setting.
- struct [CdChannel_1v0](#)
BdsDataFile: Internal CD1.0 channel information.
- class [CdDataChannel](#)
BdsDataFile: Internal CD channel information.
- struct [CdDataFormatFrame_1v0](#)
BdsDataFile: Internal CD1.0 frame information.
- class [CdFlag](#)

- BdsDataFile: Internal CD flag.*

 - class [CdPacketData](#)

BdsDataFile: Internal CD data packet.
 - class [Change](#)

This holds information on a Metadata or [Sensor](#) data database change.
 - class [ChangeGroup](#)

This holds information on a set of Changes.
 - class [Channel](#)

This class defines a seismic data [Channel](#).
 - class [ChannelInfo](#)

This class provides full Metadata information on a channel.
 - class [ChannelInfos](#)

This class provides Metadata information on a set of channels.
 - class [ChannelInstrument](#)

This class defines a [Channel](#)'s instrument.
 - class [ChannelName](#)

This class defines a full channel name.
 - class [CleanOptions](#)

This defines the set of clean options used in the `clean()` function.
 - class [CompressSteim1](#)

Steim1 un-compress class.
 - class [DataAccess](#)

This is the Data Access API interface to the BDS system.
 - class [DataAddAccess](#)

This is the DataAdd Access API interface.
 - class [DataAvail](#)

This class provides availability information on a particular period of data.
 - class [DataAvailChan](#)

This class defines availability information on a set of data.
 - class [DataBlock](#)

This class provides the actual [Sensor](#) data values contained within a single data block.
 - class [DataBlockChannel](#)

This class provides the actual seismic data values contained within a single data block along with the network↔:station:channel:source information.
 - class [DataBlockPos](#)

This defines the position of a data block in a file. It is used by the BDS data converters to order blocks by time.
 - class [DataChannel](#)

This class defines information on a single channel's set of data stored in a file.
 - class [DataCollate](#)

Not sure if this is used or what it does.
 - class [DataError](#)

This stores a data error. It includes an error number and a string as well as information on what seismic channel it is for.
 - class [DataFile](#)

This class defines the interface for generic data file access that all of the BDS data converters share.
 - class [DataFileAd22](#)

Data file convertor for AD22 format files.
 - class [DataFileAscii](#)

Data file convertor for ASCII format files.
 - class [DataFileBdrs](#)

Data file convertor for BDRS format files.

- class [DataFileBds](#)
This class implements the BDS Data File/Stream access system.
- class [DataFileBknas](#)
Data file convertor for BKNAS format files.
- class [DataFileCd](#)
Data file convertor for CD1.0 and CD1.1 file formats.
- class [DataFileCss](#)
Data file convertor for CSS format files.
- class [DataFileCssData](#)
[DataFileCss](#) internal CSS data type.
- class [DataFileGcf](#)
Data file convertor for GCF format files.
- class [DataFileIdc](#)
This class defines the interface for IDC response data file access.
- class [DataFileImms](#)
Data file convertor for IMS format files.
- class [DataFileInfo](#)
This class defines information on a [Sensor](#) data file.
- class [DataFileLac](#)
Data file convertor for LAC format files.
- class [DataFileLog](#)
Data file convertor for LOG format files.
- class [DataFileOptions](#)
This defines a list of BDS data converter options.
- class [DataFileResponse](#)
This class defines the interface for generic response data file access.
- class [DataFileSac](#)
Data file convertor for SAC format files.
- class [DataFileSeed](#)
Data file convertor for SEED file formats.
- class [DataFileStationXml](#)
This class defines the interface for generic response data file access.
- class [DataFileTapeDigitiser](#)
This class implements the TapeDigitiser's file output conversion and storing system.
- class [DataFileWra](#)
Data file convertor for WRA format files.
- class [DataFileWraAgso](#)
Data file convertor for WRA AGSO format files.
- class [DataFormat](#)
This holds information on a [Sensor](#) data format.
- class [DataFormatAll](#)
This class defines the interface for generic data file access.
- class [DataHandle](#)
This defines a handle to a sensor data stream/file when opened for read or write.
- class [DataInfo](#)
This class defines information on a set of data.
- class [Digitiser](#)
This class defines a seismic [Digitiser](#).
- class [Event](#)
This class defines a seismic event.
- class [Fap](#)

- This class defines an entry in an Amplitude/Phase [Response](#) table.*

 - class [Fir](#)

This class defines an FIR response table.
 - class [FirEntry](#)

This class defines an entry in a FIR coefficient table.
 - struct [GcfChannel](#)

[DataFileGcf](#) internal GCF channel information.
 - class [Group](#)

This holds information on a [User](#) security group.
 - class [ListRange](#)

This class defines an integer based range.
 - class [Location](#)

This class defines the physical location of a [Station](#).
 - class [Log](#)

This holds information on a [Log](#) entry.
 - class [LogSelect](#)

This defines the selection criteria when requesting a set of log entries.
 - class [Network](#)

This class defines a seismic [Network](#) organisation.
 - class [Note](#)

This holds information on a [Note](#) for general information.
 - class [Point](#)

This class defines an X,Y location.
 - class [PoleZero](#)

This class defines a Pole/Zero [Response](#).
 - class [Polynomial](#)

This class defines an [Polynomial](#) response table.
 - class [PolynomialEntry](#)

This class defines an entry in a [Polynomial](#) coefficient table.
 - class [Response](#)

This class defines a seismic [Response](#) characteristic.
 - class [ResponseObj](#)

[Response](#) object adding string conversion.
 - class [Selection](#)

This class defines a generic Metadata or [Sensor](#) data selection.
 - class [SelectionChannel](#)

This class defines an individual channel for selection.
 - class [SelectionInfo](#)

This class defines the set of Metadata or Seismic sensor data to be selected when `getSelectionInfo()` is use.
 - class [Sensor](#)

This class defines a seismic [Sensor](#).
 - class [Source](#)

This class defines a seismic data [Source](#).
 - class [SourcePriority](#)

This class defines a [Source](#) Priority entry.
 - class [SpecialChannel](#)

A Special channel identifier.
 - class [Station](#)

This class defines a seismic station.
 - class [TimePeriod](#)

This class defines a [TimePeriod](#).
 - class [User](#)

This holds information on a user.

Typedefs

- typedef **BList**< [DataFormat](#) > [DataFormats](#)

A list of the available [Sensor](#) data formats.

Enumerations

- enum [Errors](#) {
[ErrorNoMetaData](#) = 64 , [ErrorDataQuality](#) = 65 , [ErrorSlaveMode](#) = 66 , [ErrorTimeStamp](#) = 67 ,
[ErrorValidate](#) = 80 , [ErrorValidateMissingBlocks](#) = 81 , [ErrorValidateTimeBackwards](#) = 82 , [ErrorValidateFilenameTime](#) = 83 ,
[ErrorValidateMetaData](#) = 84 , [ErrorValidateFix](#) = 85 , [ErrorValidateDuplicate](#) = 86 , [ErrorValidateReorder](#) = 87
, [ErrorValidateBdsFudge](#) = 88 }
The System Error number list in addition to standard system error numbers.
- enum [Priority](#) { [PriorityLow](#) , [PriorityNormal](#) , [PriorityHigh](#) }
Priority levels.
- enum [Mode](#) { [ModeMaster](#) , [ModeSlave](#) }
BdsServer mode.
- enum [DataFlags](#) {
[DataFlagNone](#) = 0x00 , [DataFlagClipDataToTime](#) = 0x01 , [DataFlagClipDataToChannels](#) = 0x02 ,
[DataFlagMergeSegments](#) = 0x04 ,
[DataFlagNoMetadata](#) = 0x08 }
Flags when opening data files.
- enum [SelectionGroup](#) { [SelectionGroupData](#) , [SelectionGroupMetaData](#) , [SelectionGroupDataWithCount](#) }
The Selection group when making selections.
- enum [SampleFormat](#) {
[SampleFormatUnknown](#) , [SampleFormatInt16](#) , [SampleFormatInt32](#) , [SampleFormatFloat32](#) ,
[SampleFormatFloat64](#) , [SampleFormatInt24](#) }
The actual format of a data sample.
- enum [AvailType](#) { [AvailNone](#) , [AvailPartial](#) , [AvailFull](#) }
A flag defining the data availability state.
- enum [DataFormatSet](#) {
[DataFormatSetNone](#) = 0x00 , [DataFormatSetMetadataRead](#) = 0x01 , [DataFormatSetMetadataWrite](#) = 0x02 ,
[DataFormatSetSensordataRead](#) = 0x04 ,
[DataFormatSetSensordataWrite](#) = 0x08 }
Data format abilities bitset.
- enum [LocationSelect](#) { [LocationSelectAll](#) , [LocationSelectStation](#) , [LocationSelectChannel](#) }
Which Locations to select.
- enum [BdsDataType](#) { [BdsDataTypeBlock](#) = 0x42534442 , [BdsDataTypeInfo](#) = 0x30534442 , [BdsDataTypeData](#) = 0x31534442 , [BdsDataTypeInfoExtra](#) = 0x32534442 }
BdsDataFileBds: internal file block type field.
- enum [FileHeaderType](#) { [FileHeaderType_Standard](#) = 1 , [FileHeaderType_TapeDigitiser](#) = 10 }
- enum [FileSampleType](#) {
[FileSampleType_Unknown](#) , [FileSampleType_Float32](#) , [FileSampleType_Float64](#) , [FileSampleType_Int16](#) ,
[FileSampleType_Int32](#) }

Functions

- **BError** [bdsLibInit](#) ([DataAccess](#) &bds)
Initialise the bdsLib with settings from the BdsServer.
- **BError** [bdsLibInit](#) ([DataAddAccess](#) &bds)
Initialise the bdsLib with settings from the BdsServer.
- **BError** [bdsLibInit](#) ([AdminAccess](#) &bds)
Initialise the bdsLib with settings from the BdsServer.
- void [bdsDumpPoleZeros](#) ([PoleZero](#) poleZeros)
Debug print out a PoleZeros object.
- void [bdsChannelGetTypeAux](#) ([BString](#) name, [BString](#) &type, [BString](#) &aux)
Get the channel type and aux fields from a generic channel name.
- [BString](#) [bdsChannelGetName](#) ([BString](#) type, [BString](#) aux)
Create a full channel name from a channels type and aux fields.
- **BError** [bdsDataInfoSetTimeRange](#) ([DataInfo](#) &dataInfo)
Restricts the time tange of all of the [DataInfo](#)'s channels to match the [DataInfo](#)'s startTime/endTime fields.
- **BError** [bdsDataInfoFromInfo](#) ([BDictString](#) info, [DataInfo](#) &dataInfo, [Bool](#) append)
Convert info to [DataInfo](#).
- **BError** [bdsInfoFromDataInfo](#) (const [DataInfo](#) &dataInfo, [BDictString](#) &info)
Converts a [DataInfo](#) object into a [BDictString](#) list of named strings.
- **BError** [bdsDataInfoFlatten](#) ([DataInfo](#) &dataInfo)
Flattens a [DataInfo](#) to 1 segment per channel for use in dataOpen() calls.
- **BError** [bdsDataInfoMergeFlatten](#) ([DataInfo](#) &dataInfo, const [DataInfo](#) &dataInfoAdd)
Merges a [DataInfo](#) into another flattening the segments to 1 for use in dataOpen() calls.
- [BString](#) [bdsUnitsConvert](#) ([BString](#) units)
Tidy up units name to standard SI units format.
- **BError** [bdsDataInfoFromChannelInfos](#) (const [ChannelInfos](#) &channelInfos, [DataInfo](#) &dataInfo)
Sets up a [DataInfo](#) object from [ChannelInfos](#).
- static int [responseSort](#) ([Response](#) &a, [Response](#) &b)
- **BError** [bdsMetadataImportFix](#) ([ChannelInfos](#) &channelInfos, [Bool](#) stageRenummer)
Fix up [ChannelInfos](#) from import. Mainly making sure response stages and their units are correct.
- **BError** [bdsMetadataExportFix](#) ([ChannelInfos](#) &channelInfos, [Bool](#) singleResponse, [Bool](#) stageRenummer, [Bool](#) changeUnits, [Bool](#) stageGains, [Bool](#) decimation, [Bool](#) toDisplacement, [Bool](#) toNm)
Fix up [ChannelInfos](#) for export. Mainly making sure response stages and their units are correct.
- [BString](#) [bdsStationAlias](#) ([Station](#) station)
Returns the station alias if set else its name.
- void [bdsDumpSelection](#) ([Selection](#) sel)
Debug print out a [Selection](#) object.
- void [bdsDumpSelectionInfo](#) ([SelectionInfo](#) sel)
Debug print out a [SelectionInfo](#) object.
- void [bdsDumpDataInfo](#) ([DataInfo](#) dataInfo, int includeInfo=0)
Debug print out a [DataInfo](#) object.
- void [bdsDumpChannelInfos](#) (const [ChannelInfos](#) &channelInfos)
Debug print out a [ChannelInfos](#) object.
- void [bdsDumpData](#) (const [DataBlock](#) &dataBlock, int nSamples=0)
Debug print out a [DataBlock](#) object.
- void [bdsDumpLocation](#) ([Location](#) location)
Debug printout location.
- [BString](#) [bdsDataChannelInfo](#) (const [DataChannel](#) &dataChannel)
Returns a string representation of a [DataChannel](#) object.
- [BString](#) [bdsDataChannelRef](#) (const [DataChannel](#) &dataChannel)

- Returns the string reference name of a [DataChannel](#) object.*
- **BString** [bdsDataChannelRef](#) (const [ChannelInfo](#) &channelInfo)
 - Returns the string reference name of a [ChannelInfo](#) object.*
- **BError** [bdsDataChannelOverallResponse](#) (const [ChannelInfo](#) &channelInfo, [Response](#) &response)
 - Returns the overall response from the list of responses in a [ChannelInfo](#).*
- **BString** [bdsSelectionChannelInfo](#) (const [Selection](#) &selection, **BUInt** channel)
 - Returns a string describing the name and time period of a selection channel.*
- double [bdsPoleZeroGain](#) (const [PoleZero](#) &poleZero, double frequency)
 - Calculates the overall gain of the given [PoleZero](#) transfer function.*
- void [bdsPoleZeroGainPhase](#) (const [PoleZero](#) &poleZero, double frequency, double &gain, double &phase)
 - Calculates the overall gain and phase of the given [PoleZero](#) transfer function.*
- void [bdsPoleZeroToFap](#) (const [PoleZero](#) &poleZero, **BUInt** nPoints, double calibrationFrequency, double sampleFrequency, **BArray**< [Fap](#) > &fap)
 - Convert [PoleZero](#) to FAP.*
- static **BString** [fileNameTime](#) (**BTimeStamp** t)
- **BString** [bdsFileNameExpand](#) (**BString** fileName, [ChannelInfo](#) &channelInfo)
 - Default filename from a [ChannelInfo](#).*
- **BString** [bdsFileNameExpand](#) (**BString** fileName, [ChannelInfos](#) &channelInfos)
 - Default filename from a list of [ChannelInfo](#)'s.*
- **BString** [bdsFileNameExpand](#) (**BString** fileName, [Selection](#) &sel)
 - Default filename from a [Selection](#).*
- void [bdsSpecialChannelsSet](#) (const **BList**< [SpecialChannel](#) > specialChannels)
 - Set the special channels list.*
- **BList**< [SpecialChannel](#) > [bdsSpecialChannels](#) ()
 - Return list of special channels.*
- **Bool** [bdsSpecialChannelIgnore](#) (**BString** network, **BString** station, **BString** channel)
 - Check if channel should be ignored.*
- char [seedChannelInstrumentCode](#) (**BString** dataType)
 - Returns SEED instrument code from dataType.*
- **BString** [seedChannelDataType](#) (**BString** channel)
 - Returns dataType from channel name based on SEED channel name convention.*
- **BStringList** [bdsDataTypes](#) ()
 - Returns all known data types.*
- **BStringList** [bdsUnits](#) ()
 - Returns all known SI units.*
- **BString** [bdsUnitCase](#) (**BString** unit)
 - Converts character case of units.*
- **BError** [bdsUnCompressCm8](#) (**BUInt8** *buffer, **BUInt** n, **BArray**< **BInt32** > & data)
 - Uncompress CM8 formatted data.*
- **BError** [bdsUnCompressSteim1](#) (**BUInt8** *buffer, **BUInt** n, **BArray**< **BInt32** > & data)
 - Uncompress STEIM1 formatted data.*
- static **BString** [nullString](#) (**BString** s)
- **BUInt32** [crc](#) (**BUInt32** crc, void * data, int numBytes)
- static void [crclnit](#) ()
- static uint64_t [crc64](#) (const void *buffer, const uint32_t len)
- **BString** [getHexString](#) (char * data, int len)
- int [duplicateDump](#) ([DataBlock](#) &data1, [DataBlock](#) &data2, int channel)
- static **BError** [fixedString](#) (double v, int fieldWidth, int numDecimal, **BString** &str)
- void [dataCalculateDifference](#) (**BInt32** &prevValue, **BArray**< **BInt32** > & data)
- void [dataCalculateUnDifference](#) (**BInt32** &prevValue, **BArray**< **BInt32** > & data)
- **BInt32** [dataChecksum](#) (**BInt32** checksum, **BArray**< **BInt32** > & data)
- **BError** [dataCompressCm6](#) (int &prevValue1, int &prevValue2, **BArray**< **BInt32** > & data, **BString** &d)

- **BError** [dataDeCompressCm6](#) (int &prevValue1, int &prevValue2, **BString** &d, **BArray**< **BInt32** > &data)
- static void [dataConvert](#) (const **BArray**< **BFloat64** > &dataIn, **BArray**< **BInt32** > &dataOut)
- static **BString** [unitsCode](#) ([Response](#) &r)
- static **BString** [stringFormat](#) (**BTimeStamp** t)
- static **BString** [removeCR](#) (**BString** str)
- **BString** [fixedWidthValue](#) (double v, int width)
This returns a double as a fixed width string truncating the data.
- static double [roundDigits](#) (double v, int nDigits)
- static void [bdsDataFileSeedLogWarning](#) (char *str)
- static void [bdsDataFileSeedLogError](#) (char *str)
- static hptime_t [seedTime](#) (**BTimeStamp** t)
- static **BString** [seedTimeString](#) (**BTimeStamp** t)
- static **BTimeStamp** [fromSeedTimeString](#) (**BString** str)
- static void [dataConvert](#) (const **BArray**< **BFloat64** > &dataIn, **BArray**< **BInt32** > &dataOut)
- static void [dataConvert](#) (const **BArray**< **BFloat64** > &dataIn, **BArray**< **BFloat32** > &dataOut)
- static void [record_handler](#) (char *record, int reclen, void *info)

Variables

- const **BUInt32** [apiVersion](#) = 0
- static **BList**< [SpecialChannel](#) > [bdsSpecialChannelsList](#)
- SeedIDCodeToDataType [seedIDCodeToDataTypes](#) []
- const int [NetworkNameLen](#) = 3
Maximum [Network](#) name length.
- const int [StationNameLen](#) = 5
Maximum [Station](#) name length.
- const int [ChannelTypeLen](#) = 3
Maximum [Channel](#) type name length.
- const int [ChannelAuxLen](#) = 2
Maximum [Channel](#) Aux length.
- const int [SourceLen](#) = 16
Maximum [Source](#) length.
- const **BString** [BdsDataFileVersion](#) = "1.2.0"
- static uint64_t [crcVec](#) [256]
- static int [crclntDone](#)
- static char [cm6Table](#) [64]
- static **BUInt8** [cm6TableRev](#) [128]
- const char * [node_types](#) []
- const double [Scale](#) = 16777216.0
- [DataFormatAll](#) [dataFormatAll](#)

6.1.1 Typedef Documentation

6.1.1.1 DataFormats

```
typedef BList<DataFormat > Bds::DataFormats
```

A list of the available [Sensor](#) data formats.

6.1.2 Enumeration Type Documentation

6.1.2.1 Errors

enum `Bds::Errors`

The System Error number list in addition to standard system error numbers.

Enumerator

<code>ErrorNoMetaData</code>	No Metadata is available
<code>ErrorDataQuality</code>	Data quality error
<code>ErrorSlaveMode</code>	BdsServer is in slave mode
<code>ErrorTimeStamp</code>	Timestamp invalid
<code>ErrorValidate</code>	A validation error occurred
<code>ErrorValidateMissingBlocks</code>	Validation found missing blocks
<code>ErrorValidateTimeBackwards</code>	Validation found the time went backwards between blocks
<code>ErrorValidateFilenameTime</code>	Validation of the file name failed
<code>ErrorValidateMetaData</code>	There was no Metadata available
<code>ErrorValidateFix</code>	Validation has fixed some issues
<code>ErrorValidateDuplicate</code>	Validation has found duplicate blocks
<code>ErrorValidateReorder</code>	Validation has re-ordered blocks
<code>ErrorValidateBdsFudge</code>	Special BDS SensorData/Metadata changes have been applied

6.1.2.2 Priority

enum `Bds::Priority`

Priority levels.

Enumerator

<code>PriorityLow</code>	The lowest priority level
<code>PriorityNormal</code>	The normal priority level
<code>PriorityHigh</code>	The highest priority level

6.1.2.3 Mode

enum `Bds::Mode`

BdsServer mode.

Enumerator

ModeMaster	BdsServer is a master
ModeSlave	BdsServer is a slave

6.1.2.4 DataFlags

```
enum Bds::DataFlags
```

Flags when opening data files.

Enumerator

DataFlagNone	No data flags
DataFlagClipDataToTime	Clip the data to the time period requested so that data begins and ends with the sample at the requested time. Normally the BDS will return data beginning at the startTime of the data block in which the user startTime occurred and the endTime of the block that the user supplied endTime occurs so that complete original data blocks are returned.
DataFlagClipDataToChannels	When requesting data from a number of channels the start and end times per channel may be different due to missing blocks or other reasons. This option asks the BDS to truncate the data so that all channels start and end with the sample timed sample.
DataFlagMergeSegments	Data will normally be segmented at file boundaries. This option merges these segments assuming the start/end times match.
DataFlagNoMetadata	Don't include Metadata in export data files.

6.1.2.5 SelectionGroup

```
enum Bds::SelectionGroup
```

The [Selection](#) group when making selections.

Enumerator

SelectionGroupData	Select items from Sensor data
SelectionGroupMetaData	Select items from Metadata
SelectionGroupDataWithCount	Select items from Sensor data and return the number of items found

6.1.2.6 SampleFormat

```
enum Bds::SampleFormat
```

The actual format of a data sample.

Enumerator

SampleFormatUnknown	Unknown sample format
SampleFormatInt16	16 bit signed integer format
SampleFormatInt32	32 bit signed integer format
SampleFormatFloat32	IEEE 32 bit floating point format
SampleFormatFloat64	IEEE 64 bit floating point format
SampleFormatInt24	24 bit signed integer format

6.1.2.7 AvailType

```
enum Bds::AvailType
```

A flag defining the data availability state.

Enumerator

AvailNone	There is no data available There is full data available
AvailPartial	There is partial data available
AvailFull	There is no data available

6.1.2.8 DataFormatSet

```
enum Bds::DataFormatSet
```

Data format abilities bitset.

Enumerator

DataFormatSetNone	
DataFormatSetMetadataRead	
DataFormatSetMetadataWrite	
DataFormatSetSensordataRead	
DataFormatSetSensordataWrite	

6.1.2.9 LocationSelect

```
enum Bds::LocationSelect
```

Which Locations to select.

Enumerator

LocationSelectAll	
LocationSelectStation	
LocationSelectChannel	

6.1.2.10 BdsDataType

```
enum Bds::BdsDataType
```

BdsDataFileBds: internal file block type field.

Enumerator

BdsDataTypeBlock	
BdsDataTypeInfo	
BdsDataTypeData	
BdsDataTypeInfoExtra	

6.1.2.11 FileHeaderType

```
enum Bds::FileHeaderType
```

Enumerator

FileHeaderType_Standard	
FileHeaderType_TapeDigitiser	

6.1.2.12 FileSampleType

```
enum Bds::FileSampleType
```

Enumerator

FileSampleType_Unknown	
FileSampleType_Float32	
FileSampleType_Float64	
FileSampleType_Int16	
FileSampleType_Int32	

6.1.3 Function Documentation

6.1.3.1 bdsLibInit() [1/3]

```
BError Bds::bdsLibInit (
    DataAccess & bds )
```

Initialise the bdsLib with settings from the BdsServer.

6.1.3.2 bdsLibInit() [2/3]

```
BError Bds::bdsLibInit (
    DataAddAccess & bds )
```

Initialise the bdsLib with settings from the BdsServer.

6.1.3.3 bdsLibInit() [3/3]

```
BError Bds::bdsLibInit (
    AdminAccess & bds )
```

Initialise the bdsLib with settings from the BdsServer.

6.1.3.4 bdsDumpPoleZeros()

```
void Bds::bdsDumpPoleZeros (
    PoleZero poleZeros )
```

Debug print out a PoleZeros object.

6.1.3.5 bdsChannelGetTypeAux()

```
void Bds::bdsChannelGetTypeAux (
    BString name,
    BString & type,
    BString & aux )
```

Get the channel type and aux fields from a generic channel name.

Parameters

in	<i>name</i>	The channels full name
out	<i>type</i>	Returns the type component of the channel's name
out	<i>aux</i>	Returns the aux component of the channel's name

6.1.3.6 bdsChannelGetName()

```
BString Bds::bdsChannelGetName (
    BString type,
    BString aux )
```

Create a full channel name from a channels type and aux fields.

6.1.3.7 bdsDataInfoSetTimeRange()

```
BError Bds::bdsDataInfoSetTimeRange (
    DataInfo & dataInfo )
```

Restricts the time tange of all of the [DataInfo](#)'s channels to match the [DataInfo](#)'s startTime/endTime fields.

6.1.3.8 bdsDataInfoFromInfo()

```
BError Bds::bdsDataInfoFromInfo (
    BDictString info,
    DataInfo & dataInfo,
    Bool append )
```

Convert info to [DataInfo](#).

Sets up a [DataInfo](#) object from a [BDictString](#) list of named strings.

6.1.3.9 bdsInfoFromDataInfo()

```
BError Bds::bdsInfoFromDataInfo (
    const DataInfo & dataInfo,
    BDictString & info )
```

Converts a [DataInfo](#) object into a [BDictString](#) list of named strings.

6.1.3.10 bdsDataInfoFlatten()

```
BError Bds::bdsDataInfoFlatten (
    DataInfo & dataInfo )
```

Flattens a [DataInfo](#) to 1 segment per channel for use in dataOpen() calls.

6.1.3.11 bdsDataInfoMergeFlatten()

```
BError Bds::bdsDataInfoMergeFlatten (
    DataInfo & dataInfo,
    const DataInfo & dataInfoAdd )
```

Merges a [DataInfo](#) into another flattening the segments to 1 for use in dataOpen() calls.

6.1.3.12 bdsUnitsConvert()

```
BString Bds::bdsUnitsConvert (
    BString units )
```

Tidy up units name to standard SI units format.

6.1.3.13 bdsDataInfoFromChannelInfos()

```
BError Bds::bdsDataInfoFromChannelInfos (
    const ChannelInfos & channelInfos,
    DataInfo & dataInfo )
```

Sets up a Datainfo object from [ChannelInfos](#).

6.1.3.14 responseSort()

```
static int Bds::responseSort (
    Response & a,
    Response & b ) [static]
```

6.1.3.15 bdsMetadataImportFix()

```
BError Bds::bdsMetadataImportFix (
    ChannelInfos & channelInfos,
    Bool stageRenumber )
```

Fix up [ChannelInfos](#) from import. Mainly making sure response stages and their units are correct.

6.1.3.16 bdsMetadataExportFix()

```
BError Bds::bdsMetadataExportFix (
    ChannelInfos & channelInfos,
    Bool singleResponse,
    Bool stageRenumber,
    Bool changeUnits,
    Bool stageGains,
    Bool decimation,
    Bool toDisplacement,
    Bool toNm )
```

Fix up [ChannelInfos](#) for export. Mainly making sure response stages and their units are correct.

6.1.3.17 bdsStationAlias()

```
BString Bds::bdsStationAlias (
    Station station )
```

Returns the station alias if set else its name.

6.1.3.18 bdsDumpSelection()

```
void Bds::bdsDumpSelection (
    Selection sel )
```

Debug print out a [Selection](#) object.

6.1.3.19 bdsDumpSelectionInfo()

```
void Bds::bdsDumpSelectionInfo (
    SelectionInfo sel )
```

Debug print out a [SelectionInfo](#) object.

6.1.3.20 bdsDumpDataInfo()

```
void Bds::bdsDumpDataInfo (
    DataInfo dataInfo,
    int includeInfo )
```

Debug print out a [DataInfo](#) object.

6.1.3.21 bdsDumpChannelInfos()

```
void Bds::bdsDumpChannelInfos (
    const ChannelInfos & channelInfos )
```

Debug print out a [ChannelInfos](#) object.

6.1.3.22 bdsDumpData()

```
void Bds::bdsDumpData (
    const DataBlock & dataBlock,
    int nSamples )
```

Debug print out a [DataBlock](#) object.

6.1.3.23 bdsDumpLocation()

```
void Bds::bdsDumpLocation (
    Location location )
```

Debug printout location.

6.1.3.24 bdsDataChannelInfo()

```
BString Bds::bdsDataChannelInfo (
    const DataChannel & dataChannel )
```

Returns a string representation of a [DataChannel](#) object.

6.1.3.25 bdsDataChannelRef() [1/2]

```
BString Bds::bdsDataChannelRef (
    const DataChannel & dataChannel )
```

Returns the string reference name of a [DataChannel](#) object.

6.1.3.26 bdsDataChannelRef() [2/2]

```
BString Bds::bdsDataChannelRef (
    const ChannelInfo & channelInfo )
```

Returns the string reference name of a [ChannelInfo](#) object.

6.1.3.27 bdsDataChannelOverallResponse()

```
BError Bds::bdsDataChannelOverallResponse (
    const ChannelInfo & channelInfo,
    Response & response )
```

Returns the overall response from the list of responses in a [ChannelInfo](#).

6.1.3.28 bdsSelectionChannelInfo()

```
BString Bds::bdsSelectionChannelInfo (
    const Selection & selection,
    BUInt channel )
```

Returns a string describing the name and time period of a selection channel.

6.1.3.29 bdsPoleZeroGain()

```
double Bds::bdsPoleZeroGain (
    const PoleZero & poleZero,
    double frequency )
```

Calculates the overall gain of the given [PoleZero](#) transfer function.

6.1.3.30 bdsPoleZeroGainPhase()

```
void Bds::bdsPoleZeroGainPhase (
    const PoleZero & poleZero,
    double frequency,
    double & gain,
    double & phase )
```

Calculates the overall gain and phase of the given PoleZero transfer function.

6.1.3.31 bdsPoleZeroToFap()

```
void Bds::bdsPoleZeroToFap (
    const PoleZero & poleZero,
    BUInt nPoints,
    double calibrationFrequency,
    double sampleFrequency,
    BArray< Fap > & fap )
```

Convert PoleZero to FAP.

6.1.3.32 fileNameTime()

```
static BString Bds::fileNameTime (
    BTimeStamp t ) [static]
```

6.1.3.33 bdsFileNameExpand() [1/3]

```
BString Bds::bdsFileNameExpand (
    BString fileName,
    ChannelInfo & channelInfo )
```

Default filename from a ChannelInfo.

6.1.3.34 bdsFileNameExpand() [2/3]

```
BString Bds::bdsFileNameExpand (
    BString fileName,
    ChannelInfos & channelInfos )
```

Default filename from a list of ChannelInfo's.

6.1.3.35 bdsFileNameExpand() [3/3]

```
BString Bds::bdsFileNameExpand (
    BString fileName,
    Selection & sel )
```

Default filename from a [Selection](#).

6.1.3.36 bdsSpecialChannelsSet()

```
void Bds::bdsSpecialChannelsSet (
    const BList< SpecialChannel > specialChannels )
```

Set the special channels list.

6.1.3.37 bdsSpecialChannels()

```
BList< SpecialChannel > Bds::bdsSpecialChannels ( )
```

Return list of special channels.

6.1.3.38 bdsSpecialChannelIgnore()

```
Bool Bds::bdsSpecialChannelIgnore (
    BString network,
    BString station,
    BString channel )
```

Check if channel should be ignored.

6.1.3.39 seedChannelInstrumentCode()

```
char Bds::seedChannelInstrumentCode (
    BString dataType )
```

Returns SEED instrument code from dataType.

6.1.3.40 seedChannelDataType()

```
BString Bds::seedChannelDataType (
    BString channel )
```

Returns dataType from channel name based on SEED channel name convention.

6.1.3.41 bdsDataTypes()

```
BStringList Bds::bdsDataTypes ( )
```

Returns all known data types.

6.1.3.42 bdsUnits()

```
BStringList Bds::bdsUnits ( )
```

Returns all known SI units.

6.1.3.43 bdsUnitCase()

```
BString Bds::bdsUnitCase (
    BString unit )
```

Converts character case of units.

6.1.3.44 bdsUnCompressCm8()

```
BError Bds::bdsUnCompressCm8 (
    BUInt8 * buffer,
    BUInt n,
    BArray< BInt32 > & data )
```

Uncompress CM8 formatted data.

6.1.3.45 bdsUnCompressSteim1()

```
BError Bds::bdsUnCompressSteim1 (
    BUInt8 * buffer,
    BUInt n,
    BArray< BInt32 > & data )
```

Uncompress STEIM1 formatted data.

6.1.3.46 nullString()

```
static BString Bds::nullString (
    BString s ) [static]
```

6.1.3.47 crc()

```
BUInt32 Bds::crc (
    BUInt32 crc,
    void * data,
    int numBytes )
```

6.1.3.48 crcInit()

```
static void Bds::crcInit ( ) [static]
```

6.1.3.49 crc64()

```
static uint64_t Bds::crc64 (
    const void * buffer,
    const uint32_t len ) [static]
```

6.1.3.50 getHexString()

```
BString Bds::getHexString (
    char * data,
    int len )
```

6.1.3.51 duplicateDump()

```
int Bds::duplicateDump (
    DataBlock & data1,
    DataBlock & data2,
    int channel )
```

6.1.3.52 fixedString()

```
static BError Bds::fixedString (
    double v,
    int fieldWidth,
    int numDecimal,
    BString & str ) [static]
```

6.1.3.53 dataCalculateDifference()

```
void Bds::dataCalculateDifference (
    BInt32 & prevValue,
    BArray< BInt32 > & data )
```

6.1.3.54 dataCalculateUnDifference()

```
void Bds::dataCalculateUnDifference (
    BInt32 & prevValue,
    BArray< BInt32 > & data )
```

6.1.3.55 dataChecksum()

```
BInt32 Bds::dataChecksum (
    BInt32 checksum,
    BArray< BInt32 > & data )
```

6.1.3.56 dataCompressCm6()

```
BError Bds::dataCompressCm6 (
    int & prevValue1,
    int & prevValue2,
    BArray< BInt32 > & data,
    BString & d )
```

6.1.3.57 dataDeCompressCm6()

```
BError Bds::dataDeCompressCm6 (
    int & prevValue1,
    int & prevValue2,
    BString & d,
    BArray< BInt32 > & data )
```

6.1.3.58 dataConvert() [1/3]

```
static void Bds::dataConvert (
    const BArray< BFloat64 > & dataIn,
    BArray< BInt32 > & dataOut ) [static]
```

6.1.3.59 unitsCode()

```
static BString Bds::unitsCode (
    Response & r ) [static]
```

6.1.3.60 stringFormat()

```
static BString Bds::stringFormat (
    BTimeStamp t ) [static]
```

6.1.3.61 removeCR()

```
static BString Bds::removeCR (
    BString str ) [static]
```

6.1.3.62 fixedWidthValue()

```
BString Bds::fixedWidthValue (
    double v,
    int width )
```

This returns a double as a fixed width string truncating the data.

6.1.3.63 roundDigits()

```
static double Bds::roundDigits (
    double v,
    int nDigits ) [static]
```

6.1.3.64 bdsDataFileSeedLogWarning()

```
static void Bds::bdsDataFileSeedLogWarning (
    char * str ) [static]
```

6.1.3.65 bdsDataFileSeedLogError()

```
static void Bds::bdsDataFileSeedLogError (
    char * str ) [static]
```

6.1.3.66 seedTime()

```
static hptime_t Bds::seedTime (
    BTimeStamp t ) [static]
```

6.1.3.67 seedTimeString()

```
static BString Bds::seedTimeString (
    BTimeStamp t ) [static]
```

6.1.3.68 fromSeedTimeString()

```
static BTimeStamp Bds::fromSeedTimeString (
    BString str ) [static]
```

6.1.3.69 dataConvert() [2/3]

```
static void Bds::dataConvert (
    const BArray< BFloat64 > & dataIn,
    BArray< BInt32 > & dataOut ) [static]
```

6.1.3.70 dataConvert() [3/3]

```
static void Bds::dataConvert (
    const BArray< BFloat64 > & dataIn,
    BArray< BFloat32 > & dataOut ) [static]
```

6.1.3.71 record_handler()

```
static void Bds::record_handler (
    char * record,
    int reclen,
    void * info ) [static]
```

6.1.4 Variable Documentation**6.1.4.1 apiVersion**

```
const BUInt32 Bds::apiVersion = 0
```

6.1.4.2 bdsSpecialChannelsList

```
BList<SpecialChannel> Bds::bdsSpecialChannelsList [static]
```

6.1.4.3 seedICodeToDataTypes

```
SeedICodeToDataType Bds::seedICodeToDataTypes[]
```

Initial value:

```
= {
    { 'H', "seismic" },
    { 'L', "seismic" },
    { 'G', "seismic" },
    { 'M', "seismic" },
    { 'N', "seismic" },
    { 'Y', "data" },
    { 'A', "tilt" },
    { 'B', "creep" },
    { 'C', "calibration" },
    { 'D', "pressure" },
    { 'E', "testpoint" },
    { 'F', "magnetometer" },
    { 'I', "humidity" },
    { 'J', "rotation" },
    { 'K', "temperature" },
    { 'O', "waterCurrent" },
    { 'P', "geophone" },
    { 'Q', "voltage" },
    { 'R', "rainfall" },
    { 'S', "linearStrain" },
    { 'T', "tide" },
    { 'U', "bolometer" },
    { 'V', "volumetricStrain" },
    { 'W', "wind" },
    { 'X', "generated" },
    { 'Z', "beam" },
    { 0, 0 }
}
```

6.1.4.4 NetworkNameLen

```
const int Bds::NetworkNameLen = 3
```

Maximum [Network](#) name length.

6.1.4.5 StationNameLen

```
const int Bds::StationNameLen = 5
```

Maximum [Station](#) name length.

6.1.4.6 ChannelTypeLen

```
const int Bds::ChannelTypeLen = 3
```

Maximum [Channel](#) type name length.

6.1.4.7 ChannelAuxLen

```
const int Bds::ChannelAuxLen = 2
```

Maximum [Channel](#) Aux length.

6.1.4.8 SourceLen

```
const int Bds::SourceLen = 16
```

Maximum [Source](#) length.

6.1.4.9 BdsDataFileVersion

```
const BString Bds::BdsDataFileVersion = "1.2.0"
```

6.1.4.10 crcVec

```
uint64_t Bds::crcVec[256] [static]
```

6.1.4.11 crcInitDone

```
int Bds::crcInitDone [static]
```

6.1.4.12 cm6Table

```
char Bds::cm6Table[64] [static]
```

Initial value:

```
= {
    '+', '-', '0', '1', '2', '3', '4', '5',
    '6', '7', '8', '9', 'A', 'B', 'C', 'D',
    'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L',
    'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
    'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b',
    'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j',
    'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r',
    's', 't', 'u', 'v', 'w', 'x', 'y', 'z'
}
```

6.1.4.13 cm6TableRev

```
BUInt8 Bds::cm6TableRev[128] [static]
```

Initial value:

```
= {
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0x00, 0xff, 0x01, 0xff, 0xff,
    0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,
    0x0a, 0x0b, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0x0c, 0x0d, 0x0e, 0x0f, 0x10, 0x11, 0x12,
    0x13, 0x14, 0x15, 0x16, 0x17, 0x18, 0x19, 0x1a,
    0x1b, 0x1c, 0x1d, 0x1e, 0x1f, 0x20, 0x21, 0x22,
    0x23, 0x24, 0x25, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0x26, 0x27, 0x28, 0x29, 0x2a, 0x2b, 0x2c,
    0x2d, 0x2e, 0x2f, 0x30, 0x31, 0x32, 0x33, 0x34,
    0x35, 0x36, 0x37, 0x38, 0x39, 0x3a, 0x3b, 0x3c,
    0x3d, 0x3e, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff,
}
```

6.1.4.14 node_types

```
const char* Bds::node_types[]
```

Initial value:

```
= {
    "null", "document", "element", "pcdata", "cdata", "comment", "pi", "declaration"
}
```

6.1.4.15 Scale

```
const double Bds::Scale = 16777216.0
```

6.1.4.16 dataFormatAll

```
DataFormatAll Bds::dataFormatAll
```


Chapter 7

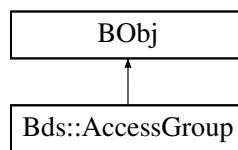
Class Documentation

7.1 Bds::AccessGroup Class Reference

This holds information on data access groups.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::AccessGroup:



Public Member Functions

- **AccessGroup** (**BUInt32** id=0, **BString** group= **BString**(), **BTimeStamp** startTime= **BTimeStamp**(), **BTimeStamp** endTime= **BTimeStamp**(), **BString** network= **BString**(), **BString** station= **BString**())
- **BString** **getType** ()
- **BError** **setMembers** (**BDictString** &members)
- **BError** **setMember** (**BString** name, **BString** value)
- **BError** **getMembers** (**BDictString** &members)
- **BError** **getMember** (**BString** name, **BString** &value)

Public Attributes

- **BUInt32** id
The unique id.
- **BString** group
The Group name.
- **BTimeStamp** startTime
The Start Time.
- **BTimeStamp** endTime
The End Time.
- **BString** network
The Network Name.
- **BString** station
The Station name.

7.1.1 Detailed Description

This holds information on data access groups.

A particular network:station may contain sensitive data. This database linked object links a period of data from a particular network:station to a security group.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AccessGroup()

```
Bds::AccessGroup::AccessGroup (
    BUInt32 id = 0,
    BString group = BString(),
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString() )
```

7.1.3 Member Function Documentation

7.1.3.1 getType()

```
BString Bds::AccessGroup::getType ( )
```

7.1.3.2 setMembers()

```
BError Bds::AccessGroup::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.1.3.3 setMember()

```
BError Bds::AccessGroup::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.1.3.4 getMembers()

```
BError Bds::AccessGroup::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.1.3.5 getMember()

```
BError Bds::AccessGroup::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.1.4 Member Data Documentation

7.1.4.1 id

```
BUInt32 Bds::AccessGroup::id
```

The unique id.

7.1.4.2 group

```
BString Bds::AccessGroup::group
```

The [Group](#) name.

7.1.4.3 startTime

```
BTimeStamp Bds::AccessGroup::startTime
```

The Start Time.

7.1.4.4 endTime

BTimeStamp Bds::AccessGroup::endTime

The End Time.

7.1.4.5 network

BString Bds::AccessGroup::network

The [Network](#) Name.

7.1.4.6 station

BString Bds::AccessGroup::station

The [Station](#) name.

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

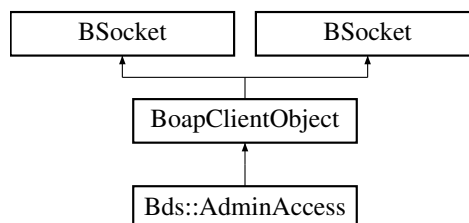
- [BdsD.h](#)
- [BdsD.cc](#)

7.2 Bds::AdminAccess Class Reference

This is the [AdminAccess](#) Access API interface.

```
#include <BdsC.h>
```

Inheritance diagram for Bds::AdminAccess:



Public Member Functions

- [AdminAccess](#) (**BString** name="")
BDS RPC access class.
- **BError** [connect](#) (**BString** user, **BString** password)
Provides user/password information.
- **BError** [validateUser](#) (**BString** user, **BString** email)
Checks the user given name or email.
- **BError** [setUser](#) (**BString** user, **BString** email)
Sets user to given name or email.
- **BError** [setUserReal](#) ()
Sets user back to real user.
- **BError** [getVersion](#) (**BString** &version, **BString** &name)
Gets the software version and server name.
- **BError** [userGetList](#) (**BList**< [User](#) > &users)
Get list of Users.
- **BError** [userUpdate](#) (**BInt32** append, [User](#) user, **BUInt32** &id)
Update or append a user entry.
- **BError** [userDelete](#) (**BUInt32** id)
Delete a user entry.
- **BError** [userGetFromId](#) (**BUInt32** id, [User](#) &user)
Get user info given user ID.
- **BError** [userGet](#) ([User](#) &user)
Get user info of the currently loogged in user.
- **BError** [userSet](#) ([User](#) user)
Set user info of the currently loogged in user.
- **BError** [userGetGroups](#) (**BList**< **BString** > &groups)
Get list of groups the user belongs to.
- **BError** [userGetOptions](#) (**BDict**< **BString** > &items)
Get users options.
- **BError** [userSetOptions](#) (**BDict**< **BString** > &items)
Set users options.
- **BError** [groupGetList](#) (**BList**< [Group](#) > &groups)
Get list of Groups.
- **BError** [groupUpdate](#) (**BInt32** append, [Group](#) group, **BUInt32** &id)
Update or append a group entry.
- **BError** [groupDelete](#) (**BUInt32** id)
Delete a group entry.
- **BError** [accessGroupGetList](#) (**BList**< [AccessGroup](#) > &accessGroups)
Get list of AccessGroups.
- **BError** [accessGroupUpdate](#) (**BInt32** append, [AccessGroup](#) group, **BUInt32** &id)
Update or append an [AccessGroup](#) entry.
- **BError** [accessGroupDelete](#) (**BUInt32** id)
Delete an [AccessGroup](#) entry.
- **BError** [getSelectionInfo](#) ([SelectionGroup](#) group, [Selection](#) selectionIn, [SelectionInfo](#) &selectionInfo)
Get information on possible selections. Use in GUI programs to list options available.
- **BError** [getSelections](#) ([SelectionGroup](#) group, [Selection](#) selectionIn, [Selection](#) &selectionOut)
Get selection list.
- **BError** [networkGetList](#) (**BList**< [Network](#) > &networks)
Get list of Networks.
- **BError** [networkUpdate](#) (**BInt32** append, [Network](#) network, **BUInt32** &id)

- Add or update a [Network](#) entry.

 - **BError** [networkDelete](#) (**BUInt32** id)

Delete a [Network](#) entry.
- **BError** [stationGetList](#) ([Selection](#) sel, **BList**< [Station](#) > &stations)

Get list of [Stations](#).
- **BError** [stationUpdate](#) (**BInt32** append, [Station](#) station, **BUInt32** &id)

Add or update a [Station](#) entry.
- **BError** [stationDelete](#) (**BUInt32** id)

Delete a [Station](#) entry.
- **BError** [locationGetList](#) ([Selection](#) sel, **BList**< [Location](#) > &locations)

Get list of [Station](#), [Channel](#) or both [Locations](#) depending on the sel.locationSelect parameter.
- **BError** [locationUpdate](#) (**BInt32** append, [Location](#) location, **BUInt32** &id)

Add or update a [Station Location](#) entry.
- **BError** [locationDelete](#) (**BUInt32** id)

Delete a [Station Location](#) entry.
- **BError** [channelGetList](#) ([Selection](#) sel, **BList**< [Channel](#) > &channels)

Get list of [Channels](#).
- **BError** [channelGet](#) (**BUInt32** id, [Channel](#) &channel)

Get a specific channel given its ID.
- **BError** [channelUpdate](#) (**BInt32** append, [Channel](#) channel, **BUInt32** &id)

Add or update a [Channel](#) entry.
- **BError** [channelDelete](#) (**BUInt32** id)

Delete a [Channel](#) entry.
- **BError** [sourceGetList](#) (**BList**< [Source](#) > &sources)

Get list of [Sources](#).
- **BError** [sourceUpdate](#) (**BInt32** append, [Source](#) source, **BUInt32** &id)

Add or update a [Source](#) entry.
- **BError** [sourceDelete](#) (**BUInt32** id)

Delete a [Source](#) entry.
- **BError** [sourcePriorityGetList](#) (**BList**< [SourcePriority](#) > &sourcePriorities)

Get list of [SourcePriorities](#).
- **BError** [sourcePriorityUpdate](#) (**BInt32** append, [SourcePriority](#) sourcePriority, **BUInt32** &id)
- **BError** [sourcePriorityDelete](#) (**BUInt32** id)
- **BError** [channellInstrumentGetList](#) ([Selection](#) sel, **BList**< [ChannellInstrument](#) > &channellInstruments)

Get list of [Channel Instruments](#).
- **BError** [channellInstrumentUpdate](#) (**BInt32** append, [ChannellInstrument](#) channellInstrument, **BUInt32** &id)

Add or update a [Instrument](#) entry.
- **BError** [channellInstrumentDelete](#) (**BUInt32** id)

Delete an [Instrument](#) entry.
- **BError** [digitiserGetList](#) ([Selection](#) sel, **BList**< [Digitiser](#) > &digitisers)

Get list of [Digitisers](#).
- **BError** [digitiserGet](#) (**BUInt32** id, [Digitiser](#) &digitiser)

Get a [Digitiser](#) object given its ID.
- **BError** [digitiserUpdate](#) (**BInt32** append, [Digitiser](#) digitiser, **BUInt32** &id)

Add or update a [Digitiser](#) entry.
- **BError** [digitiserDelete](#) (**BUInt32** id)

Delete a [Digitiser](#) entry.
- **BError** [sensorGetList](#) ([Selection](#) sel, **BList**< [Sensor](#) > &sensors)

Get list of [Sensors](#).
- **BError** [sensorGet](#) (**BUInt32** id, [Sensor](#) &sensor)

Get a [Sensor](#) object given its ID.

- **BError** [sensorUpdate](#) (**BInt32** append, [Sensor](#) sensor, **BUInt32** &id)
Add or update a [Sensor](#) entry.
- **BError** [sensorDelete](#) (**BUInt32** id)
Delete a [Sensor](#) entry.
- **BError** [calibrationGetList](#) ([Selection](#) sel, **BList**< [Calibration](#) > &calibrations)
Get list of [Calibrations](#).
- **BError** [calibrationUpdate](#) (**BInt32** append, [Calibration](#) calibration, **BUInt32** &id)
Add or update a [Calibration](#) entry.
- **BError** [calibrationDelete](#) (**BUInt32** id)
Delete a [Calibration](#) entry.
- **BError** [responseGetList](#) ([Selection](#) sel, **BList**< [Response](#) > &responses)
Get list of [Responses](#).
- **BError** [responseUpdate](#) (**BInt32** append, [Response](#) response, **BUInt32** &id)
Add or update a [Response](#) entry.
- **BError** [responseDelete](#) (**BUInt32** id)
Delete a [Response](#) entry.
- **BError** [eventGetList](#) ([Selection](#) sel, **BList**< [Event](#) > &events)
Get list of [Events](#).
- **BError** [eventUpdate](#) (**BInt32** append, [Event](#) event, **BUInt32** &id)
Add or update a [Event](#) entry.
- **BError** [eventDelete](#) (**BUInt32** id)
Delete a [Event](#) entry.
- **BError** [specialChannelGetList](#) ([Selection](#) sel, **BList**< [SpecialChannel](#) > &specialChannels)
Get list of [Special Channels](#).
- **BError** [specialChannelUpdate](#) (**BInt32** append, [SpecialChannel](#) specialChannel, **BUInt32** &id)
Add or update a [SpecialChannel](#) entry.
- **BError** [specialChannelDelete](#) (**BUInt32** id)
Delete a [SpecialChannel](#) entry.
- **BError** [metadataGetChannelInfo](#) ([Selection](#) sel, [ChannelInfos](#) &channelInfos)
Return the channel [MetaData](#) in structured form.
- **BError** [metadataGetFormatted](#) ([Selection](#) sel, **BString** format, **BArray**< **BUInt8** > &data)
Return the channel [MetaData](#) in a particular format.
- **BError** [dataFileGetList](#) ([Selection](#) sel, **BList**< [DataFileInfo](#) > &dataFile)
Get list of [DataFiles](#).
- **BError** [dataFileUpdate](#) (**BInt32** append, [DataFileInfo](#) dataFile, **BUInt32** &id)
Add or update a [DataFile](#) entry.
- **BError** [dataFileDelete](#) (**BUInt32** id)
Delete a [DataFile](#) entry.
- **BError** [dataChannelGetList](#) ([Selection](#) sel, **BList**< [DataChannel](#) > &dataChannel)
Get list of [DataChannels](#).
- **BError** [dataChannelUpdate](#) (**BInt32** append, [DataChannel](#) dataChannel, **BUInt32** &id)
Add or update a [DataChannel](#) entry.
- **BError** [dataChannelDelete](#) (**BUInt32** id)
Delete a [DataChannel](#) entry.
- **BError** [dataAvailability](#) ([Selection](#) selection, **BUInt32** num, **BArray**< [DataAvailChan](#) > &dataAvailChans)
Return availability for data matching the given selection parameters. If num > 0 segment ito this number of fixed time segments.
- **BError** [dataSearch](#) ([Selection](#) selection, [DataInfo](#) &dataInfo)
Search for data matching the given selection parameters.
- **BError** [dataGetChannelInfo](#) ([DataInfo](#) dataInfo, [ChannelInfos](#) &channelInfos)
Return the channel [MetaData](#) in structured form.

- **Error** `dataOpen` (`DataInfo` dataInfo, `BString` mode, `BString` format, `BUInt32` flags, `DataHandle` &data↵
Handle)
Open a data file.
- **Error** `dataGetInfo` (`DataHandle` dataHandle, `BUInt32` infoExtra, `DataInfo` &dataInfo)
Get information on the data file.
- **Error** `dataGetNotes` (`DataHandle` dataHandle, `BList`< `Note` > ¬es)
Get notes on the data file.
- **Error** `dataGetWarnings` (`DataHandle` dataHandle, `BList`< `BString` > &warnings)
Get information on the data file.
- **Error** `dataGetBlock` (`DataHandle` dataHandle, `BUInt32` channel, `BUInt32` segment, `BUInt32` block↵
Number, `DataBlock` & data)
Return a block of data.
- **Error** `dataSeekBlock` (`DataHandle` dataHandle, `BUInt32` channel, `BUInt32` segment, `BTimeStamp` time,
`BUInt32` &blockNumber)
Searches for a data block matching the time given.
- **Error** `dataSetInfo` (`DataHandle` dataHandle, `DataInfo` dataInfo)
Set the info when writing to a file.
- **Error** `dataPutBlock` (`DataHandle` dataHandle, `DataBlock` data)
Send a block of data.
- **Error** `dataClose` (`DataHandle` dataHandle, **Error** error, `BInt32` del)
Close a file.
- **Error** `dataFormattedRead` (`DataHandle` dataHandle, `BUInt32` number, `BArray`< `BUInt8` > & data)
Read the raw data from the file.
- **Error** `dataFormattedGetLength` (`DataHandle` dataHandle, `BUInt64` & length)
Read the length of the raw formatted data file.
- **Error** `dataRealtimeConfig` (`BInt32` enable, `Selection` sel)
Configures the sending of real-time data blocks.
- **Error** `dataRealtimeGet` (`BUInt32` numBlocks, `BUInt32` &numBlocksAvailable, `BList`< `DataBlockChannel`
> &dataBlocks)
Returns the number of data blocks available and up to numBlocks of these.
- **Error** `changeGroupStart` (`ChangeGroup` changeGroup)
Start a new `ChangeGroup` when making a set of changes to the BDS's database.
- **Error** `changeGroupEnd` ()
End a `ChangeGroup`.
- **Error** `changeGroupGetList` (`ListRange` range, `BList`< `ChangeGroup` > &changeGroups)
Return a list of `ChangeGroups`.
- **Error** `changeGroupDelete` (`BTimeStamp` beforeDate, `BString` type, `BInt32` empty)
Delete a `ChangeGroup`.
- **Error** `changeGetListNumber` (`BUInt32` id, `BUInt32` & number)
Get the number of changes in a `ChangeGroup`.
- **Error** `changeGetList` (`BUInt32` id, `ListRange` range, `BList`< `Change` > &changes)
Get a list of `Changes`.
- **Error** `changeDelete` (`BTimeStamp` beforeDate, `BString` type)
Delete a `Change`.
- **Error** `noteGetList` (`Selection` sel, `BList`< `Note` > ¬es)
Get a list of `Notes`.
- **Error** `noteUpdate` (`BInt32` append, `Note` note, `BUInt32` &id)
Add or update a `Note`.
- **Error** `noteDelete` (`BUInt32` id)
Delete a `Note`.
- **Error** `noteWriteDocument` (`BUInt32` id, `BString` format, `BArray`< `BUInt8` > data)

- Given a [Note](#) write a document associated with it.

 - **BError** [noteReadDocument](#) ([BUInt32](#) id, [BString](#) & format, [BArray](#)< [BUInt8](#) > & data)

Read a document associated with a [Note](#).
- **BError** [logGetList](#) ([LogSelect](#) sel, [BList](#)< [Log](#) > &logs)

Get list of log entries.
- **BError** [logUpdate](#) ([BInt32](#) append, [Log](#) log, [BUInt32](#) &id)

Add or Update a [Log](#) item.
- **BError** [logDelete](#) ([BUInt32](#) id)

Delete a [Log](#) item.
- **BError** [logAppend](#) ([BString](#) type, [BUInt32](#) priority, [BString](#) subSystem, [BString](#) title, [BString](#) description)

Append a log item.
- **BError** [statisticsGet](#) ([BDict](#)< [BString](#) > &info)

Get a list of system statistics.
- **BError** [serverConfigurationGet](#) ([BDict](#)< [BString](#) > &items)

Get server configuration parameters.
- **BError** [dataFormatGetList](#) ([BList](#)< [DataFormat](#) > &formats)

Get list of supported data formats.
- **BError** [transactionStart](#) ()

Starts a set of transactions.
- **BError** [transactionEnd](#) ([BInt32](#) abort)

Ends a set of transactions.
- **BError** [modeSet](#) ([Mode](#) mode, [Mode](#) &previousMode)

Changes the system mode from Master to slave.
- **BError** [modeSnapshotPause](#) ([BInt32](#) on)

Enables/disables backup synchronisation pause.
- **BError** [clean](#) ([CleanOptions](#) cleanOptions)

Cleans the system logs and Changes information.
- **BError** [databaseBackup](#) ([BString](#) &ref)

Backup the database.
- **BError** [databaseRestore](#) ([BString](#) ref, [BString](#) type)

Restore the database.
- **BError** [sqlQuery](#) ([BString](#) query, [BList](#)< [BDict](#)< [BString](#) > > &result)

A low level SQL access function.
- **BError** [extraCall](#) ([BUInt32](#) function, [BString](#) args, [BString](#) &result)

A special function to add new functions to the API prior to a full API update.

Additional Inherited Members

7.2.1 Detailed Description

This is the [AdminAccess](#) Access API interface.

This object provides the set of API functions that make RPC network calls to a BdsServer. It provides the full unresiticed data access API allowing all data read and write operations to be performed.

To use this BdsServer access object you would create one like "DataAccess bds;". Then you would connect it to a particular BdsServer running on a particular host using the connectService(in String) function. Once connected you need to login to the BdsServer as a particular user using the connect(in String user, in String password). After your have successfully logged in you can use the various objects methods. These result in a remote procedure call (RPC) to the BdsServer which Will imlement the function and return any data and normally an errors status (**BError**).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AdminAccess()

```
Bds::AdminAccess::AdminAccess (
    BString name = "" )
```

BDS RPC access class.

Parameters

<i>name</i>	The URL of the remote BOAP object name to connect to.
-------------	---

This object constructor takes the name of the remote object to connect to. However this is normally passed as "" as the **connectService()** function is normally used to perform the actual remote connection.

7.2.3 Member Function Documentation

7.2.3.1 connect()

```
AdminAccess::connect (
    BString user,
    BString password )
```

Provides user/password information.

Python: (**BError** err) = connect(PythonStr user, PythonStr password);

7.2.3.2 validateUser()

```
BError Bds::AdminAccess::validateUser (
    BString user,
    BString email )
```

Checks the user given name or email.

Python: (**BError** err) = validateUser(PythonStr user, PythonStr email);

Parameters

<i>user</i>	The user's name
<i>email</i>	The users email address

Checks to see if the user as named exists or if just the email address is provided, if any user with that email address exists.

7.2.3.3 setUser()

```
BError Bds::AdminAccess::setUser (
    BString user,
    BString email )
```

Sets user to given name or email.

Python: (**BError** err) = setUser(PythonStr user, PythonStr email);

Parameters

<i>user</i>	The user's name
<i>email</i>	The users email address

Changes the connection to be logged in as the given user by their user name or email address. This function will only be allowed if the current user is in the userSet group. The bdsAutodrm user normally has this set so it can change the BdsServer's connection to be that of the BdsAutodrm user for example.

7.2.3.4 setUserReal()

```
AdminAccess::setUserReal ( )
```

Sets user back to real user.

Python: (**BError** err) = [setUserReal\(\)](#);

7.2.3.5 getVersion()

```
AdminAccess::getVersion (
    BString & version,
    BString & name )
```

Gets the software version and server name.

Python: (**BError** err, PythonStr version, PythonStr name) = [getVersion\(\)](#);

7.2.3.6 userGetList()

```
BError Bds::AdminAccess::userGetList (
    BList< User > & users )
```

Get list of Users.

Python: (**BError** err, BList<User > users) = [userGetList\(\)](#);

Parameters

<i>users</i>	The returned list of all users
--------------	--------------------------------

This returns the complete list of users on the system

7.2.3.7 updateUser()

```
AdminAccess::userUpdate (
    BInt32 append,
    User user,
    BUInt32 & id )
```

Update or append a user entry.

Python: (**BError** err, PythonInt id) = userUpdate(BInt32 append, User user);

7.2.3.8 userDelete()

```
AdminAccess::userDelete (
    BUInt32 id )
```

Delete a user entry.

Python: (**BError** err) = userDelete(PythonInt id);

7.2.3.9 userGetFromId()

```
AdminAccess::userGetFromId (
    BUInt32 id,
    User & user )
```

Get user info given user ID.

Python: (**BError** err, User user) = userGetFromId(PythonInt id);

7.2.3.10 userGet()

```
AdminAccess::userGet (
    User & user )
```

Get user info of the currently loogged in user.

Python: (**BError** err, User user) = userGet();

7.2.3.11 userSet()

```
AdminAccess::userSet (
    User user )
```

Set user info of the currently loogged in user.

Python: (**BError** err) = [userSet\(User user\)](#);

7.2.3.12 userGetGroups()

```
AdminAccess::userGetGroups (
    BList< BString > & groups )
```

Get list of groups the user belongs to.

Python: (**BError** err, BList<BString > groups) = [userGetGroups\(\)](#);

7.2.3.13 userGetOptions()

```
AdminAccess::userGetOptions (
    BDict< BString > & items )
```

Get users options.

Python: (**BError** err, BDict<BString > items) = [userGetOptions\(\)](#);

7.2.3.14 userSetOptions()

```
AdminAccess::userSetOptions (
    BDict< BString > & items )
```

Set users options.

Python: (**BError** err, BDict<BString > items) = [userSetOptions\(\)](#);

7.2.3.15 groupGetList()

```
AdminAccess::groupGetList (
    BList< Group > & groups )
```

Get list of Groups.

Python: (**BError** err, BList<Group > groups) = [groupGetList\(\)](#);

7.2.3.16 groupUpdate()

```
AdminAccess::groupUpdate (
    BInt32 append,
    Group group,
    BUInt32 & id )
```

Update or append a group entry.

Python: (**BError** err, PythonInt id) = groupUpdate(BInt32 append, Group group);

7.2.3.17 groupDelete()

```
AdminAccess::groupDelete (
    BUInt32 id )
```

Delete a group entry.

Python: (**BError** err) = groupDelete(PythonInt id);

7.2.3.18 accessGroupGetList()

```
AdminAccess::accessGroupGetList (
    BList< AccessGroup > & accessGroups )
```

Get list of AccessGroups.

Python: (**BError** err, BList<AccessGroup > accessGroups) = [accessGroupGetList\(\)](#);

7.2.3.19 accessGroupUpdate()

```
AdminAccess::accessGroupUpdate (
    BInt32 append,
    AccessGroup group,
    BUInt32 & id )
```

Update or append an [AccessGroup](#) entry.

Python: (**BError** err, PythonInt id) = accessGroupUpdate(BInt32 append, AccessGroup group);

7.2.3.20 accessGroupDelete()

```
AdminAccess::accessGroupDelete (
    BUInt32 id )
```

Delete an [AccessGroup](#) entry.

Python: (**BError** err) = accessGroupDelete(PythonInt id);

7.2.3.21 getSelectionInfo()

```
BError Bds::AdminAccess::getSelectionInfo (
    SelectionGroup group,
    Selection selectionIn,
    SelectionInfo & selectionInfo )
```

Get information on possible selections. Use in GUI programs to list options available.

Python: (**BError** err, [SelectionInfo](#) selectionInfo) = getSelectionInfo(SelectionGroup group, Selection selectionIn);

Parameters

<i>group</i>	The type of selection info requested. Can be SelectionGroupData to get selections for Sensor data, SelectionGroupMetaData for Metadata selections and SelectionGroupDataWithCount as for SelectionGroupData but also returns the total number of DataChannels.
<i>selectionIn</i>	The current selection
<i>selectionInfo</i>	The returned selection information

Used for selection systems such as that within a Gui to restrict the options available given partial selection parameters. Given a basic [Selection](#) returns all of the [Selection](#) options available based on the [Sensor](#) data or Metadata in the BDS. For example if the selectionIn defines one particular [Network](#) and one particular [Station](#), the getSelectionInfo function will return a list of Channels and Sources for that base selection.

7.2.3.22 getSelections()

```
BError Bds::AdminAccess::getSelections (
    SelectionGroup group,
    Selection selectionIn,
    Selection & selectionOut )
```

Get selection list.

Python: (**BError** err, [Selection](#) selectionOut) = getSelections(SelectionGroup group, Selection selectionIn);

Parameters

<i>group</i>	The type of selection info requested. Currently ignored.
<i>selectionIn</i>	The selection which may include regular expressions and Station arrays.
<i>selectionOut</i>	The full Sensor data selection after regular expressions have been expanded and any Array's expanded into a list of Stations.

This is used to expand a [Sensor](#) data [Selection](#) to a full [Selection](#) list.

7.2.3.23 networkGetList()

```
AdminAccess::networkGetList (
    BList< Network > & networks )
```

Get list of Networks.

Python: (**BError** err, BList<Network > networks) = networkGetList();

7.2.3.24 networkUpdate()

```
AdminAccess::networkUpdate (
    BInt32 append,
    Network network,
    BUInt32 & id )
```

Add or update a [Network](#) entry.

Python: (**BError** err, PythonInt id) = networkUpdate(BInt32 append, Network network);

7.2.3.25 networkDelete()

```
AdminAccess::networkDelete (
    BUInt32 id )
```

Delete a [Network](#) entry.

Python: (**BError** err) = networkDelete(PythonInt id);

7.2.3.26 stationGetList()

```
BError Bds::AdminAccess::stationGetList (
    Selection sel,
    BList< Station > & stations )
```

Get list of Stations.

Python: (**BError** err, BList<Station > stations) = stationGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>stations</i>	The list of Stations matching the Selection .

This uses the [Network:Station](#) parts of the [Selection](#) object to return a list of matching Stations.

7.2.3.27 stationUpdate()

```
AdminAccess::stationUpdate (
    BInt32 append,
    Station station,
    BUInt32 & id )
```

Add or update a [Station](#) entry.

Python: (**BError** err, PythonInt id) = stationUpdate(BInt32 append, Station station);

7.2.3.28 stationDelete()

```
AdminAccess::stationDelete (
    BUInt32 id )
```

Delete a [Station](#) entry.

Python: (**BError** err) = stationDelete(PythonInt id);

7.2.3.29 locationGetList()

```
BError Bds::AdminAccess::locationGetList (
    Selection sel,
    BList< Location > & locations )
```

Get list of [Station](#), [Channel](#) or both Locations depending on the sel.locationSelect parameter.

Python: (**BError** err, BList<Location > locations) = locationGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>locations</i>	The list of Locations matching the Selection .

This uses the [Network:Station](#) and optionally the [Channel](#) parts of the [Selection](#) object to return a list of matching Stations. The [Selection.locationSelect](#) parameter can be set to `LocationSelectAll`, `LocationSelectStation` or `LocationSelectChannel`. If set to `LocationSelectStation` will return the locations of the Stations matched. If set to `LocationSelectChannel` will return the locations of the Channels matched. If set to `LocationSelectAll` (The default) will return the locations of the Channels matched if they have Locations or otherwise that of the Stations.

7.2.3.30 locationUpdate()

```
AdminAccess::locationUpdate (
    BInt32 append,
    Location location,
    BUInt32 & id )
```

Add or update a [Station Location](#) entry.

Python: (**BError** err, PythonInt id) = locationUpdate(BInt32 append, Location location);

7.2.3.31 locationDelete()

```
AdminAccess::locationDelete (
    BUInt32 id )
```

Delete a [Station Location](#) entry.

Python: (**BError** err) = locationDelete(PythonInt id);

7.2.3.32 channelGetList()

```
BError Bds::AdminAccess::channelGetList (
    Selection sel,
    BList< Channel > & channels )
```

Get list of Channels.

Python: (**BError** err, BList<Channel > channels) = channelGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>channels</i>	The list of Channels matching the Selection .

If the selection.channelId is set to a value other than 0, this will return a list of one [Channel](#) which matches that id. Otherwise it will return a list of Channels matchin the [Network:Station:Channel](#) regular expression fields of the selection.

7.2.3.33 channelGet()

```
AdminAccess::channelGet (
    BUInt32 id,
    Channel & channel )
```

Get a specific channel given its ID.

Python: (**BError** err, Channel channel) = channelGet(PythonInt id);

7.2.3.34 channelUpdate()

```
AdminAccess::channelUpdate (
    BInt32 append,
    Channel channel,
    BUInt32 & id )
```

Add or update a Channel entry.

Python: (**BError** err, PythonInt id) = channelUpdate(BInt32 append, Channel channel);

7.2.3.35 channelDelete()

```
AdminAccess::channelDelete (
    BUInt32 id )
```

Delete a Channel entry.

Python: (**BError** err) = channelDelete(PythonInt id);

7.2.3.36 sourceGetList()

```
AdminAccess::sourceGetList (
    BList< Source > & sources )
```

Get list of Sources.

Python: (**BError** err, BList<Source > sources) = sourceGetList();

7.2.3.37 sourceUpdate()

```
AdminAccess::sourceUpdate (
    BInt32 append,
    Source source,
    BUInt32 & id )
```

Add or update a Source entry.

Python: (**BError** err, PythonInt id) = sourceUpdate(BInt32 append, Source source);

7.2.3.38 sourceDelete()

```
AdminAccess::sourceDelete (
    BUInt32 id )
```

Delete a [Source](#) entry.

Python: (**BError** err) = sourceDelete(PythonInt id);

7.2.3.39 sourcePriorityGetList()

```
AdminAccess::sourcePriorityGetList (
    BList< SourcePriority > & sourcePrioritys )
```

Get list of SourcePriorities.

Python: (**BError** err, BList<SourcePriority > sourcePrioritys) = [sourcePriorityGetList\(\)](#);

7.2.3.40 sourcePriorityUpdate()

```
AdminAccess::sourcePriorityUpdate (
    BUInt32 append,
    SourcePriority sourcePriority,
    BUInt32 & id )
```

Python: (**BError** err, PythonInt id) = sourcePriorityUpdate(BInt32 append, SourcePriority sourcePriority);

7.2.3.41 sourcePriorityDelete()

```
AdminAccess::sourcePriorityDelete (
    BUInt32 id )
```

Python: (**BError** err) = sourcePriorityDelete(PythonInt id);

7.2.3.42 channelInstrumentGetList()

```
BError Bds::AdminAccess::channelInstrumentGetList (
    Selection sel,
    BList< ChannelInstrument > & channelInstruments )
```

Get list of [Channel](#) Instruments.

Python: (**BError** err, BList<ChannelInstrument > channelInstruments) = channelInstrumentGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>channelInstruments</i>	The list of ChannelInstruments matching the Selection .

This uses the Network:Station:Channel:Source fields of the Selection to return a list of matching Channel Instruments.

7.2.3.43 channelInstrumentUpdate()

```
AdminAccess::channelInstrumentUpdate (
    BInt32 append,
    ChannelInstrument channelInstrument,
    BUInt32 & id )
```

Add or update a Instrument entry.

Python: (**BError** err, PythonInt id) = channelInstrumentUpdate(BInt32 append, ChannelInstrument channelInstrument);

7.2.3.44 channelInstrumentDelete()

```
AdminAccess::channelInstrumentDelete (
    BUInt32 id )
```

Delete an Instrument entry.

Python: (**BError** err) = channelInstrumentDelete(PythonInt id);

7.2.3.45 digitiserGetList()

```
BError Bds::AdminAccess::digitiserGetList (
    Selection sel,
    BList< Digitiser > & digitisers )
```

Get list of Digitisers.

Python: (**BError** err, BList<Digitiser > digitisers) = digitiserGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>digitisers</i>	The list of Digitisers matching the Selection.

If selection.digitiserId is set, then return the single matching Digitiser. Otherwise use the Selection's Network:Station:Channel:Source fields to return a list of matching Digitisers.

7.2.3.46 digitiserGet()

```
AdminAccess::digitiserGet (
    BUInt32 id,
    Digitiser & digitiser )
```

Get a Digitiser object given its ID.

Python: (**BError** err, Digitiser digitiser) = digitiserGet(PythonInt id);

7.2.3.47 digitiserUpdate()

```
AdminAccess::digitiserUpdate (
    BInt32 append,
    Digitiser digitiser,
    BUInt32 & id )
```

Add or update a [Digitiser](#) entry.

Python: (**BError** err, PythonInt id) = digitiserUpdate(BInt32 append, Digitiser digitiser);

7.2.3.48 digitiserDelete()

```
AdminAccess::digitiserDelete (
    BUInt32 id )
```

Delete a [Digitiser](#) entry.

Python: (**BError** err) = digitiserDelete(PythonInt id);

7.2.3.49 sensorGetList()

```
BError Bds::AdminAccess::sensorGetList (
    Selection sel,
    BList< Sensor > & sensors )
```

Get list of Sensors.

Python: (**BError** err, BList<Sensor > sensors) = sensorGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>sensors</i>	The list of Sensors matching the Selection .

If selection.sensorId is set, then return the single matching [Sensor](#). Otherwise use the [Selection](#)'s Network:↔ Station:Channel:Source fields to return a list of matching Sensors.

7.2.3.50 sensorGet()

```
AdminAccess::sensorGet (
    BUInt32 id,
    Sensor & sensor )
```

Get a [Sensor](#) object given its ID.

Python: (**BError** err, [Sensor](#) sensor) = sensorGet(PythonInt id);

7.2.3.51 sensorUpdate()

```
AdminAccess::sensorUpdate (
    BInt32 append,
    Sensor sensor,
    BUInt32 & id )
```

Add or update a [Sensor](#) entry.

Python: (**BError** err, PythonInt id) = sensorUpdate(BInt32 append, Sensor sensor);

7.2.3.52 sensorDelete()

```
AdminAccess::sensorDelete (
    BUInt32 id )
```

Delete a [Sensor](#) entry.

Python: (**BError** err) = sensorDelete(PythonInt id);

7.2.3.53 calibrationGetList()

```
AdminAccess::calibrationGetList (
    Selection sel,
    BList< Calibration > & calibrations )
```

Get list of Calibrations.

Python: (**BError** err, BList<Calibration > calibrations) = calibrationGetList(Selection sel);

7.2.3.54 calibrationUpdate()

```
AdminAccess::calibrationUpdate (
    BInt32 append,
    Calibration calibration,
    BUInt32 & id )
```

Add or update a [Calibration](#) entry.

Python: (**BError** err, PythonInt id) = calibrationUpdate(BInt32 append, Calibration calibration);

7.2.3.55 calibrationDelete()

```
AdminAccess::calibrationDelete (
    BUInt32 id )
```

Delete a [Calibration](#) entry.

Python: (**BError** err) = calibrationDelete(PythonInt id);

7.2.3.56 responseGetList()

```
BError Bds::AdminAccess::responseGetList (
    Selection sel,
    BList< Response > & responses )
```

Get list of Responses.

Python: (**BError** err, BList<Response > responses) = responseGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>responses</i>	The list of Responses matching the Selection .

Use the [Selection](#)'s Network:Station:Channel:[Source](#) fields to return a list of matching Responses.

7.2.3.57 responseUpdate()

```
AdminAccess::responseUpdate (
    BInt32 append,
    Response response,
    BUInt32 & id )
```

Add or update a [Response](#) entry.

Python: (**BError** err, PythonInt id) = responseUpdate(BInt32 append, Response response);

7.2.3.58 responseDelete()

```
AdminAccess::responseDelete (
    BUInt32 id )
```

Delete a [Response](#) entry.

Python: (**BError** err) = responseDelete(PythonInt id);

7.2.3.59 eventGetList()

```
BError Bds::AdminAccess::eventGetList (
    Selection sel,
    BList< Event > & events )
```

Get list of Events.

Python: (**BError** err, BList<Event > events) = eventGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>events</i>	The list of Events matching the Selection .

If sel.id or sel.eventId is set then return the [Event](#) with this ID. Otherwise if sel.name is set return an [Event](#) with a title matching this regular expression between sel.startTime and sel.endTime.. Otherwise match any events between sel.startTime and sel.endTime.

7.2.3.60 eventUpdate()

```
AdminAccess::eventUpdate (
```

```

    BInt32 append,
    Event event,
    BUInt32 & id )

```

Add or update a [Event](#) entry.

Python: (**BError** err, PythonInt id) = eventUpdate(BInt32 append, Event event);

7.2.3.61 eventDelete()

```

AdminAccess::eventDelete (
    BUInt32 id )

```

Delete a [Event](#) entry.

Python: (**BError** err) = eventDelete(PythonInt id);

7.2.3.62 specialChannelGetList()

```

BError Bds::AdminAccess::specialChannelGetList (
    Selection sel,
    BList< SpecialChannel > & specialChannels )

```

Get list of Special Channels.

Python: (**BError** err, BList<SpecialChannel > specialChannels) = specialChannelGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>specialChannels</i>	The list of SpecialChannels matching the Selection .

This will return a list of SpecialChannels matchin the Network:Station:[Channel](#) regular expression fields of the selection.

7.2.3.63 specialChannelUpdate()

```

AdminAccess::specialChannelUpdate (
    BInt32 append,
    SpecialChannel specialChannel,
    BUInt32 & id )

```

Add or update a [SpecialChannel](#) entry.

Python: (**BError** err, PythonInt id) = specialChannelUpdate(BInt32 append, SpecialChannel specialChannel);

7.2.3.64 specialChannelDelete()

```

AdminAccess::specialChannelDelete (
    BUInt32 id )

```

Delete a [SpecialChannel](#) entry.

Python: (**BError** err) = specialChannelDelete(PythonInt id);

7.2.3.65 metadataGetChannelInfo()

```
BError Bds::AdminAccess::metadataGetChannelInfo (
    Selection sel,
    ChannelInfos & channelInfos )
```

Return the channel MetaData in structured form.

Python: (**BError** err, ChannelInfos channelInfos) = metadataGetChannelInfo(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>channelInfos</i>	Metadata information for the channels

This function is designed to help when exporting data from the BDS. It will attempt to gather all of the Metadata present for the Channels selected. Depending on the dataType of the selected Channels it will check that the expected Metadata is present and if not return an Error. For Channels of dataType "seismic" it will check that the following Metadata information exists: Station, Channel, Location, Calibration, Response, ChannelInstrument. For each of the selected Channels there will be an array of ChannelInfo objects. The BdsServer will segment the Metadata in time periods at each point an of this Metadata is changed. This each ChannelInfo object will return a set of Metadata over a time period that has no change in Metadata. When exporting data it can be time segmented to match this.

7.2.3.66 metadataGetFormatted()

```
BError Bds::AdminAccess::metadataGetFormatted (
    Selection sel,
    BString format,
    BArray< BUInt8 > & data )
```

Return the channel MetaData in a particular format.

Python: (**BError** err, BArray<BUInt8 > data) = metadataGetFormatted(Selection sel, PythonStr format);

Parameters

<i>sel</i>	Channels selection
<i>format</i>	The requested Metadata output format. This should match a format specified by any of the BDS data convertors.
<i>data</i>	The raw file data for the format given in bytes.

This function gathers all of the Metadata for the channels matching the Selection and then finds a suitable BDS data format conertor to export this Metadata in the chosen format. The resulation array of Byte data can be written to a suitable file.

7.2.3.67 dataFileGetList()

```
BError Bds::AdminAccess::dataFileGetList (
    Selection sel,
    BList< DataFileInfo > & dataFile )
```

Get list of DataFiles.

Python: (**BError** err, BList<DataFileInfo > dataFile) = dataFileGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>dataFiles</i>	The returned DataFiles

If sel.id is set, then returns the single [DataFile](#) matching this id. Otherwise uses the [Selection](#)'s Network:Station:↔ Channel:Source fields to select a list of DataFiles.

7.2.3.68 dataFileUpdate()

```
AdminAccess::dataFileUpdate (
    BInt32 append,
    DataFileInfo dataFile,
    BUInt32 & id )
```

Add or update a [DataFile](#) entry.

Python: (**BError** err, PythonInt id) = dataFileUpdate(BInt32 append, DataFileInfo dataFile);

7.2.3.69 dataFileDelete()

```
AdminAccess::dataFileDelete (
    BUInt32 id )
```

Delete a [DataFile](#) entry.

Python: (**BError** err) = dataFileDelete(PythonInt id);

7.2.3.70 dataChannelGetList()

```
BError Bds::AdminAccess::dataChannelGetList (
    Selection sel,
    BList< DataChannel > & dataChannel )
```

Get list of DataChannels.

Python: (**BError** err, BList<DataChannel > dataChannel) = dataChannelGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>dataChannels</i>	The returned DataChannel

Uses the [Selection](#)'s Network:Station:Channel:Source fields to select a list of DataChannels.

7.2.3.71 dataChannelUpdate()

```
AdminAccess::dataChannelUpdate (
    BInt32 append,
    DataChannel dataChannel,
    BUInt32 & id )
```

Add or update a [DataChannel](#) entry.

Python: (**BError** err, PythonInt id) = dataChannelUpdate(BInt32 append, DataChannel dataChannel);

7.2.3.72 dataChannelDelete()

```
AdminAccess::dataChannelDelete (
    BUInt32 id )
```

Delete a [DataChannel](#) entry.

Python: (**BError** err) = dataChannelDelete(PythonInt id);

7.2.3.73 dataAvailability()

```
BError Bds::AdminAccess::dataAvailability (
    Selection selection,
    BInt32 num,
    BArray< DataAvailChan > & dataAvailChans )
```

Return availability for data matching the given selection parameters. If num > 0 segment into this number of fixed time segments.

Python: (**BError** err, BArray<DataAvailChan > dataAvailChans) = dataAvailability(Selection selection, PythonInt num);

Parameters

<i>sel</i>	The selection information
<i>num</i>	The number of availability time segments to return.
<i>dataAvailChans</i>	Information on the availability of sensor data for the selected channel

This function will return a list of [Sensor](#) data availability information for the selected Channels. If num > 0 then it will restrict the number of time segments to that number. The time segments will be linearly spaced in time. If num ==0 then the system will return a data availability segment for each [DataChannel](#) continuous segment. This can be slow and return a very large number of segments.

7.2.3.74 dataSearch()

```
BError Bds::AdminAccess::dataSearch (
    Selection selection,
    DataInfo & dataInfo )
```

Search for data matching the given selection parameters.

Python: (**BError** err, [DataInfo](#) dataInfo) = dataSearch(Selection selection);

Parameters

<i>selection</i>	Channel Selection information
<i>dataInfo</i>	Sensor data selection information with data segmented by actual data file segments with extra information if available

This will perform a search for [Sensor](#) data matching the [Selection](#). Be careful with the selection as their could be a lot of data matching that will then hit resource limits. There are restrictions on the maximum time range allowed (MaxTimePeriod) and the maximum [DataChannel](#) segments returned (MaxNumChannels). The return [DataInfo](#) object can be interdated for information on the data, such as sampleRates and/or used in the [dataOpen\(\)](#) function for actual access to the [Sensor](#) data.

7.2.3.75 dataGetChannelInfo()

```
BError Bds::AdminAccess::dataGetChannelInfo (
    DataInfo dataInfo,
    ChannelInfos & channelInfos )
```

Return the channel MetaData in structured form.

Python: (**BError** err, [ChannelInfos](#) channelInfos) = dataGetChannelInfo(DataInfo dataInfo);

Parameters

<i>dataInfo</i>	Data selection
<i>channelInfos</i>	Metadata information for the channels

This function returns all of the Metadata for the given Channels specified in the dataInfo selection. It uses the [metadataGetChannelInfo\(\)](#) function converting the [DataInfo](#) selection to a more general [Selection](#) first.

7.2.3.76 dataOpen()

```
BError Bds::AdminAccess::dataOpen (
    DataInfo dataInfo,
    BString mode,
    BString format,
    BUInt32 flags,
    DataHandle & dataHandle )
```

Open a data file.

Parameters

in	<i>dataInfo</i>	The sensor data to open selection
in	<i>mode</i>	The open format. The mode should be set to "w" for writing data, "a" when appending data and "r" for reading data.
in	<i>format</i>	What format to open the data stream as. This can be API for raw BDS API access or one of the supported formats that the BDS is able to convert to.
in	<i>flags</i>	A bitset of flags as defined by Bds::DataFlags .
out	<i>dataHandle</i>	The handle for the open data set

Python: (**BError** err, [DataHandle](#) dataHandle) = dataOpen(DataInfo dataInfo, PythonStr mode, PythonStr format, PythonInt flags);

Parameters

<i>dataInfo</i>	Data selection defining the Sensor data Channels
<i>mode</i>	The open mode. This can be "r" for read access, "w" for write access and "aday" to append to a day file.
<i>format</i>	This is the format to use. It can be "API-SM" for multiplexed by sample or "API-CM" for multiplexed by channel. For reads it can also be one of the BDS data convertors supported formats.
<i>flags</i>	Bit set of DataFlags. Can include: DataFlagClipDataToTime, DataFlagClipDataToChannels, DataFlagMergeSegments and DataFlagNoMetadata
<i>dataHandle</i>	The returned data handle to access this data set

This function opens a set of [Sensor](#) data files in the BdsServer for read, write or append access. When the format is "API-SM" or "API-CM" then the raw data blocks can be accessed using the [dataSeekBlock\(\)](#) and [dataGetBlock\(\)](#) functions. **Note** that the "API-SM" format requires the Channels selected to be synchronously sampled and so can only be used for certain import/export data formats where this is the case. The flags are used when reading data. Normally whole contiguous DataBlocks are returned. The actual starttime and endtime of the overall data may thus be before and after the selection. The DataFlagClipDataToTime "clips" the first and last blocks, removing samples, so the time match the actual selection. The DataFlagClipDataToChannels "clips" the blocks so that all of the Channels contain data over the selection period as some Channels otherwise could be missing data for a portion of the [Selection](#). DataFlagMergeSegments merges data segments that have matching end and start times into one continuous segment. DataFlagNoMetadata ignores Metadata when exporting data.

7.2.3.77 dataGetInfo()

```
BError Bds::AdminAccess::dataGetInfo (
    DataHandle dataHandle,
    BUInt32 infoExtra,
    DataInfo & dataInfo )
```

Get information on the data file.

Python: (**BError** err, [DataInfo](#) dataInfo) = dataGetInfo(DataHandle dataHandle, PythonInt infoExtra);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>infoExtra</i>	A boolean flag to say to include the extra Metadata contained within the files
<i>dataInfo</i>	The resulting fully filled out DataInfo describing the data.

This functions returns fuller, more detailed, information on the [Sensor](#) data using Metadata stored within the data files themselves. The BDS data format files can store an extensive amount of Metadata from the various import formats supported. A fair amount of this Metadata is freeform and this is for general information purposes only. The returned sampleRate is the samplerate calculated on the actual data samples taking into account start and end times as well as actual numbers of continuous samples. The [DataInfo](#) returned has a set of time segmented information per [Channel](#). Each of the [DataInfo](#) segments defines a cotiguous set of data with no time discontinuities.

7.2.3.78 dataGetNotes()

```
BError Bds::AdminAccess::dataGetNotes (
    DataHandle dataHandle,
    BList< Note > & notes )
```

Get notes on the data file.

Python: (**BError** err, BList<Note > notes) = dataGetNotes(DataHandle dataHandle);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>notes</i>	A list of Notes on the data

Returns a list of system and user added notes for the data from the database.

7.2.3.79 dataGetWarnings()

```
BError Bds::AdminAccess::dataGetWarnings (
    DataHandle dataHandle,
    BList< BString > & warnings )
```

Get information on the data file.

Python: (**BError** err, BList<BString > warnings) = dataGetWarnings(DataHandle dataHandle);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>warnings</i>	A list of warning strings from the datafiles

Returns a list of warning strings contained within the data files, typically from the data file import convertors.

7.2.3.80 dataGetBlock()

```
BError Bds::AdminAccess::dataGetBlock (
    DataHandle dataHandle,
    BUInt32 channel,
    BUInt32 segment,
    BUInt32 blockNumber,
    DataBlock & data )
```

Return a block of data.

Python: (**BError** err, DataBlock data) = dataGetBlock(DataHandle dataHandle, PythonInt channel, PythonInt segment, PythonInt blockNumber);

Parameters

<i>dataHandle</i>	The opened data set's handle
-------------------	------------------------------

Parameters

<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>blockNumber</i>	The block number to fetch
<i>dataBlock</i>	The returned data block

This returns the given block number from the set of opened data files. If channel is other than 0 then it will return the data for the given channel number. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return a [DataBlock](#) with [Sensor](#) data for all of the channels. If Segment is 0 then the block number is referenced to the start of the set of files. If a particular segment number is given then the returned block number is from within that segment.

7.2.3.81 dataSeekBlock()

```
BError Bds::AdminAccess::dataSeekBlock (
    DataHandle dataHandle,
    BUInt32 channel,
    BUInt32 segment,
    BTimeStamp time,
    BUInt32 & blockNumber )
```

Searches for a data block matching the time given.

Python: (**BError** err, PythonInt blockNumber) = dataSeekBlock(DataHandle dataHandle, PythonInt channel, PythonInt segment, BTimeStamp time);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>time</i>	The time of a sample to search for
<i>blockNumber</i>	The returned block number

This seeks for the a [DataBlock](#) in the set of files that contains a sample for the time provided. If channel is other than 0 then it seeks for a [DataBlock](#) for the given channel. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return the location of a dataBlock containing all of the Channels. If Segment is 0 then the block number returned is referenced to the start of the set of files. If a particular segment number is given then the seek and the returned block number is within that segment.

7.2.3.82 dataSetInfo()

```
AdminAccess::dataSetInfo (
    DataHandle dataHandle,
    DataInfo dataInfo )
```

Set the info when writing to a file.

Python: (**BError** err) = dataSetInfo(DataHandle dataHandle, DataInfo dataInfo);

7.2.3.83 dataPutBlock()

```
AdminAccess::dataPutBlock (
    DataHandle dataHandle,
    DataBlock data )
```

Send a block of data.

Python: (**BError** err) = dataPutBlock(DataHandle dataHandle, DataBlock data);

7.2.3.84 dataClose()

```
AdminAccess::dataClose (
    DataHandle dataHandle,
    BError error,
    BInt32 del )
```

Close a file.

Python: (**BError** err) = dataClose(DataHandle dataHandle, BError error, BInt32 del);

7.2.3.85 dataFormattedRead()

```
BError Bds::AdminAccess::dataFormattedRead (
    DataHandle dataHandle,
    BUInt32 number,
    BArray< BUInt8 > & data )
```

Read the raw data from the file.

Python: (**BError** err, BArray<BUInt8 > data) = dataFormattedRead(DataHandle dataHandle, PythonInt number);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>number</i>	The maximum number of Bytes to read
<i>data</i>	The set of data bytes in the requested format

Given a handle to a set of data files opened for read in an external data format supported by the BDS data converters, read the raw formatted data bytes from the converted file.

7.2.3.86 dataFormattedGetLength()

```
AdminAccess::dataFormattedGetLength (
    DataHandle dataHandle,
    BUInt64 & length )
```

Read the length of the raw formatted data file.

Python: (**BError** err, PythonInt length) = dataFormattedGetLength(DataHandle dataHandle);

7.2.3.87 dataRealtimeConfig()

```
BError Bds::AdminAccess::dataRealtimeConfig (
    BInt32 enable,
    Selection sel )
```

Configures the sending of real-time data blocks.

Python: (**BError** err) = dataRealtimeConfig(**BInt32** enable, **Selection** sel);

Parameters

<i>enable</i>	Enable/disable the sending of a real-time data stream
<i>sel</i>	The selection of channels to send

This function has yet to be implemented. It is designed to enable or disable the sending of a stream of data from the BdsServer to a data viewing client.

7.2.3.88 dataRealtimeGet()

```
BError Bds::AdminAccess::dataRealtimeGet (
    BUInt32 numBlocks,
    BUInt32 & numBlocksAvailable,
    BList< DataBlockChannel > & dataBlocks )
```

Returns the number of data blocks available and up to numBlocks of these.

Python: (**BError** err, PythonInt numBlocksAvailable, **BList**<**DataBlockChannel** > dataBlocks) = dataRealtimeGet(PythonInt numBlocks);

Parameters

<i>numBlocks</i>	How many blocks to return at a time
<i>numBlocksAvailable</i>	The number of blocks available in this stream
<i>dataBlocks</i>	A list of the DataBlocks

This function has yet to be implemented. It is designed to return a set of real-time DataBlocks from the BdsServer to a data viewing client.

7.2.3.89 changeGroupStart()

```
AdminAccess::changeGroupStart (
    ChangeGroup changeGroup )
```

Start a new **ChangeGroup** when making a set of changes to the BDS's database.

Python: (**BError** err) = changeGroupStart(**ChangeGroup** changeGroup);

7.2.3.90 changeGroupEnd()

```
AdminAccess::changeGroupEnd ( )
```

End a [ChangeGroup](#).

Python: (**BError** err) = [changeGroupEnd\(\)](#);

7.2.3.91 changeGroupGetList()

```
AdminAccess::changeGroupGetList (
    ListRange range,
    BList< ChangeGroup > & changeGroups )
```

Return a list of ChangeGroups.

Python: (**BError** err, BList<ChangeGroup > changeGroups) = [changeGroupGetList\(ListRange range\)](#);

7.2.3.92 changeGroupDelete()

```
AdminAccess::changeGroupDelete (
    BTimeStamp beforeDate,
    BString type,
    BInt32 empty )
```

Delete a [ChangeGroup](#).

Python: (**BError** err) = [changeGroupDelete\(BTimeStamp beforeDate, PythonStr type, BInt32 empty\)](#);

7.2.3.93 changeGetListNumber()

```
AdminAccess::changeGetListNumber (
    BUInt32 id,
    BUInt32 & number )
```

Get the number of changes in a [ChangeGroup](#).

Python: (**BError** err, PythonInt number) = [changeGetListNumber\(PythonInt id\)](#);

7.2.3.94 changeGetList()

```
AdminAccess::changeGetList (
    BUInt32 id,
    ListRange range,
    BList< Change > & changes )
```

Get a list of Changes.

Python: (**BError** err, BList<Change > changes) = [changeGetList\(PythonInt id, ListRange range\)](#);

7.2.3.95 changeDelete()

```
AdminAccess::changeDelete (
    BTimeStamp beforeDate,
    BString type )
```

Delete a [Change](#).

Python: (**BError** err) = changeDelete(BTimeStamp beforeDate, PythonStr type);

7.2.3.96 noteGetList()

```
BError Bds::AdminAccess::noteGetList (
    Selection sel,
    BList< Note > & notes )
```

Get a list of Notes.

Python: (**BError** err, BList<Note > notes) = noteGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>notes</i>	The returned Notes

Uses the [Selection](#)'s Network:Station:Channel:[Source](#) fields to select a list of Notes.

7.2.3.97 noteUpdate()

```
AdminAccess::noteUpdate (
    BInt32 append,
    Note note,
    BUInt32 & id )
```

Add or update a [Note](#).

Python: (**BError** err, PythonInt id) = noteUpdate(BInt32 append, Note note);

7.2.3.98 noteDelete()

```
AdminAccess::noteDelete (
    BUInt32 id )
```

Delete a [Note](#).

Python: (**BError** err) = noteDelete(PythonInt id);

7.2.3.99 noteWriteDocument()

```
AdminAccess::noteWriteDocument (
    BUInt32 id,
    BString format,
    BArray< BUInt8 > data )
```

Given a [Note](#) write a document associated with it.

Python: (**BError** err) = noteWriteDocument(PythonInt id, PythonStr format, BArray<BUInt8 > data);

7.2.3.100 noteReadDocument()

```
AdminAccess::noteReadDocument (
    BUInt32 id,
    BString & format,
    BArray< BUInt8 > & data )
```

Read a document associated with a [Note](#).

Python: (**BError** err, PythonStr format, BArray<BUInt8 > data) = noteReadDocument(PythonInt id);

7.2.3.101 logGetList()

```
BError Bds::AdminAccess::logGetList (
    LogSelect sel,
    BList< Log > & logs )
```

Get list of log entries.

Python: (**BError** err, BList<Log > logs) = logGetList(LogSelect sel);

Parameters

<i>sel</i>	Log selection
<i>logs</i>	The returned Log

Uses the [LogSelect](#) to select a set of system logs and return these.

7.2.3.102 logUpdate()

```
AdminAccess::logUpdate (
    BInt32 append,
    Log log,
    BUInt32 & id )
```

Add or Update a [Log](#) item.

Python: (**BError** err, PythonInt id) = logUpdate(BInt32 append, Log log);

7.2.3.103 logDelete()

```
AdminAccess::logDelete (
    BUInt32 id )
```

Delete a [Log](#) item.

Python: (**BError** err) = logDelete(PythonInt id);

7.2.3.104 logAppend()

```
BError Bds::AdminAccess::logAppend (
    BString type,
    BUInt32 priority,
    BString subSystem,
    BString title,
    BString description )
```

Append a log item.

Python: (**BError** err) = logAppend(PythonStr type, PythonInt priority, PythonStr subSystem, PythonStr title, PythonStr description);

Parameters

<i>type</i>	The log entries type: error, warning, notice, debug or all.
<i>priority</i>	The log events priority from 0 to 5 with 5 being the highest priority
<i>subSystem</i>	The subsystem the log entry is from
<i>title</i>	A title for the log entry
<i>description</i>	A more detailed description of the log entry

This allows BDS xclients, perhaps data import clients, to add BDS system log entries.

7.2.3.105 statisticsGet()

```
BError Bds::AdminAccess::statisticsGet (
    BDict< BString > & info )
```

Get a list of system statistics.

Python: (**BError** err, BDict<BString > info) = [statisticsGet\(\)](#);

Parameters

<i>info</i>	A dictionary of BdsServer statistics
-------------	--------------------------------------

This returns runtime statistics from the BdsServer

7.2.3.106 serverConfigurationGet()

```
BError Bds::AdminAccess::serverConfigurationGet (
    BDict< BString > & items )
```

Get server configuration parameters.

Python: (**BError** err, **BDict**<**BString** > items) = [serverConfigurationGet\(\)](#);

Parameters

<i>info</i>	A dictionary of BdsServer configuration entries
-------------	---

This returns the list of server configuration parameters.

7.2.3.107 dataFormatGetList()

```
BError Bds::AdminAccess::dataFormatGetList (
    BList< DataFormat > & formats )
```

Get list of supported data formats.

Python: (**BError** err, **BList**<[DataFormat](#) > formats) = [dataFormatGetList\(\)](#);

Parameters

<i>formats</i>	A list of all data file formats
----------------	---------------------------------

This returns the list of all of the data file formats supported by the BdsServer's data convertors.

7.2.3.108 transactionStart()

```
BError Bds::AdminAccess::transactionStart ( )
```

Starts a set of transactions.

Python: (**BError** err) = [transactionStart\(\)](#);

Starts a database transaction. A database transaction allows an atomic update of a set of database parameters with the option to abort the whole change.

Parameters

<i>abort</i>	Abort or complete the transaction normally
--------------	--

Completes a database transaction with an option to abort the changes.

7.2.3.109 transactionEnd()

```
AdminAccess::transactionEnd (
    BInt32 abort )
```

Ends a set of transactions.

Python: (**BError** err) = [transactionEnd\(BInt32 abort\)](#);

7.2.3.110 modeSet()

```
BError Bds::AdminAccess::modeSet (
    Mode mode,
    Mode & previousMode )
```

Changes the system mode from Master to slave.

Python: (**BError** err, Mode previousMode) = [modeSet\(Mode mode\)](#);

Parameters

<i>mode</i>	The new server mode
<i>previousMode</i>	The current server mode returned

Changes the mode of a BdsServer's operation. A BdsServer can be in ModeMaster or ModeSlave modes. In ModeSlave it will act as a read only BdsServer which can be used for resilience, data protection or for performance purposes.

7.2.3.111 modeSnapshotPause()

```
BError Bds::AdminAccess::modeSnapshotPause (
    BInt32 on )
```

Enables/disables backup synchronisation pause.

Python: (**BError** err) = [modeSnapshotPause\(BInt32 on\)](#);

Parameters

<i>on</i>	Enabvle/disable pause
-----------	-----------------------

Preforms a snapshot pause. When in this mode all changes to the database and data files will be blocked untill the end of the pause. This allows a backup to be performed with consistant files.

7.2.3.112 clean()

```
BError Bds::AdminAccess::clean (
    CleanOptions cleanOptions )
```


Cleans the system logs and Changes information.

Python: (**BError** err) = [clean](#)([CleanOptions](#) cleanOptions);

Parameters

<i>cleanOptions</i>	Clean options. This has the options logs, changes and deletedFiles
---------------------	--

This function can be used to clean up old data. It should be called once in a while to perform this work. It will clean the given items that are over a year old.

7.2.3.113 databaseBackup()

```
BError Bds::AdminAccess::databaseBackup (
    BString & ref )
```

Backup the database.

Python: (**BError** err, PythonStr ref) = [databaseBackup\(\)](#);

Parameters

<i>ref</i>	Reference and file name component
------------	-----------------------------------

This will perform a complete BDS database backup into a file with the given reference name.

7.2.3.114 databaseRestore()

```
BError Bds::AdminAccess::databaseRestore (
    BString ref,
    BString type )
```

Restore the database.

Python: (**BError** err) = databaseRestore(PythonStr ref, PythonStr type);

Parameters

<i>ref</i>	Reference and file name component
<i>type</i>	The typed of database tables to restore. Can be admin, data, metadata or all

This will perform a restore of the selected sets of database tables

7.2.3.115 sqlQuery()

```
BError Bds::AdminAccess::sqlQuery (
    BString query,
    BList< BDict< BString > > & result )
```

A low level SQL access function.

Python: (**BError** err, **BList**< **BDict**<**BString** > > result) = sqlQuery(PythonStr query);

Parameters

<i>query</i>	The SQL query
<i>result</i>	A list of the returned strings from the database query

This will perform a raw database SQL query. Use with caution!

7.2.3.116 extraCall()

```
BError Bds::AdminAccess::extraCall (
    BUInt32 function,
    BString args,
    BString & result )
```

A special function to add new functions to the API prior to a full API update.

Python: (**BError** err, PythonStr result) = extraCall(PythonInt function, PythonStr args);

Parameters

<i>function</i>	Integer number of an additional function
<i>args</i>	Parameters to the function is string form
<i>result</i>	return from the function in string form

This has not been implemented but is there to add new functions to the API prior to a full API update

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

- [BdsC.h](#)
- [BdsC.cc](#)
- [BdsLib.dox](#)
- [/src/blacknest/bds/bds/doc/BdsPython.dox](#)
- [/src/blacknest/bds/bds/doc/bdsApi-extra.dox](#)

7.3 Bds::ArrayChannel Class Reference

This class defines an array's channel.

```
#include <BdsD.h>
```

Public Member Functions

- [ArrayChannel](#) (**BString** network= **BString**(), **BString** station= **BString**(), **BString** channel= **BString**(), **BFloat64** arrayOffsetEast=0, **BFloat64** arrayOffsetNorth=0)

Public Attributes

- **BString** [network](#)
The [Network](#) this station belongs to if for a particular network.
- **BString** [station](#)
The Stations name.
- **BString** [channel](#)
The Channels name.
- **BFloat64** [arrayOffsetEast](#)
The [Location](#) offset in in an array in an easterly direction in metres.
- **BFloat64** [arrayOffsetNorth](#)
The [Location](#) offset in in an array in a northerly direction in metres.

7.3.1 Detailed Description

This class defines an array's channel.

A [Station](#) could be an individual station or an array of stations. If it is an array of stations the [Station](#) object contains the list of Stations and Channels that make up the array.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 ArrayChannel()

```
Bds::ArrayChannel::ArrayChannel (
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BFloat64 arrayOffsetEast = 0,
    BFloat64 arrayOffsetNorth = 0 )
```

7.3.3 Member Data Documentation

7.3.3.1 network

```
BString Bds::ArrayChannel::network
```

The [Network](#) this station belongs to if for a particular network.

7.3.3.2 station

BString Bds::ArrayChannel::station

The Stations name.

7.3.3.3 channel

BString Bds::ArrayChannel::channel

The Channels name.

7.3.3.4 arrayOffsetEast

BFloat64 Bds::ArrayChannel::arrayOffsetEast

The [Location](#) offset in in an array in an easterly direction in metres.

7.3.3.5 arrayOffsetNorth

BFloat64 Bds::ArrayChannel::arrayOffsetNorth

The [Location](#) offset in in an array in a northerly direction in metres.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.4 Bds::BdsDataBlock Struct Reference

BdsDataFileBds: internal fixed size BDS Data Block.

```
#include <BdsDataFileBds.h>
```

Public Attributes

- [BdsDataBlockHeader](#) header
- char [data](#) [4]

The packet data.

7.4.1 Detailed Description

BdsDataFileBds: internal fixed size BDS Data Block.

7.4.2 Member Data Documentation

7.4.2.1 header

`BdsDataBlockHeader` `Bds::BdsDataBlock::header`

7.4.2.2 data

`char` `Bds::BdsDataBlock::data[4]`

The packet data.

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

- `/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h`

7.5 Bds::BdsDataBlockHeader Struct Reference

BdsDataFileBds: internal fixed size BDS Data Block header.

```
#include <BdsDataFileBds.h>
```

Public Attributes

- `BUInt32` `type`
Blocks type, contains magic number for synchronisation.
- `BUInt32` `length`
Packet length.
- `BUInt32` `packetOffset`
Offset to first packet header within block or 0 if no packet header within block.

7.5.1 Detailed Description

BdsDataFileBds: internal fixed size BDS Data Block header.

7.5.2 Member Data Documentation

7.5.2.1 type

BUInt32 Bds::BdsDataBlockHeader::type

Blocks type, contains magic number for synchronisation.

7.5.2.2 length

BUInt32 Bds::BdsDataBlockHeader::length

Packet length.

7.5.2.3 packetOffset

BUInt32 Bds::BdsDataBlockHeader::packetOffset

Offset to first packet header within block or 0 if no packet header within block.

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h](#)

7.6 Bds::BdsDataBlockPos Class Reference

BdsDataFileBds: internal file storage data block position.

```
#include <BdsDataFileBds.h>
```

Public Member Functions

- [BdsDataBlockPos](#) ([BTimeStamp](#) startTime=0, [BTimeStamp](#) endTime=0, [BUInt32](#) channel=0, [BUInt32](#) numChannels=0, [BUInt32](#) segment=0, [BUInt64](#) position=0, [BUInt64](#) numSamples=0)
- [int](#) operator< (const [BdsDataBlockPos](#) &b) const

Public Attributes

- **BTimeStamp** [startTime](#)
- **BTimeStamp** [endTime](#)
- **BUInt32** [channel](#)
- **BUInt32** [numChannels](#)
- **BUInt32** [segment](#)
- **BUInt64** [position](#)
- **BUInt64** [numSamples](#)

7.6.1 Detailed Description

BdsDataFileBds: internal file storage data block position.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 BdsDataBlockPos()

```
Bds::BdsDataBlockPos::BdsDataBlockPos (
    BTimeStamp startTime = 0,
    BTimeStamp endTime = 0,
    BUInt32 channel = 0,
    BUInt32 numChannels = 0,
    BUInt32 segment = 0,
    BUInt64 position = 0,
    BUInt64 numSamples = 0 ) [inline]
```

7.6.3 Member Function Documentation

7.6.3.1 operator<()

```
int Bds::BdsDataBlockPos::operator< (
    const BdsDataBlockPos & b ) const [inline]
```

7.6.4 Member Data Documentation

7.6.4.1 startTime

BTimeStamp Bds::BdsDataBlockPos::startTime

7.6.4.2 endTime

BTimeStamp Bds::BdsDataBlockPos::endTime

7.6.4.3 channel

BUInt32 Bds::BdsDataBlockPos::channel

7.6.4.4 numChannels

BUInt32 Bds::BdsDataBlockPos::numChannels

7.6.4.5 segment

BUInt32 Bds::BdsDataBlockPos::segment

7.6.4.6 position

BUInt64 Bds::BdsDataBlockPos::position

7.6.4.7 numSamples

BUInt64 Bds::BdsDataBlockPos::numSamples

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

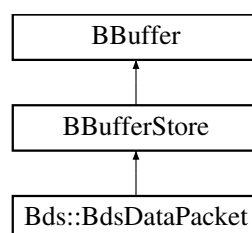
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h](#)

7.7 Bds::BdsDataPacket Class Reference

BdsDataFileBds: internal file storage packet.

```
#include <BdsDataFileBds.h>
```

Inheritance diagram for Bds::BdsDataPacket:



Public Member Functions

- [BdsDataPacket](#) ()
- [~BdsDataPacket](#) ()
- void [clear](#) ()
- void [reset](#) ()
- void [setChecksumAndLength](#) ()
- **BError** [validateChecksum](#) ()
- **BError** [setHeader](#) (const [BdsDataPacketHeader](#) &header)
- **BError** [getHeader](#) ([BdsDataPacketHeader](#) &header)
- void [dump](#) ()

Additional Inherited Members

7.7.1 Detailed Description

BdsDataFileBds: internal file storage packet.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 BdsDataPacket()

```
Bds::BdsDataPacket::BdsDataPacket ( )
```

7.7.2.2 ~BdsDataPacket()

```
Bds::BdsDataPacket::~~BdsDataPacket ( )
```

7.7.3 Member Function Documentation

7.7.3.1 clear()

```
void Bds::BdsDataPacket::clear ( )
```

7.7.3.2 reset()

```
void Bds::BdsDataPacket::reset ( )
```

7.7.3.3 setChecksumAndLength()

```
void Bds::BdsDataPacket::setChecksumAndLength ( )
```

7.7.3.4 validateChecksum()

```
BError Bds::BdsDataPacket::validateChecksum ( )
```

7.7.3.5 setHeader()

```
BError Bds::BdsDataPacket::setHeader (
    const BdsDataPacketHeader & header )
```

7.7.3.6 getHeader()

```
BError Bds::BdsDataPacket::getHeader (
    BdsDataPacketHeader & header )
```

7.7.3.7 dump()

```
void Bds::BdsDataPacket::dump ( )
```

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.cpp](#)

7.8 Bds::BdsDataPacketHeader Struct Reference

BdsDataFileBds internal file storage packet header.

```
#include <BdsDataFileBds.h>
```

Public Attributes

- **BUInt32** [type](#)
Packets type.
- **BUInt32** [length](#)
Length in bytes of packet.
- **BUInt32** [streamlet](#)
The streamlet id.
- **BUInt32** [sequence](#)
The streamlet packet sequence number.
- **BUInt32** [checksum](#)
Checksum of packet.
- **BTimeStamp** [startTime](#)
The time of the first sample.
- **BTimeStamp** [endTime](#)
The time of the last sample + 1.

7.8.1 Detailed Description

BdsDataFileBds internal file storage packet header.

7.8.2 Member Data Documentation

7.8.2.1 type

BUInt32 `Bds::BdsDataPacketHeader::type`

Packets type.

7.8.2.2 length

BUInt32 `Bds::BdsDataPacketHeader::length`

Length in bytes of packet.

7.8.2.3 streamlet

BUInt32 `Bds::BdsDataPacketHeader::streamlet`

The streamlet id.

7.8.2.4 sequence

BUInt32 Bds::BdsDataPacketHeader::sequence

The streamlet packet sequence number.

7.8.2.5 checksum

BUInt32 Bds::BdsDataPacketHeader::checksum

Checksum of packet.

7.8.2.6 startTime

BTimeStamp Bds::BdsDataPacketHeader::startTime

The time of the first sample.

7.8.2.7 endTime

BTimeStamp Bds::BdsDataPacketHeader::endTime

The time of the last sample + 1.

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h](#)

7.9 Bds::BdsDataSegment Class Reference

BdsDataFileBds: internal file storage data segment.

```
#include <BdsDataFileBds.h>
```

Public Member Functions

- [BdsDataSegment](#) ()
- int [operator<](#) (const [BdsDataSegment](#) &b) const

Public Attributes

- **BTimeStamp** [startTime](#)
- **BTimeStamp** [endTime](#)
- **BUInt32** [numBlocks](#)
- **BUInt32** [numSamples](#)
- **double** [sampleRate](#)
- **BArray**< [BdsDataBlockPos](#) > [blocks](#)

7.9.1 Detailed Description

BdsDataFileBds: internal file storage data segment.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 BdsDataSegment()

```
Bds::BdsDataSegment::BdsDataSegment ( ) [inline]
```

7.9.3 Member Function Documentation

7.9.3.1 operator<()

```
int Bds::BdsDataSegment::operator< (
    const BdsDataSegment & b ) const [inline]
```

7.9.4 Member Data Documentation

7.9.4.1 startTime

```
BTimeStamp Bds::BdsDataSegment::startTime
```

7.9.4.2 endTime

```
BTimeStamp Bds::BdsDataSegment::endTime
```

7.9.4.3 numBlocks

BUInt32 Bds::BdsDataSegment::numBlocks

7.9.4.4 numSamples

BUInt32 Bds::BdsDataSegment::numSamples

7.9.4.5 sampleRate

double Bds::BdsDataSegment::sampleRate

7.9.4.6 blocks

BArray<[BdsDataBlockPos](#)> Bds::BdsDataSegment::blocks

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h](#)

7.10 Bds::BdsDataStreamlet Class Reference

BdsDataFileBds: internal file storage data streamlet.

```
#include <BdsDataFileBds.h>
```

Public Member Functions

- [BdsDataStreamlet](#) ()

Public Attributes

- **BUInt32** [packetNumber](#)
- **BUInt64** [position](#)
- **BUInt32** [channel](#)
- **BUInt32** [numChannels](#)
- **BArray**< [BdsDataBlockPos](#) > [blocks](#)
- **BArray**< [BdsDataSegment](#) > [segments](#)

7.10.1 Detailed Description

BdsDataFileBds: internal file storage data streamlet.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 BdsDataStreamlet()

```
Bds::BdsDataStreamlet::BdsDataStreamlet ( ) [inline]
```

7.10.3 Member Data Documentation

7.10.3.1 packetNumber

```
BUInt32 Bds::BdsDataStreamlet::packetNumber
```

7.10.3.2 position

```
BUInt64 Bds::BdsDataStreamlet::position
```

7.10.3.3 channel

```
BUInt32 Bds::BdsDataStreamlet::channel
```

7.10.3.4 numChannels

```
BUInt32 Bds::BdsDataStreamlet::numChannels
```

7.10.3.5 blocks

```
BArray<BdsDataBlockPos> Bds::BdsDataStreamlet::blocks
```


7.10.3.6 segments

```
BArray<BdsDataSegment> Bds::BdsDataStreamlet::segments
```

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

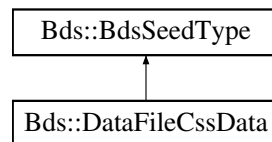
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h](#)

7.11 Bds::BdsSeedType Class Reference

BdsDataFileSeed internal parent for all SEED types.

```
#include <BdsSeedType.h>
```

Inheritance diagram for Bds::BdsSeedType:



Public Member Functions

- [BdsSeedType](#) ()
- **BError** [getInt](#) (char ** **data**, int size, int &v)
- **BError** [getUInt](#) (char ** **data**, int size, unsigned int &v)
- **BError** [getDouble](#) (char ** **data**, int size, double &v)
- **BError** [getString](#) (char ** **data**, int size, **BString** &v)
- **BError** [getStringVariable](#) (char ** **data**, int size, **BString** &v)
- **BError** [appendInt](#) (**BString** &s, int v, int size)
- **BError** [appendDouble](#) (**BString** &s, double v, int size, int precision)
- **BError** [appendExp](#) (**BString** &s, double v, int size, int precision, int sign)
- **BError** [appendString](#) (**BString** &s, **BString** v, int size)
- **BError** [appendStringVariable](#) (**BString** &s, **BString** v, int size)

7.11.1 Detailed Description

BdsDataFileSeed internal parent for all SEED types.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 BdsSeedType()

```
Bds::BdsSeedType::BdsSeedType ( )
```

7.11.3 Member Function Documentation

7.11.3.1 getInt()

```
BError Bds::BdsSeedType::getInt (
    char ** data,
    int size,
    int & v )
```

7.11.3.2 getUInt()

```
BError Bds::BdsSeedType::getUInt (
    char ** data,
    int size,
    unsigned int & v )
```

7.11.3.3 getDouble()

```
BError Bds::BdsSeedType::getDouble (
    char ** data,
    int size,
    double & v )
```

7.11.3.4 getString()

```
BError Bds::BdsSeedType::getString (
    char ** data,
    int size,
    BString & v )
```

7.11.3.5 getStringVariable()

```
BError Bds::BdsSeedType::getStringVariable (
    char ** data,
    int size,
    BString & v )
```

7.11.3.6 appendInt()

```
BError Bds::BdsSeedType::appendInt (
    BString & s,
    int v,
    int size )
```

7.11.3.7 appendDouble()

```
BError Bds::BdsSeedType::appendDouble (
    BString & s,
    double v,
    int size,
    int precision )
```

7.11.3.8 appendExp()

```
BError Bds::BdsSeedType::appendExp (
    BString & s,
    double v,
    int size,
    int precision,
    int sign )
```

7.11.3.9 appendString()

```
BError Bds::BdsSeedType::appendString (
    BString & s,
    BString v,
    int size )
```

7.11.3.10 appendStringVariable()

```
BError Bds::BdsSeedType::appendStringVariable (
    BString & s,
    BString v,
    int size )
```

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

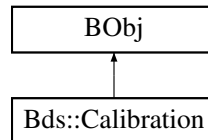
- /src/blacknest/bds/bds/bdsDataLib/BdsSeed/[BdsSeedType.h](#)
- /src/blacknest/bds/bds/bdsDataLib/BdsSeed/[BdsSeedType.cpp](#)

7.12 Bds::Calibration Class Reference

This class defines a calibration setting.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Calibration:



Public Member Functions

- **Calibration** (BUInt32 id=0, BTimeStamp startTime= BTimeStamp(), BTimeStamp endTime= BTimeStamp(), BString network= BString(), BString station= BString(), BString channel= BString(), BString source= BString(), BString name= BString(), BFloat64 samplingFrequency=0, BFloat64 calibrationFrequency=0, BFloat64 calibrationFactor=0, BString calibrationUnits= BString(), BString calibrationUnitsDesc= BString(), BFloat64 rawCalibrationFrequency=0, BFloat64 rawCalibrationFactor=0, BString rawCalibrationUnits= BString(), BFloat64 depth=0, BFloat64 waterLevel=0, BFloat64 horizontalAngle=0, BFloat64 verticalAngle=0)
- BString getType ()
- BError setMembers (BDictString &members)
- BError setMember (BString name, BString value)
- BError getMembers (BDictString &members)
- BError getMember (BString name, BString &value)

Public Attributes

- BUInt32 id
The ID.
- BTimeStamp startTime
The Start Time.
- BTimeStamp endTime
The End Time.
- BString network
The Network/Organisation Name.
- BString station
The station.
- BString channel
The channel.
- BString source
The source.
- BString name
The Calibrations name, "Main", "Measured".
- BFloat64 samplingFrequency
The sample rate used in Hz.
- BFloat64 calibrationFrequency
The post response compensation frequency that the CalibrationFactor value is valid for in Hz.

- **BFloat64** [calibrationFactor](#)
The post response compensation scaling value to apply to the data to normalise data to the units. This is a measured value at the calibration frequency.
- **BString** [calibrationUnits](#)
The post response compensation measurment SI units such as "m".
- **BString** [calibrationUnitsDesc](#)
A description of the calibrationUnits.
- **BFloat64** [rawCalibrationFrequency](#)
The raw data frequency that the CalibrationFactor value is valid for in Hz.
- **BFloat64** [rawCalibrationFactor](#)
The raw data scaling value to apply to the data to normalise data to the units. This is a measured value at the calibration frequency.
- **BString** [rawCalibrationUnits](#)
The raw data measurment SI units such as "m".
- **BFloat64** [depth](#)
The depth of the sensor below ground level in meters.
- **BFloat64** [waterLevel](#)
Elevation of the water surface in meters for underwater sites, where 0 is sea level.
- **BFloat64** [horizontalAngle](#)
The Sensors channel placement horizontal angle in degrees clockwise from north.
- **BFloat64** [verticalAngle](#)
The Sensors channel placement vertical angle in degrees degrees with zero = vertically up.

7.12.1 Detailed Description

This class defines a calibration setting.

Each channel has the core information of a `samplingFrequency` and a `calibrationFactor` (scaling factor) associated with it at a particular `calibrationFrequency`. The `calibrationUnits` defines the units of the sensor data after all of the [Response](#) frequency response's have been applied. Responses could add integration stages that will change the original sensor data's units to the given `calibrationUnits`. The `raw*` parameters, if given, are the raw sensor data's parameters before the Responses are applied. This is useful for simple raw data graph plotting etc. There may be additional calibration information such as the depth of the sensor and its positional angles.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Calibration()

```
Bds::Calibration::Calibration (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString(),
    BString name = BString(),
    BFloat64 samplingFrequency = 0,
```

```

BFloat64 calibrationFrequency = 0,
BFloat64 calibrationFactor = 0,
BString calibrationUnits = BString(),
BString calibrationUnitsDesc = BString(),
BFloat64 rawCalibrationFrequency = 0,
BFloat64 rawCalibrationFactor = 0,
BString rawCalibrationUnits = BString(),
BFloat64 depth = 0,
BFloat64 waterLevel = 0,
BFloat64 horizontalAngle = 0,
BFloat64 verticalAngle = 0 )

```

7.12.3 Member Function Documentation

7.12.3.1 getType()

```
BString Bds::Calibration::getType ( )
```

7.12.3.2 setMembers()

```

BError Bds::Calibration::setMembers (
    BDictString & members ) [virtual]

```

Reimplemented from **BObj**.

7.12.3.3 setMember()

```

BError Bds::Calibration::setMember (
    BString name,
    BString value ) [virtual]

```

Reimplemented from **BObj**.

7.12.3.4 getMembers()

```

BError Bds::Calibration::getMembers (
    BDictString & members ) [virtual]

```

Reimplemented from **BObj**.

7.12.3.5 getMember()

```
BError Bds::Calibration::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.12.4 Member Data Documentation

7.12.4.1 id

```
BUInt32 Bds::Calibration::id
```

The ID.

7.12.4.2 startTime

```
BTimeStamp Bds::Calibration::startTime
```

The Start Time.

7.12.4.3 endTime

```
BTimeStamp Bds::Calibration::endTime
```

The End Time.

7.12.4.4 network

```
BString Bds::Calibration::network
```

The Network/Organisation Name.

7.12.4.5 station

BString Bds::Calibration::station

The station.

7.12.4.6 channel

BString Bds::Calibration::channel

The channel.

7.12.4.7 source

BString Bds::Calibration::source

The source.

7.12.4.8 name

BString Bds::Calibration::name

The Calibrations name, "Main", "Measured".

7.12.4.9 samplingFrequency

BFloat64 Bds::Calibration::samplingFrequency

The sample rate used in Hz.

7.12.4.10 calibrationFrequency

BFloat64 Bds::Calibration::calibrationFrequency

The post response compensation frequency that the CalibrationFactor value is valid for in Hz.

7.12.4.11 calibrationFactor

BFloat64 Bds::Calibration::calibrationFactor

The post response compensation scaling value to apply to the data to normalise data to the units. This is a measured value at the calibration frequency.

7.12.4.12 calibrationUnits

BString Bds::Calibration::calibrationUnits

The post response compensation measurment SI units such as "m".

7.12.4.13 calibrationUnitsDesc

BString Bds::Calibration::calibrationUnitsDesc

A description of the calibrationUnits.

7.12.4.14 rawCalibrationFrequency

BFloat64 Bds::Calibration::rawCalibrationFrequency

The raw data frequency that the CalibrationFactor value is valid for in Hz.

7.12.4.15 rawCalibrationFactor

BFloat64 Bds::Calibration::rawCalibrationFactor

The raw data scaling value to apply to the data to normalise data to the units. This is a measured value at the calibration frequency.

7.12.4.16 rawCalibrationUnits

BString Bds::Calibration::rawCalibrationUnits

The raw data measurment SI units such as "m".

7.12.4.17 depth

BFloat64 Bds::Calibration::depth

The depth of the sensor below ground level in meters.

7.12.4.18 waterLevel

BFloat64 Bds::Calibration::waterLevel

Elevation of the water surface in meters for underwater sites, where 0 is sea level.

7.12.4.19 horizontalAngle

BFloat64 Bds::Calibration::horizontalAngle

The Sensors channel placement horizontal angle in degrees clockwise from north.

7.12.4.20 verticalAngle

BFloat64 Bds::Calibration::verticalAngle

The Sensors channel placement vertical angle in degrees degrees with zero = vertically up.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.13 Bds::CdChannel_1v0 Struct Reference

BdsDataFile: Internal CD1.0 channel information.

```
#include <BdsDataFileCd.h>
```

Public Attributes

- **BUInt8** [auth](#)
- **BUInt8** [compress](#)
- **BUInt8** [spare0](#)
- **BUInt8** [spare1](#)
- **BFloat32** [calibrationFactor](#)
- **BFloat32** [calibrationPeriod](#)
- char [name](#) [16]
- char [stationName](#) [16]
- char [channelName](#) [16]
- **BUInt32** [channel](#)

7.13.1 Detailed Description

BdsDataFile: Internal CD1.0 channel information.

7.13.2 Member Data Documentation

7.13.2.1 auth

BUInt8 Bds::CdChannel_1v0::auth

7.13.2.2 compress

BUInt8 Bds::CdChannel_1v0::compress

7.13.2.3 spare0

BUInt8 Bds::CdChannel_1v0::spare0

7.13.2.4 spare1

BUInt8 Bds::CdChannel_1v0::spare1

7.13.2.5 calibrationFactor

BFloat32 Bds::CdChannel_1v0::calibrationFactor

7.13.2.6 calibrationPeriod

BFloat32 Bds::CdChannel_1v0::calibrationPeriod

7.13.2.7 name

```
char Bds::CdChannel_lv0::name[16]
```

7.13.2.8 stationName

```
char Bds::CdChannel_lv0::stationName[16]
```

7.13.2.9 channelName

```
char Bds::CdChannel_lv0::channelName[16]
```

7.13.2.10 channel

```
BUInt32 Bds::CdChannel_lv0::channel
```

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h](#)

7.14 Bds::CdDataChannel Class Reference

BdsDataFile: Internal CD channel information.

```
#include <BdsDataFileCd.h>
```

Public Attributes

- **BString** [station](#)
- **BString** [channel](#)
- char [mode](#) [24]
- char [status](#) [32]
- **BTimeStamp** [startTime](#)
- **BUInt32** [period](#)
- **BUInt32** [numSamples](#)
- **BUInt32** [dataSize](#)
- **BUInt8** * [data](#)

7.14.1 Detailed Description

BdsDataFile: Internal CD channel information.

7.14.2 Member Data Documentation

7.14.2.1 station

BString Bds::CdDataChannel::station

7.14.2.2 channel

BString Bds::CdDataChannel::channel

7.14.2.3 mode

char Bds::CdDataChannel::mode[24]

7.14.2.4 status

char Bds::CdDataChannel::status[32]

7.14.2.5 startTime

BTimeStamp Bds::CdDataChannel::startTime

7.14.2.6 period

BUInt32 Bds::CdDataChannel::period

7.14.2.7 numSamples

```
BUInt32 Bds::CdDataChannel::numSamples
```

7.14.2.8 dataSize

```
BUInt32 Bds::CdDataChannel::dataSize
```

7.14.2.9 data

```
BUInt8* Bds::CdDataChannel::data
```

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h](#)

7.15 Bds::CdDataFormatFrame_1v0 Struct Reference

BdsDataFile: Internal CD1.0 frame information.

```
#include <BdsDataFileCd.h>
```

Public Attributes

- BUInt32 [frameType](#)
- BUInt32 [frameLength](#)
- BUInt32 [maxFrameLength](#)
- BUInt32 [numChannels](#)
- BUInt32 [period](#)
- [CdChannel_1v0 channels](#) [100]

7.15.1 Detailed Description

BdsDataFile: Internal CD1.0 frame information.

7.15.2 Member Data Documentation

7.15.2.1 frameType

BUInt32 Bds::CdDataFormatFrame_1v0::frameType

7.15.2.2 frameLength

BUInt32 Bds::CdDataFormatFrame_1v0::frameLength

7.15.2.3 maxFrameLength

BUInt32 Bds::CdDataFormatFrame_1v0::maxFrameLength

7.15.2.4 numChannels

BUInt32 Bds::CdDataFormatFrame_1v0::numChannels

7.15.2.5 period

BUInt32 Bds::CdDataFormatFrame_1v0::period

7.15.2.6 channels

CdChannel_1v0 Bds::CdDataFormatFrame_1v0::channels[100]

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

- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileCd.h](#)

7.16 Bds::CdFlag Class Reference

BdsDataFile: Internal CD flag.

```
#include <BdsDataFileCd.h>
```

Public Member Functions

- [CdFlag](#) ()

Public Attributes

- int [dead](#)
- int [zeroed](#)

7.16.1 Detailed Description

BdsDataFile: Internal CD flag.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 CdFlag()

```
Bds::CdFlag::CdFlag ( ) [inline]
```

7.16.3 Member Data Documentation

7.16.3.1 dead

```
int Bds::CdFlag::dead
```

7.16.3.2 zeroed

```
int Bds::CdFlag::zeroed
```

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h](#)

7.17 Bds::CdPacketData Class Reference

BdsDataFile: Internal CD data packet.

```
#include <BdsDataFileCd.h>
```


Public Attributes

- **BUInt32** [frameType](#)
- **BUInt32** [trailerOffset](#)
- char [creator](#) [8]
- char [destination](#) [8]
- **BUInt64** [sequenceNum](#)
- **BUInt32** [series](#)
- **BUInt32** [numChannels](#)
- **BUInt32** [period](#)
- **BTimeStamp** [startTime](#)
- **BArray**< [CdDataChannel](#) > [channels](#)
- **BUInt32** [authKey](#)
- **BUInt32** [authSize](#)
- char * [auth](#)
- **BUInt64** [crc](#)

7.17.1 Detailed Description

BdsDataFile: Internal CD data packet.

7.17.2 Member Data Documentation

7.17.2.1 frameType

BUInt32 Bds::CdPacketData::frameType

7.17.2.2 trailerOffset

BUInt32 Bds::CdPacketData::trailerOffset

7.17.2.3 creator

char Bds::CdPacketData::creator[8]

7.17.2.4 destination

char Bds::CdPacketData::destination[8]

7.17.2.5 **sequenceNum**

BUInt64 Bds::CdPacketData::sequenceNum

7.17.2.6 **series**

BUInt32 Bds::CdPacketData::series

7.17.2.7 **numChannels**

BUInt32 Bds::CdPacketData::numChannels

7.17.2.8 **period**

BUInt32 Bds::CdPacketData::period

7.17.2.9 **startTime**

BTimeStamp Bds::CdPacketData::startTime

7.17.2.10 **channels**

BArray<[CdDataChannel](#)> Bds::CdPacketData::channels

7.17.2.11 **authKey**

BUInt32 Bds::CdPacketData::authKey

7.17.2.12 **authSize**

BUInt32 Bds::CdPacketData::authSize

7.17.2.13 auth

```
char* Bds::CdPacketData::auth
```

7.17.2.14 crc

```
BUInt64 Bds::CdPacketData::crc
```

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

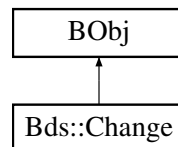
- /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h

7.18 Bds::Change Class Reference

This holds information on a Medatdata or [Sensor](#) data database change.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Change:



Public Member Functions

- [Change](#) (BUInt32 id=0, BUInt32 changeGroupId=0, BTimeStamp time= BTimeStamp(), BString type= BString(), BString table= BString(), BUInt32 rowId=0)
- BString [getType](#) ()
- BError [setMembers](#) (BDictString &members)
- BError [setMember](#) (BString name, BString value)
- BError [getMembers](#) (BDictString &members)
- BError [getMember](#) (BString name, BString &value)

Public Attributes

- BUInt32 [id](#)
The unique id.
- BUInt32 [changeGroupId](#)
The [Change](#) group ID.
- BTimeStamp [time](#)
The Time the change was made.
- BString [type](#)
The change type.
- BString [table](#)
The database table affected.
- BUInt32 [rowId](#)
The database row affected.

7.18.1 Detailed Description

This holds information on a Metadata or [Sensor](#) data database change.

Whenever a change is made to the BDS Metadata or [Sensor](#) data a [Change](#) object is added to the BDS Changes database. This describes which database table and object that was added or modified. [Change](#)'s are normally grouped together by a [ChangeGroup](#).

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Change()

```
Bds::Change::Change (
    BUInt32 id = 0,
    BUInt32 changeGroupId = 0,
    BTimeStamp time = BTimeStamp(),
    BString type = BString(),
    BString table = BString(),
    BUInt32 rowId = 0 )
```

7.18.3 Member Function Documentation

7.18.3.1 getType()

```
BString Bds::Change::getType ( )
```

7.18.3.2 setMembers()

```
BError Bds::Change::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from [BObj](#).

7.18.3.3 setMember()

```
BError Bds::Change::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from [BObj](#).

7.18.3.4 getMembers()

```
BError Bds::Change::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.18.3.5 getMember()

```
BError Bds::Change::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.18.4 Member Data Documentation

7.18.4.1 id

```
BUInt32 Bds::Change::id
```

The unique id.

7.18.4.2 changeGroupId

```
BUInt32 Bds::Change::changeGroupId
```

The [Change](#) group ID.

7.18.4.3 time

```
BTimeStamp Bds::Change::time
```

The Time the change was made.

7.18.4.4 type

BString Bds::Change::type

The change type.

7.18.4.5 table

BString Bds::Change::table

The database table affected.

7.18.4.6 rowId

BUInt32 Bds::Change::rowId

The database row affected.

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

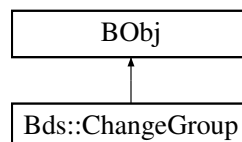
- [BdsD.h](#)
- [BdsD.cc](#)

7.19 Bds::ChangeGroup Class Reference

This holds information on a set of Changes.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::ChangeGroup:



Public Member Functions

- [ChangeGroup](#) (**BUInt32** id=0, **BTimeStamp** time= **BTimeStamp**(), **BString** type= **BString**(), **BString** user= **BString**(), **BString** title= **BString**(), **BString** description= **BString**())
- **BString** [getType](#) ()
- **BError** [setMembers](#) (**BDictString** &members)
- **BError** [setMember](#) (**BString** name, **BString** value)
- **BError** [getMembers](#) (**BDictString** &members)
- **BError** [getMember](#) (**BString** name, **BString** &value)

Public Attributes

- **BUInt32** [id](#)
The unique id.
- **BTimeStamp** [time](#)
The Time the change was made.
- **BString** [type](#)
The type of change.
- **BString** [user](#)
The user who made the change.
- **BString** [title](#)
The Changes title.
- **BString** [description](#)
The Description of the change.

7.19.1 Detailed Description

This holds information on a set of Changes.

A set of changes to the BDS database are grouped into a [ChangeGroup](#). This could be a set of changes whilst a user is logged in or by a program making a set of changes.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 ChangeGroup()

```
Bds::ChangeGroup::ChangeGroup (
    BUInt32 id = 0,
    BTimeStamp time = BTimeStamp(),
    BString type = BString(),
    BString user = BString(),
    BString title = BString(),
    BString description = BString() )
```

7.19.3 Member Function Documentation

7.19.3.1 getType()

```
BString Bds::ChangeGroup::getType ( )
```

7.19.3.2 setMembers()

```
BError Bds::ChangeGroup::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.19.3.3 setMember()

```
BError Bds::ChangeGroup::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.19.3.4 getMembers()

```
BError Bds::ChangeGroup::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.19.3.5 getMember()

```
BError Bds::ChangeGroup::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.19.4 Member Data Documentation

7.19.4.1 id

```
BUInt32 Bds::ChangeGroup::id
```

The unique id.

7.19.4.2 time

BTimeStamp Bds::ChangeGroup::time

The Time the change was made.

7.19.4.3 type

BString Bds::ChangeGroup::type

The type of change.

7.19.4.4 user

BString Bds::ChangeGroup::user

The user who made the change.

7.19.4.5 title

BString Bds::ChangeGroup::title

The Changes title.

7.19.4.6 description

BString Bds::ChangeGroup::description

The Description of the change.

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

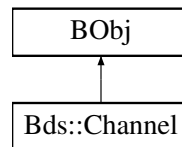
- [BdsD.h](#)
- [BdsD.cc](#)

7.20 Bds::Channel Class Reference

This class defines a seismic data [Channel](#).

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Channel:



Public Member Functions

- [Channel](#) ([BUInt32](#) id=0, [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTime](#)←
[Stamp](#)(), [BString](#) network= [BString](#)(), [BString](#) station= [BString](#)(), [BString](#) channel= [BString](#)(), [BString](#)
channelType= [BString](#)(), [BString](#) channelAux= [BString](#)(), [BString](#) dataType= [BString](#)(), [BString](#)
description= [BString](#)())
- [BString](#) getType ()
- [BError](#) setMembers ([BDictString](#) &members)
- [BError](#) setMember ([BString](#) name, [BString](#) value)
- [BError](#) getMembers ([BDictString](#) &members)
- [BError](#) getMember ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) id
Unique ID when stored in a database or for other uses.
- [BTimeStamp](#) startTime
The Start Time.
- [BTimeStamp](#) endTime
The End Time the channel was available.
- [BString](#) network
The [Network](#) Name.
- [BString](#) station
The Stations name.
- [BString](#) channel
The channels name (often as <channelType>_<channelAux>)
- [BString](#) channelType
The channels type (component of station field)
- [BString](#) channelAux
The channels auxiliary identifier (component of station field)
- [BString](#) dataType
The Type of data (seismic, seismicUnknown, data, log, unknown, empty)
- [BString](#) description
The channels description.

7.20.1 Detailed Description

This class defines a seismic data [Channel](#).

This class defines a seismic data channel with network:station:channel definitions. The class also splits the channel's name field into channelType and channelAux (channel name is <channelType>_<channelAux> to allow easy database searches etc. [Note](#) that the channelType and channelAux fields are calculated from the channel field and should not be changed. As well as seismic data, a [Channel](#) can contain other data types.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Channel()

```
Bds::Channel::Channel (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString channelType = BString(),
    BString channelAux = BString(),
    BString dataType = BString(),
    BString description = BString() )
```

7.20.3 Member Function Documentation

7.20.3.1 getType()

```
BString Bds::Channel::getType ( )
```

7.20.3.2 setMembers()

```
BError Bds::Channel::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from [BObj](#).

7.20.3.3 setMember()

```
BError Bds::Channel::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.20.3.4 getMembers()

```
BError Bds::Channel::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.20.3.5 getMember()

```
BError Bds::Channel::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.20.4 Member Data Documentation

7.20.4.1 id

```
BUInt32 Bds::Channel::id
```

Unique ID when stored in a database or for other uses.

7.20.4.2 startTime

```
BTimeStamp Bds::Channel::startTime
```

The Start Time.

7.20.4.3 endTime

BTimeStamp Bds::Channel::endTime

The End Time the channel was available.

7.20.4.4 network

BString Bds::Channel::network

The [Network](#) Name.

7.20.4.5 station

BString Bds::Channel::station

The Stations name.

7.20.4.6 channel

BString Bds::Channel::channel

The channels name (often as <channelType>_<channelAux>)

7.20.4.7 channelType

BString Bds::Channel::channelType

The channels type (component of station field)

7.20.4.8 channelAux

BString Bds::Channel::channelAux

The channels auxiliary identifier (component of station field)

7.20.4.9 dataType

BString Bds::Channel::dataType

The Type of data (seismic, seismicUnknown, data, log, unknown, empty)

7.20.4.10 description

BString Bds::Channel::description

The channels description.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.21 Bds::ChannelInfo Class Reference

This class provides full Metadata information on a channel.

```
#include <BdsD.h>
```

Public Member Functions

- **ChannelInfo** (**BTimeStamp** startTime= **BTimeStamp**(), **BTimeStamp** endTime= **BTimeStamp**(), **Station** station=Station(), **Location** stationLocation=Location(), **Channel** channel=Channel(), **Location** channelLocation=Location(), **BString** source= **BString**(), **Digitiser** digitiser=Digitiser(), **Sensor** sensor=Sensor(), **Calibration** calibration=Calibration(), **BList**< **Response** > responses= **BList**< **Response** >())

Public Attributes

- **BTimeStamp** startTime
The Start Time.
- **BTimeStamp** endTime
The End Time.
- **Station** station
The Station info.
- **Location** stationLocation
The Station location.
- **Channel** channel
The Channel data.
- **Location** channelLocation
The Channel location.
- **BString** source
The data source.
- **Digitiser** digitiser
The Digitiser in use.
- **Sensor** sensor
The Sensor in use.
- **Calibration** calibration
The Calibration info.
- **BList**< **Response** > responses
The list of frequency responses.

7.21.1 Detailed Description

This class provides full Metadata information on a channel.

This returns the Metadata available for a channel over a particular time period. There are likely to be multiple [ChannelInfo](#) objects over larger time periods, one for each change in Metadata. The [ChannelInfos](#) object contains an array of these [ChannelInfo](#) objects.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 ChannelInfo()

```
Bds::ChannelInfo::ChannelInfo (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    Station station = Station(),
    Location stationLocation = Location(),
    Channel channel = Channel(),
    Location channelLocation = Location(),
    BString source = BString(),
    Digitiser digitiser = Digitiser(),
    Sensor sensor = Sensor(),
    Calibration calibration = Calibration(),
    BList< Response > responses = BList<Response >() )
```

7.21.3 Member Data Documentation

7.21.3.1 startTime

BTimeStamp Bds::ChannelInfo::startTime

The Start Time.

7.21.3.2 endTime

BTimeStamp Bds::ChannelInfo::endTime

The End Time.

7.21.3.3 station

`Station` `Bds::ChannelInfo::station`

The `Station` info.

7.21.3.4 stationLocation

`Location` `Bds::ChannelInfo::stationLocation`

The `Station` location.

7.21.3.5 channel

`Channel` `Bds::ChannelInfo::channel`

The `Channel` data.

7.21.3.6 channelLocation

`Location` `Bds::ChannelInfo::channelLocation`

The `Channel` location.

7.21.3.7 source

`BString` `Bds::ChannelInfo::source`

The data source.

7.21.3.8 digitiser

`Digitiser` `Bds::ChannelInfo::digitiser`

The `Digitiser` in use.

7.21.3.9 sensor

[Sensor](#) Bds::ChannelInfo::sensor

The [Sensor](#) in use.

7.21.3.10 calibration

[Calibration](#) Bds::ChannelInfo::calibration

The [Calibration](#) info.

7.21.3.11 responses

[BList<Response >](#) Bds::ChannelInfo::responses

The list of frequency responses.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.22 Bds::ChannelInfos Class Reference

This class provides Metadata information on a set of channels.

```
#include <BdsD.h>
```

Public Member Functions

- [ChannelInfos](#) ([Station array=Station\(\)](#), [BArray< BArray< ChannelInfo > > channels= BArray< BArray< ChannelInfo > >\(\)](#))

Public Attributes

- [Station array](#)
The array if all channels are from an array.
- [BArray< BArray< ChannelInfo > > channels](#)
The per channel per time segment Metadata. For each channel there can be one or more [ChannelInfo](#) objects. A new [ChannelInfo](#) object is introduced whenever an item of Metadata changes.

7.22.1 Detailed Description

This class provides Metadata information on a set of channels.

This provides all of the Metadata information for a set of channels over a given time period. For each channel there can be one or more [ChannelInfo](#) objects each defining a set of consistent Metadata over a particular time period. The BdsServer will create a separate [ChannelInfo](#) object on each change in Metadata when the user asks for a set of metadata over some time period.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 ChannelInfos()

```
Bds::ChannelInfos::ChannelInfos (
    Station array = Station(),
    BArray< BArray< ChannelInfo > > channels = BArray< BArray<ChannelInfo > >()
)
```

7.22.3 Member Data Documentation

7.22.3.1 array

```
Station Bds::ChannelInfos::array
```

The array if all channels are from an array.

7.22.3.2 channels

```
BArray< BArray<ChannelInfo > > Bds::ChannelInfos::channels
```

The per channel per time segment Metadata. For each channel there can be one or more [ChannelInfo](#) objects. A new [ChannelInfo](#) object is introduced whenever an item of Metadata changes.

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

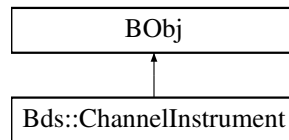
- [BdsD.h](#)
- [BdsD.cc](#)

7.23 Bds::ChannelInstrument Class Reference

This class defines a [Channel](#)'s instrument.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::ChannelInstrument:



Public Member Functions

- [ChannelInstrument](#) ([BUInt32](#) id=0, [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTimeStamp](#)(), [BUInt32](#) channelId=0, [BString](#) source= [BString](#)(), [BUInt32](#) digitiserId=0, [BUInt32](#) sensorId=0)
- [BString](#) getType ()
- [BError](#) setMembers ([BDictString](#) &members)
- [BError](#) setMember ([BString](#) name, [BString](#) value)
- [BError](#) getMembers ([BDictString](#) &members)
- [BError](#) getMember ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) id
Unique ID when stored in a database or for other uses.
- [BTimeStamp](#) startTime
The Start Time.
- [BTimeStamp](#) endTime
The End Time the channel was available.
- [BUInt32](#) channelId
The channels Id.
- [BString](#) source
The source.
- [BUInt32](#) digitiserId
The [Digitiser](#) in use.
- [BUInt32](#) sensorId
The sensor in use.

7.23.1 Detailed Description

This class defines a [Channel](#)'s instrument.

This [ChannelInstrument](#) links a seismic data channel with a particular sensor and digitiser. [Note](#) that it is possible to share sensors and digitisers between channels if wanted for generic sensor/digitiser definitions. However if particular serial numbers are needed the sensor/digitiser needs to be unique.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 ChannelInstrument()

```
Bds::ChannelInstrument::ChannelInstrument (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BUInt32 channelId = 0,
    BString source = BString(),
    BUInt32 digitiserId = 0,
    BUInt32 sensorId = 0 )
```

7.23.3 Member Function Documentation

7.23.3.1 getType()

```
BString Bds::ChannelInstrument::getType ( )
```

7.23.3.2 setMembers()

```
BError Bds::ChannelInstrument::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.23.3.3 setMember()

```
BError Bds::ChannelInstrument::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.23.3.4 getMembers()

```
BError Bds::ChannelInstrument::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.23.3.5 getMember()

```
BError Bds::ChannelInstrument::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.23.4 Member Data Documentation

7.23.4.1 id

```
BUInt32 Bds::ChannelInstrument::id
```

Unique ID when stored in a database or for other uses.

7.23.4.2 startTime

```
BTimeStamp Bds::ChannelInstrument::startTime
```

The Start Time.

7.23.4.3 endTime

```
BTimeStamp Bds::ChannelInstrument::endTime
```

The End Time the channel was available.

7.23.4.4 channelId

```
BUInt32 Bds::ChannelInstrument::channelId
```

The channels Id.

7.23.4.5 source

```
BString Bds::ChannelInstrument::source
```

The source.

7.23.4.6 digitiserId

```
BUInt32 Bds::ChannelInstrument::digitiserId
```

The [Digitiser](#) in use.

7.23.4.7 sensorId

```
BUInt32 Bds::ChannelInstrument::sensorId
```

The sensor in use.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.24 Bds::ChannelName Class Reference

This class defines a full channel name.

```
#include <BdsD.h>
```

Public Member Functions

- [ChannelName](#) ([BString](#) [network](#)= [BString](#)(), [BString](#) [station](#)= [BString](#)(), [BString](#) [channel](#)= [BString](#)(), [BString](#) [source](#)= [BString](#)())

Public Attributes

- **BString** [network](#)
The Channels network.
- **BString** [station](#)
The Channels station.
- **BString** [channel](#)
The Channels name.
- **BString** [source](#)
The Channels source.

7.24.1 Detailed Description

This class defines a full channel name.

A channel's data is fully defined by the Network:Station:Channel:[Source](#). This class stores all of these component names.

7.24.2 Constructor & Destructor Documentation

7.24.2.1 ChannelName()

```
Bds::ChannelName::ChannelName (
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString() )
```

7.24.3 Member Data Documentation

7.24.3.1 network

```
BString Bds::ChannelName::network
```

The Channels network.

7.24.3.2 station

```
BString Bds::ChannelName::station
```

The Channels station.

7.24.3.3 channel

```
BString Bds::ChannelName::channel
```

The Channels name.

7.24.3.4 source

```
BString Bds::ChannelName::source
```

The Channels source.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.25 Bds::CleanOptions Class Reference

This defines the set of clean options used in the clean() function.

```
#include <BdsD.h>
```

Public Member Functions

- [CleanOptions](#) ([BInt32](#) logs=0, [BInt32](#) changes=0, [BInt32](#) deletedFiles=0)

Public Attributes

- [BInt32](#) logs
Clean the Logs.
- [BInt32](#) changes
Clean the changes.
- [BInt32](#) deletedFiles
Clean deleted data files.

7.25.1 Detailed Description

This defines the set of clean options used in the clean() function.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 CleanOptions()

```
Bds::CleanOptions::CleanOptions (
    BInt32 logs = 0,
    BInt32 changes = 0,
    BInt32 deletedFiles = 0 )
```

7.25.3 Member Data Documentation

7.25.3.1 logs

BInt32 Bds::CleanOptions::logs

Clean the Logs.

7.25.3.2 changes

BInt32 Bds::CleanOptions::changes

Clean the changes.

7.25.3.3 deletedFiles

BInt32 Bds::CleanOptions::deletedFiles

Clean deleted data files.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.26 Bds::CompressSteim1 Class Reference

Steim1 un-compress class.

```
#include <BdsCompress.h>
```

Public Member Functions

- [CompressSteim1](#) ()
- void [setByteOrder](#) (int swap)
- void [clear](#) ()
- **BError** [unCompress](#) (void *buffer, **BUInt** numSamples, **BArray**< **BInt32** > & data)

7.26.1 Detailed Description

Steim1 un-compress class.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 CompressSteim1()

```
Bds::CompressSteim1::CompressSteim1 ( )
```

7.26.3 Member Function Documentation

7.26.3.1 setByteOrder()

```
void Bds::CompressSteim1::setByteOrder (
    int swap )
```

7.26.3.2 clear()

```
void Bds::CompressSteim1::clear ( )
```

7.26.3.3 unCompress()

```
BError Bds::CompressSteim1::unCompress (
    void * buffer,
    BUInt numSamples,
    BArray< BInt32 > & data )
```

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

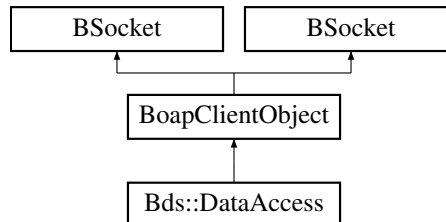
- [/src/blacknest/bds/bds/bdsDataLib/BdsCompress.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsCompress.cpp](#)

7.27 Bds::DataAccess Class Reference

This is the Data Access API interface to the BDS system.

```
#include <BdsC.h>
```

Inheritance diagram for Bds::DataAccess:



Public Member Functions

- **DataAccess** (**BString** name="")
BDS RPC access class.
- **BError connect** (**BString** user, **BString** password)
Provides user/password information for secure connection.
- **BError validateUser** (**BString** user, **BString** email)
Checks the user given name or email.
- **BError setUser** (**BString** user, **BString** email)
Sets user to given name or email.
- **BError setUserReal** ()
Sets user back to real user.
- **BError getVersion** (**BString** &version, **BString** &name)
Gets the software version and server name.
- **BError userGetFromId** (**BUInt32** id, **User** &user)
Get user info given user ID.
- **BError userGet** (**User** &user)
Get user info of the currently logged in user.
- **BError userSet** (**User** user)
Set user info of the currently logged in user.
- **BError userGetGroups** (**BList**< **BString** > &groups)
Get list of groups the user belongs to.
- **BError userGetOptions** (**BDict**< **BString** > &items)
Get users options.
- **BError userSetOptions** (**BDict**< **BString** > &items)
Set users options.
- **BError groupGetList** (**BList**< **Group** > &groups)
Get list of Groups.
- **BError networkGetList** (**BList**< **Network** > &networks)
Get list of Networks.
- **BError stationGetList** (**Selection** sel, **BList**< **Station** > &stations)
Get list of Stations.
- **BError channelGetList** (**Selection** sel, **BList**< **Channel** > &channels)
Get list of Channels.

- **BError** `sourceGetList (BList< Source > &sources)`
Get list of Sources.
- **BError** `sourcePriorityGetList (BList< SourcePriority > &sourcePriorities)`
Get list of SourcePriorities.
- **BError** `dataFileGetList (Selection sel, BList< DataFileInfo > &dataFile)`
Get list of DataFiles.
- **BError** `dataChannelGetList (Selection sel, BList< DataChannel > &dataChannel)`
Get list of DataChannels.
- **BError** `channelInstrumentGetList (Selection sel, BList< ChannelInstrument > &channelInstruments)`
Get list of Instruments.
- **BError** `digitiserGetList (Selection sel, BList< Digitiser > &digitisers)`
Get list of Digitisers.
- **BError** `digitiserGet (BUInt32 id, Digitiser &digitiser)`
Get a Digitiser object given its ID.
- **BError** `sensorGetList (Selection sel, BList< Sensor > &sensors)`
Get list of Sensors.
- **BError** `sensorGet (BUInt32 id, Sensor &sensor)`
Get a Sensor object given its ID.
- **BError** `calibrationGetList (Selection sel, BList< Calibration > &calibrations)`
Get list of Calibrations.
- **BError** `responseGetList (Selection sel, BList< Response > &responses)`
Get list of Responses.
- **BError** `locationGetList (Selection sel, BList< Location > &locations)`
Get list of Station, Channel or both Locations depending on the sel.locationSelect parameter.
- **BError** `eventGetList (Selection sel, BList< Event > &events)`
Get list of Events.
- **BError** `specialChannelGetList (Selection sel, BList< SpecialChannel > &specialChannels)`
Get list of Special Channels.
- **BError** `metadataGetChannelInfo (Selection sel, ChannelInfos &channelInfos)`
Return the channel MetaData in structured form.
- **BError** `metadataGetFormatted (Selection sel, BString format, BArray< BUInt8 > & data)`
Return the channel MetaData in a particular format.
- **BError** `getSelectionInfo (SelectionGroup group, Selection selectionIn, SelectionInfo &selectionInfo)`
Get information on possible selections. Use in GUI programs to list options available.
- **BError** `getSelections (SelectionGroup group, Selection selectionIn, Selection &selectionOut)`
Get selection list.
- **BError** `dataAvailability (Selection selection, BUInt32 num, BArray< DataAvailChan > &dataAvailChans)`
Return availability for data matching the given selection parameters. If num > 0 segment ito this number of fixed time segments.
- **BError** `dataSearch (Selection selection, DataInfo &dataInfo)`
Search for data matching the given selection parameters.
- **BError** `dataGetChannelInfo (DataInfo dataInfo, ChannelInfos &channelInfos)`
Return the channel MetaData in structured form.
- **BError** `dataOpen (DataInfo dataInfo, BString mode, BString format, BUInt32 flags, DataHandle &data↔ Handle)`
Open a data file.
- **BError** `dataGetInfo (DataHandle dataHandle, BUInt32 infoExtra, DataInfo &dataInfo)`
Get information on the data file.
- **BError** `dataGetNotes (DataHandle dataHandle, BList< Note > ¬es)`
Get notes on the data file.
- **BError** `dataGetWarnings (DataHandle dataHandle, BList< BString > &warnings)`

- Get information on the data file.*

 - **BError** [dataSeekBlock](#) ([DataHandle](#) dataHandle, **BUInt32** channel, **BUInt32** segment, **BTimeStamp** time, **BUInt32** &blockNumber)

Searches for a data block matching the time given.
- **BError** [dataGetBlock](#) ([DataHandle](#) dataHandle, **BUInt32** channel, **BUInt32** segment, **BUInt32** blockNumber, [DataBlock](#) & data)

Return a block of data.
- **BError** [dataClose](#) ([DataHandle](#) dataHandle, **BError** error, **BInt32** del)

Close a file.
- **BError** [dataFormattedRead](#) ([DataHandle](#) dataHandle, **BUInt32** number, **BArray**< **BUInt8** > & data)

Read the raw data from the file.
- **BError** [dataFormattedGetLength](#) ([DataHandle](#) dataHandle, **BUInt64** & length)

Read the length of the raw formatted data file.
- **BError** [dataRealtimeConfig](#) (**BInt32** enable, [Selection](#) sel)

Configures the sending of real-time data blocks.
- **BError** [dataRealtimeGet](#) (**BUInt32** numBlocks, **BUInt32** &numBlocksAvailable, **BList**< [DataBlockChannel](#) > &dataBlocks)

Returns the number of data blocks available and up to numBlocks of these.
- **BError** [noteGetList](#) ([Selection](#) sel, **BList**< [Note](#) > ¬es)

Return a list of Notes.
- **BError** [noteUpdate](#) (**BInt32** append, [Note](#) note, **BUInt32** &id)

Add or update a Note.
- **BError** [noteWriteDocument](#) (**BUInt32** id, **BString** format, **BArray**< **BUInt8** > data)

Given a Note write a document associated with it.
- **BError** [noteReadDocument](#) (**BUInt32** id, **BString** & format, **BArray**< **BUInt8** > & data)

Read a document associated with a Note.
- **BError** [logUpdate](#) (**BInt32** append, [Log](#) log, **BUInt32** &id)

Add or update a Log item.
- **BError** [logAppend](#) (**BString** type, **BUInt32** priority, **BString** subSystem, **BString** title, **BString** description)

Append a log item.
- **BError** [modeSet](#) ([Mode](#) mode, [Mode](#) &previousMode)

Changes the system mode from Master to slave.
- **BError** [modeSnapshotPause](#) (**BInt32** on)

Enables/disables backup synchronisation pause.
- **BError** [clean](#) ([CleanOptions](#) cleanOptions)

Cleans the system logs and Changes information.
- **BError** [databaseBackup](#) (**BString** &ref)

Backup the database.
- **BError** [statisticsGet](#) (**BDict**< **BString** > &info)

Get a list of system statistics.
- **BError** [serverConfigurationGet](#) (**BDict**< **BString** > &items)

Get server configuration parameters.
- **BError** [dataFormatGetList](#) (**BList**< [DataFormat](#) > &formats)

Get list of data formats.

Additional Inherited Members

7.27.1 Detailed Description

This is the Data Access API interface to the BDS system.

This object provides the set of API functions that make RPC network calls to a BdsServer. It provides the basic and restricted user orientated read only, data access API.

To use this BdsServer access object you would create one like "DataAccess bds;". Then you would connect it to a particular BdsServer running on a particular host using the connectService(in String) function. Once connected you need to login to the BdsServer as a particular user using the connect(in String user, in String password). After your have successfully logged in you can use the various objects methods. These result in a remote procedure call (RPC) to the BdsServer which Will imlement the function and return any data and normally an errors status (**BError**).

7.27.2 Constructor & Destructor Documentation

7.27.2.1 DataAccess()

```
Bds::DataAccess::DataAccess (
    BString name = "" )
```

BDS RPC access class.

Parameters

<i>name</i>	The URL of the remote BOAP object name to connect to.
-------------	---

This object constructor takes the name of the remote object to connect to. However this is normally passed as "" as the **connectService()** function is normally used to perform the actual remote connection.

7.27.3 Member Function Documentation

7.27.3.1 connect()

```
DataAccess::connect (
    BString user,
    BString password )
```

Provides user/password information for secure connection.

Python: (**BError** err) = connect(PythonStr user, PythonStr password);

7.27.3.2 validateUser()

```
BError Bds::DataAccess::validateUser (
    BString user,
    BString email )
```

Checks the user given name or email.

Python: (**BError** err) = validateUser(PythonStr user, PythonStr email);

Parameters

<i>user</i>	The user's name
<i>email</i>	The users email address

Checks to see if the user as named exists or if just the email address is provided, if any user with that email address exists.

7.27.3.3 setUser()

```
BError Bds::DataAccess::setUser (
    BString user,
    BString email )
```

Sets user to given name or email.

Python: (**BError** err) = setUser(PythonStr user, PythonStr email);

Parameters

<i>user</i>	The user's name
<i>email</i>	The users email address

Changes the connection to be logged in as the given user by their user name or email address. This function will only be allowed if the current user is in the userSet group. The bdsAutodrm user normally has this set so it can change the BdsServer's connection to be that of the BdsAutodrm user for example.

7.27.3.4 setUserReal()

```
DataAccess::setUserReal ( )
```

Sets user back to real user.

Python: (**BError** err) = [setUserReal\(\)](#);

7.27.3.5 getVersion()

```
DataAccess::getVersion (
    BString & version,
    BString & name )
```

Gets the software version and server name.

Python: (**BError** err, PythonStr version, PythonStr name) = [getVersion\(\)](#);

7.27.3.6 userGetFromId()

```
.DataAccess::userGetFromId (
    BUInt32 id,
    User & user )
```

Get user info given user ID.

Python: (**BError** err, [User](#) user) = userGetFromId(PythonInt id);

7.27.3.7 userGet()

```
.DataAccess::userGet (
    User & user )
```

Get user info of the currently loogged in user.

Python: (**BError** err, [User](#) user) = [userGet\(\)](#);

7.27.3.8 userSet()

```
.DataAccess::userSet (
    User user )
```

Set user info of the currently loogged in user.

Python: (**BError** err) = [userSet\(User user\)](#);

7.27.3.9 userGetGroups()

```
.DataAccess::userGetGroups (
    BList< BString > & groups )
```

Get list of groups the user belongs to.

Python: (**BError** err, **BList**<**BString**> groups) = [userGetGroups\(\)](#);

7.27.3.10 userGetOptions()

```
.DataAccess::userGetOptions (
    BDict< BString > & items )
```

Get users options.

Python: (**BError** err, **BDict**<**BString**> items) = [userGetOptions\(\)](#);

7.27.3.11 userSetOptions()

```
DataAccess::userSetOptions (
    BDict< BString > & items )
```

Set users options.

Python: (**BError** err, BDict<BString > items) = [userSetOptions\(\)](#);

7.27.3.12 groupGetList()

```
DataAccess::groupGetList (
    BList< Group > & groups )
```

Get list of Groups.

Python: (**BError** err, BList<Group > groups) = [groupGetList\(\)](#);

7.27.3.13 networkGetList()

```
DataAccess::networkGetList (
    BList< Network > & networks )
```

Get list of Networks.

Python: (**BError** err, BList<Network > networks) = [networkGetList\(\)](#);

7.27.3.14 stationGetList()

```
BError Bds::DataAccess::stationGetList (
    Selection sel,
    BList< Station > & stations )
```

Get list of Stations.

Python: (**BError** err, BList<Station > stations) = [stationGetList\(Selection sel\)](#);

Parameters

<i>sel</i>	The Selection
<i>stations</i>	The list of Stations matching the Selection .

This uses the [Network:Station](#) parts of the [Selection](#) object to return a list of matching Stations.

7.27.3.15 channelGetList()

```
BError Bds::DataAccess::channelGetList (
    Selection sel,
    BList< Channel > & channels )
```

Get list of Channels.

Python: (**BError** err, BList<Channel > channels) = channelGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>channels</i>	The list of Channels matching the Selection .

If the selection.channelId is set to a value other than 0, this will return a list of one [Channel](#) which matches that id. Otherwise it will return a list of Channels matchin the Network:Station:[Channel](#) regular expression fields of the selection.

7.27.3.16 sourceGetList()

```
DataAccess::sourceGetList (
    BList< Source > & sources )
```

Get list of Sources.

Python: (**BError** err, BList<Source > sources) = [sourceGetList\(\)](#);

7.27.3.17 sourcePriorityGetList()

```
DataAccess::sourcePriorityGetList (
    BList< SourcePriority > & sourcePrioritys )
```

Get list of SourcePriorities.

Python: (**BError** err, BList<SourcePriority > sourcePrioritys) = [sourcePriorityGetList\(\)](#);

7.27.3.18 dataFileGetList()

```
BError Bds::DataAccess::dataFileGetList (
    Selection sel,
    BList< DataFileInfo > & dataFile )
```

Get list of DataFiles.

Python: (**BError** err, BList<DataFileInfo > dataFile) = dataFileGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>dataFiles</i>	The returned DataFiles

If sel.id is set, then returns the single [DataFile](#) matching this id. Otherwise uses the [Selection](#)'s Network:Station:↵ Channel:[Source](#) fields to select a list of DataFiles.

7.27.3.19 dataChannelGetList()

```
BError Bds::DataAccess::dataChannelGetList (
    Selection sel,
    BList< DataChannel > & dataChannel )
```

Get list of DataChannels.

Python: (**BError** err, BList<DataChannel > dataChannel) = dataChannelGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>dataChannels</i>	The returned DataChannel

Uses the [Selection](#)'s Network:Station:Channel:Source fields to select a list of DataChannels.

7.27.3.20 channelInstrumentGetList()

```
BError Bds::DataAccess::channelInstrumentGetList (
    Selection sel,
    BList< ChannelInstrument > & channelInstruments )
```

Get list of Instruments.

Python: (**BError** err, BList<ChannelInstrument > channelInstruments) = channelInstrumentGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>channelInstruments</i>	The list of ChannelInstruments matching the Selection .

This uses the Network:Station:Channel:Source fields of the [Selection](#) to return a list of matching [Channel](#) Instruments.

7.27.3.21 digitiserGetList()

```
BError Bds::DataAccess::digitiserGetList (
    Selection sel,
    BList< Digitiser > & digitisers )
```

Get list of Digitisers.

Python: (**BError** err, BList<Digitiser > digitisers) = digitiserGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>digitisers</i>	The list of Digitisers matching the Selection .

If selection.digitiserId is set, then return the single matching [Digitiser](#). Otherwise use the [Selection](#)'s Network:↔ Station:Channel:Source fields to return a list of matching Digitisers.

7.27.3.22 digitiserGet()

```

DataAccess::digitiserGet (
    BUInt32 id,
    Digitiser & digitiser )

```

Get a [Digitiser](#) object given its ID.

Python: (**BError** err, [Digitiser](#) digitiser) = digitiserGet(PythonInt id);

7.27.3.23 sensorGetList()

```

BError Bds::DataAccess::sensorGetList (
    Selection sel,
    BList< Sensor > & sensors )

```

Get list of Sensors.

Python: (**BError** err, BList<Sensor > sensors) = sensorGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>sensors</i>	The list of Sensors matching the Selection .

If selection.sensorId is set, then return the single matching [Sensor](#). Otherwise use the [Selection](#)'s Network:↔ Station:Channel:Source fields to return a list of matching Sensors.

7.27.3.24 sensorGet()

```

DataAccess::sensorGet (
    BUInt32 id,
    Sensor & sensor )

```

Get a [Sensor](#) object given its ID.

Python: (**BError** err, [Sensor](#) sensor) = sensorGet(PythonInt id);

7.27.3.25 calibrationGetList()

```

DataAccess::calibrationGetList (
    Selection sel,
    BList< Calibration > & calibrations )

```

Get list of Calibrations.

Python: (**BError** err, BList<Calibration > calibrations) = calibrationGetList(Selection sel);

7.27.3.26 responseGetList()

```
BError Bds::DataAccess::responseGetList (
    Selection sel,
    BList< Response > & responses )
```

Get list of Responses.

Python: (**BError** err, BList<Response > responses) = responseGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>responses</i>	The list of Responses matching the Selection .

Use the [Selection](#)'s Network:Station:Channel:[Source](#) fields to return a list of matching Responses.

7.27.3.27 locationGetList()

```
BError Bds::DataAccess::locationGetList (
    Selection sel,
    BList< Location > & locations )
```

Get list of [Station](#), [Channel](#) or both Locations depending on the sel.locationSelect parameter.

Python: (**BError** err, BList<Location > locations) = locationGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>locations</i>	The list of Locations matching the Selection .

This uses the [Network:Station](#) and optionally the [Channel](#) parts of the [Selection](#) object to return a list of matching Stations. The [Selection.locationSelect](#) parameter can be set to LocationSelectAll, LocationSelectStation or LocationSelectChannel. If set to LocationSelectStation will return the locations of the Stations matched. If set to LocationSelectChannel will return the locations of the Channels matched. If set to LocationSelectAll (The default) will return the locations of the Channels matched if they have Locations or otherwise that of the Stations.

7.27.3.28 eventGetList()

```
BError Bds::DataAccess::eventGetList (
    Selection sel,
    BList< Event > & events )
```

Get list of Events.

Python: (**BError** err, BList<Event > events) = eventGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>events</i>	The list of Events matching the Selection .

If sel.id or sel.eventId is set then return the [Event](#) with this ID. Otherwise if sel.name is set return an [Event](#) with a title matching this regular expression between sel.startTime and sel.endTime.. Otherwise match any events between sel.startTime and sel.endTime.

7.27.3.29 specialChannelGetList()

```
BError Bds::DataAccess::specialChannelGetList (
    Selection sel,
    BList< SpecialChannel > & specialChannels )
```

Get list of Special Channels.

Python: (**BError** err, BList<SpecialChannel > specialChannels) = specialChannelGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>specialChannels</i>	The list of SpecialChannels matching the Selection .

This will return a list of SpecialChannels matchin the Network:Station:[Channel](#) regular expression fields of the selection.

7.27.3.30 metadataGetChannelInfo()

```
BError Bds::DataAccess::metadataGetChannelInfo (
    Selection sel,
    ChannelInfos & channelInfos )
```

Return the channel MetaData in structured form.

Python: (**BError** err, [ChannelInfos](#) channelInfos) = metadataGetChannelInfo(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>channelInfos</i>	Metadata information for the channels

This function is designed to help when exporting data from the BDS. It will attempt to gather all of the Metadata present for the Channels selected. Depending on the dataType of the selected Channels it will check that the expected Metadata is present and if not return an Error. For Channels of dataType "seismic" it will check that the following Metadata information exists: [Station](#), [Channel](#), [Location](#), [Calibration](#), [Response](#), [ChannelInstrument](#). For each of the selected Channels there will be an array of [ChannelInfo](#) objects. The BdsServer will segment the Metadata in time periods at each point an of this Metadata is changed. This each [ChannelInfo](#) object will return a set of Metadata over a time period that has no change in Metadata. When exporting data it can be time segmented to match this.

7.27.3.31 metadataGetFormatted()

```
BError Bds::DataAccess::metadataGetFormatted (
    Selection sel,
```

```
BString format,
BArray< BUInt8 > & data )
```

Return the channel MetaData in a particular format.

Python: (**BError** err, BArray<BUInt8 > data) = metadataGetFormatted(Selection sel, PythonStr format);

Parameters

<i>sel</i>	Channels selection
<i>format</i>	The requested Metadata output format. This should match a format specified by any of the BDS data convertors.
<i>data</i>	The raw file data for the format given in bytes.

This function gathers all of the Metadata for the channels matching the [Selection](#) and then finds a suitable BDS data format conertor to export this Metadata in the chosen format. The resultion array of Byte data can be written to a suitable file.

7.27.3.32 getSelectionInfo()

```
BError Bds::DataAccess::getSelectionInfo (
    SelectionGroup group,
    Selection selectionIn,
    SelectionInfo & selectionInfo )
```

Get information on possible selections. Use in GUI programs to list options available.

Python: (**BError** err, [SelectionInfo](#) selectionInfo) = getSelectionInfo(SelectionGroup group, Selection selectionIn);

Parameters

<i>group</i>	The type of selection info requested. Can be SelectionGroupData to get selections for Sensor data, SelectionGroupMetaData for Metadata selectrions and SelectionGroupDataWithCount as for SelectionGroupData but alos returns the total number of DataChannels.
<i>selectionIn</i>	The current selection
<i>selectionInfo</i>	The returned selection information

Used for selection systems such as that within a Gui to restrict the options available given partial selection parameters. Given a basic [Selection](#) returns all of the [Selection](#) options available based on the [Sensor](#) data or Metadata in the BDS. For example if the selectionIn defines one particular [Network](#) and one particular [Station](#), the get↵SelectionInfo function will return a list of Channels and Sources for that base selection.

7.27.3.33 getSelections()

```
BError Bds::DataAccess::getSelections (
    SelectionGroup group,
    Selection selectionIn,
    Selection & selectionOut )
```

Get selection list.

Python: (**BError** err, [Selection](#) selectionOut) = getSelections(SelectionGroup group, Selection selectionIn);

Parameters

<i>group</i>	The type of selection info requested. Currently ignored.
<i>selectionIn</i>	The selection which may include regular expressions and Station arrays.
<i>selectionOut</i>	The full Sensor data selection after regular expressions have been expanded and any Array's expanded into a list of Stations.

This is used to expand a [Sensor](#) data [Selection](#) to a full [Selection](#) list.

7.27.3.34 dataAvailability()

```
BError Bds::DataAccess::dataAvailability (
    Selection selection,
    BUInt32 num,
    BArray< DataAvailChan > & dataAvailChans )
```

Return availability for data matching the given selection parameters. If num > 0 segment ito this number of fixed time segments.

Python: (**BError** err, **BArray**<[DataAvailChan](#) > dataAvailChans) = dataAvailability([Selection](#) selection, **PythonInt** num);

Parameters

<i>sel</i>	The selection information
<i>num</i>	The number of availability time segments to return.
<i>dataAvailChans</i>	Information on the availability of sensor data for the selected channel

This function will return a list of [Sensor](#) data availability information for the selected Channels. If num > 0 then it will restrict the number of time segments to that number. The time segments will be linearly spaced in time. If num ==0 then the system will return a data availabiliy segment for each [DataChannel](#) continuous segment. This can be slow and return a very large number of segments.

7.27.3.35 dataSearch()

```
BError Bds::DataAccess::dataSearch (
    Selection selection,
    DataInfo & dataInfo )
```

Search for data matching the given selection parameters.

Python: (**BError** err, [DataInfo](#) dataInfo) = dataSearch([Selection](#) selection);

Parameters

<i>selection</i>	Channel Selection information
<i>dataInfo</i>	Sensor data selection information with data segmented by actual data file segments with extra information if available

This will perform a search for [Sensor](#) data matching the [Selection](#). Be careful with the selction as their could be

a lot of data matching that will then hit resource limits. There are restrictions on the maximum time range allowed (MaxTimePeriod) and the maximum [DataChannel](#) segments returned (MaxNumChannels). The return [DataInfo](#) object can be interdated for information on the data, such as sampleRates and/or used in the [dataOpen\(\)](#) function for actual access to the [Sensor](#) data.

7.27.3.36 dataGetChannelInfo()

```
BError Bds::DataAccess::dataGetChannelInfo (
    DataInfo dataInfo,
    ChannelInfos & channelInfos )
```

Return the channel MetaData in structured form.

Python: (**BError** err, [ChannelInfos](#) channelInfos) = dataGetChannelInfo(DataInfo dataInfo);

Parameters

<i>dataInfo</i>	Data selection
<i>channelInfos</i>	Metadata information for the channels

This function returns all of the Metadata for the given Channels specified in the dataInfo selection. It uses the [metadataGetChannelInfo\(\)](#) function converting the [DataInfo](#) selection to a more general [Selection](#) first.

7.27.3.37 dataOpen()

```
BError Bds::DataAccess::dataOpen (
    DataInfo dataInfo,
    BString mode,
    BString format,
    BUInt32 flags,
    DataHandle & dataHandle )
```

Open a data file.

Parameters

in	<i>dataInfo</i>	The sensor data to open selection
in	<i>mode</i>	The open format. The mode should be set to "w" for writing data, "a" when appending data and "r" for reading data.
in	<i>format</i>	What format to open the data stream as. This can be API for raw BDS API access or one of the supported formats that the BDS is able to convert to.
in	<i>flags</i>	A bitset of flags as defined by Bds::DataFlags .
out	<i>dataHandle</i>	The handle for the open data set

Python: (**BError** err, [DataHandle](#) dataHandle) = dataOpen(DataInfo dataInfo, PythonStr mode, PythonStr format, PythonInt flags);

Parameters

<i>dataInfo</i>	Data selection defining the Sensor data Channels
-----------------	--

Parameters

<i>mode</i>	The open mode. This can be "r" for read access, "w" for write access and "aday" to append to a day file.
<i>format</i>	This is the format to use. It can be "API-SM" for multiplexed by sample or "API-CM" for multiplexed by channel. For reads it can also be one of the BDS data convertors supported formats.
<i>flags</i>	Bit set of DataFlags. Can include: DataFlagClipDataToTime, DataFlagClipDataToChannels, DataFlagMergeSegments and DataFlagNoMetadata
<i>dataHandle</i>	The returned data handle to access this data set

This function opens a set of [Sensor](#) data files in the BdsServer for read, write or append access. When the format is "API-SM" or "API-CM" then the raw data blocks can be accessed using the [dataSeekBlock\(\)](#) and [dataGetBlock\(\)](#) functions. **Note** that the "API-SM" format requires the Channels selected to be synchronously sampled and so can only be used for certain import/export data formats where this is the case. The flags are used when reading data. Normally whole contiguous DataBlocks are returned. The actual starttime and endtime of the overall data may thus be before and after the selection. The DataFlagClipDataToTime "clips" the first and last blocks, removing samples, so the time match the actual selection. The DataFlagClipDataToChannels "clips" the blocks so that all of the Channels contain data over the selection period as some Channels otherwise could be missing data for a portion of the [Selection](#). DataFlagMergeSegments merges data segments that have matching end and start times into one continuous segment. DataFlagNoMetadata ignores Metadata when exporting data.

7.27.3.38 dataGetInfo()

```
BError Bds::DataAccess::dataGetInfo (
    DataHandle dataHandle,
    BUInt32 infoExtra,
    DataInfo & dataInfo )
```

Get information on the data file.

Python: (**BError** err, [DataInfo](#) dataInfo) = dataGetInfo(DataHandle dataHandle, PythonInt infoExtra);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>infoExtra</i>	A boolean flag to say to include the extra Metadata contained within the files
<i>dataInfo</i>	The resulting fully filled out DataInfo describing the data.

This functions returns fuller, more detailed, information on the [Sensor](#) data using Metadata stored within the data files themselves. The BDS data format files can store an extensive amount of Metadata from the various import formats supported. A fair amount of this Metadata is freeform and this is for general information purposes only. The returned sampleRate is the samplerate calculated on the actual data samples taking into account start and end times as well as actual numbers of continuous samples. The [DataInfo](#) returned has a set of time segmented information per [Channel](#). Each of the [DataInfo](#) segments defines a cotiguous set of data with no time discontinuities.

7.27.3.39 dataGetNotes()

```
BError Bds::DataAccess::dataGetNotes (
    DataHandle dataHandle,
    BList< Note > & notes )
```

Get notes on the data file.

Python: (**BError** err, BList<Note > notes) = dataGetNotes(DataHandle dataHandle);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>notes</i>	A list of Notes on the data

Returns a list of system and user added notes for the data from the database.

7.27.3.40 dataGetWarnings()

```
BError Bds::DataAccess::dataGetWarnings (
    DataHandle dataHandle,
    BList< BString > & warnings )
```

Get information on the data file.

Python: (**BError** err, BList<BString > warnings) = dataGetWarnings(DataHandle dataHandle);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>warnings</i>	A list of warning strings from the datafiles

Returns a list of warning strings contained within the data files, typically from the data file import convertors.

7.27.3.41 dataSeekBlock()

```
BError Bds::DataAccess::dataSeekBlock (
    DataHandle dataHandle,
    BUInt32 channel,
    BUInt32 segment,
    BTimeStamp time,
    BUInt32 & blockNumber )
```

Searches for a data block matching the time given.

Python: (**BError** err, PythonInt blockNumber) = dataSeekBlock(DataHandle dataHandle, PythonInt channel, PythonInt segment, BTimeStamp time);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>time</i>	The time of a sample to search for
<i>blockNumber</i>	The returned block number

This seeks for the a [DataBlock](#) in the set of files that contains a sample for the time provided. If channel is other than 0 then it seeks for a [DataBlock](#) for the given channel. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return the location of a dataBlock containing all of the Channels. If Segment is 0 then the block number returned is referenced to the start of the set of files. If a particular segment number is given then the seek and the returned block number is within that segment.

7.27.3.42 dataGetBlock()

```
BError Bds::DataAccess::dataGetBlock (
    DataHandle dataHandle,
    BUInt32 channel,
    BUInt32 segment,
    BUInt32 blockNumber,
    DataBlock & data )
```

Return a block of data.

Python: (**BError** err, [DataBlock](#) data) = dataGetBlock(DataHandle dataHandle, PythonInt channel, PythonInt segment, PythonInt blockNumber);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>blockNumber</i>	The block number to fetch
<i>dataBlock</i>	The returned data block

This returns the given block number from the set of opened data files. If channel is other than 0 then it will return the data for the given channel number. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return a [DataBlock](#) with [Sensor](#) data for all of the channels. If Segment is 0 then the block number is referenced to the start of the set of files. If a particular segment number is given then the returned block number is from within that segment.

7.27.3.43 dataClose()

```
DataAccess::dataClose (
    DataHandle dataHandle,
    BError error,
    BInt32 del )
```

Close a file.

Python: (**BError** err) = dataClose(DataHandle dataHandle, BError error, BInt32 del);

7.27.3.44 dataFormattedRead()

```
BError Bds::DataAccess::dataFormattedRead (
    DataHandle dataHandle,
    BUInt32 number,
    BArray< BUInt8 > & data )
```

Read the raw data from the file.

Python: (**BError** err, BArray<BUInt8 > data) = dataFormattedRead(DataHandle dataHandle, PythonInt number);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>number</i>	The maximum number of Bytes to read
<i>data</i>	The set of data bytes in the requested format

Given a handle to a set of data files opened for read in an external data format supported by the BDS data converters, read the raw formatted data bytes from the converted file.

7.27.3.45 dataFormattedGetLength()

```
DataAccess::dataFormattedGetLength (
    DataHandle dataHandle,
    BUInt64 & length )
```

Read the length of the raw formatted data file.

Python: (**BError** err, PythonInt length) = dataFormattedGetLength(DataHandle dataHandle);

7.27.3.46 dataRealtimeConfig()

```
BError Bds::DataAccess::dataRealtimeConfig (
    BInt32 enable,
    Selection sel )
```

Configures the sending of real-time data blocks.

Python: (**BError** err) = dataRealtimeConfig(BInt32 enable, Selection sel);

Parameters

<i>enable</i>	Enable/disable the sending of a real-time data stream
<i>sel</i>	The selection of channels to send

This function has yet to be implemented. It is designed to enable or disable the sending of a stream of data from the BdsServer to a data viewing client.

7.27.3.47 dataRealtimeGet()

```
BError Bds::DataAccess::dataRealtimeGet (
    BUInt32 numBlocks,
    BUInt32 & numBlocksAvailable,
    BList< DataBlockChannel > & dataBlocks )
```

Returns the number of data blocks available and up to numBlocks of these.

Python: (**BError** err, PythonInt numBlocksAvailable, BList<DataBlockChannel > dataBlocks) = dataRealtimeGet(PythonInt numBlocks);

Parameters

<i>numBlocks</i>	How many blocks to return at a time
<i>numBlocksAvailable</i>	The number of blocks available in this stream
<i>dataBlocks</i>	A list of the DataBlocks

This function has yet to be implemented. It is designed to return a set of real-time DataBlocks from the BdsServer to a data viewing client.

7.27.3.48 noteGetList()

```
BError Bds::DataAccess::noteGetList (
    Selection sel,
    BList< Note > & notes )
```

Return a list of Notes.

Python: (**BError** err, BList<Note > notes) = noteGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>notes</i>	The returned Notes

Uses the [Selection](#)'s Network:Station:Channel:Source fields to select a list of Notes.

7.27.3.49 noteUpdate()

```
DataAccess::noteUpdate (
    BInt32 append,
    Note note,
    BUInt32 & id )
```

Add or update a [Note](#).

Python: (**BError** err, PythonInt id) = noteUpdate(BInt32 append, Note note);

7.27.3.50 noteWriteDocument()

```
DataAccess::noteWriteDocument (
    BUInt32 id,
    BString format,
    BArray< BUInt8 > data )
```

Given a [Note](#) write a document associated with it.

Python: (**BError** err) = noteWriteDocument(PythonInt id, PythonStr format, BArray<BUInt8 > data);

7.27.3.51 noteReadDocument()

```
DataAccess::noteReadDocument (
    BUInt32 id,
    BString & format,
    BArray< BUInt8 > & data )
```

Read a document associated with a [Note](#).

Python: (**BError** err, PythonStr format, BArray<BUInt8 > data) = noteReadDocument(PythonInt id);

7.27.3.52 logUpdate()

```
DataAccess::logUpdate (
    BInt32 append,
    Log log,
    BUInt32 & id )
```

Add or update a [Log](#) item.

Python: (**BError** err, PythonInt id) = logUpdate(BInt32 append, Log log);

7.27.3.53 logAppend()

```
BError Bds::DataAccess::logAppend (
    BString type,
    BUInt32 priority,
    BString subSystem,
    BString title,
    BString description )
```

Append a log item.

Python: (**BError** err) = logAppend(PythonStr type, PythonInt priority, PythonStr subSystem, PythonStr title, PythonStr description);

Parameters

<i>type</i>	The log entries type: error, warning, notice, debug or all.
<i>priority</i>	The log events priority from 0 to 5 with 5 being the highest priority
<i>subSystem</i>	The subsystem the log entry is from
<i>title</i>	A title for the log entry
<i>description</i>	A more detailed description of the log entry

This allows BDS xclients, perhaps data import clients, to add BDS system log entries.

7.27.3.54 modeSet()

```
BError Bds::DataAccess::modeSet (
    Mode mode,
    Mode & previousMode )
```

Changes the system mode from Master to slave.

Python: (**BError** err, Mode previousMode) = modeSet(Mode mode);

Parameters

<i>mode</i>	The new server mode
<i>previousMode</i>	The current server mode returned

Changes the mode of a BdsServer's operation. A BdsServer can be in ModeMaster or ModeSlave modes. In ModeSlave it will act as a read only BdsServer which can be used for resilience, data protection or for peerformance purposes.

7.27.3.55 modeSnapshotPause()

```
BError Bds::DataAccess::modeSnapshotPause (
    BInt32 on )
```

Enables/disables backup synchronisation pause.

Python: (**BError** err) = modeSnapshotPause(BInt32 on);

Parameters

<i>on</i>	Enablvle/disable pause
-----------	------------------------

Preforms a snapshot pause. When in this mode all changes to the database and data files will be blocked untill the end of the pause. This allows a backup to be performed with consistant files.

7.27.3.56 clean()

```
BError Bds::DataAccess::clean (
    CleanOptions cleanOptions )
```

Cleans the system logs and Changes information.

Python: (**BError** err) = clean(CleanOptions cleanOptions);

Parameters

<i>cleanOptions</i>	Clean options. This has the options logs, changes and deletedFiles
---------------------	--

This function can be used to clean up old data. It should be called once in a while to perform this work. It will clean the given items that are over a year old.

7.27.3.57 databaseBackup()

```
BError Bds::DataAccess::databaseBackup (
    BString & ref )
```


Backup the database.

Python: (**BError** err, PythonStr ref) = [databaseBackup\(\)](#);

Parameters

<i>ref</i>	Reference and file name component
------------	-----------------------------------

This will perform a complete BDS database backup into a file with the given reference name.

7.27.3.58 statisticsGet()

```
BError Bds::DataAccess::statisticsGet (
    BDict< BString > & info )
```

Get a list of system statistics.

Python: (**BError** err, BDict<BString > info) = [statisticsGet\(\)](#);

Parameters

<i>info</i>	A dictionary of BdsServer statistics
-------------	--------------------------------------

This returns runtime statistics from the BdsServer

7.27.3.59 serverConfigurationGet()

```
BError Bds::DataAccess::serverConfigurationGet (
    BDict< BString > & items )
```

Get server configuration parameters.

Python: (**BError** err, BDict<BString > items) = [serverConfigurationGet\(\)](#);

Parameters

<i>info</i>	A dictionary of BdsServer configuration entries
-------------	---

This returns the list of server configuration parameters.

7.27.3.60 dataFormatGetList()

```
BError Bds::DataAccess::dataFormatGetList (
    BList< DataFormat > & formats )
```

Get list of data formats.

Python: (**BError** err, BList<DataFormat > formats) = [dataFormatGetList\(\)](#);

Parameters

<i>formats</i>	A list of all data file formats
----------------	---------------------------------

This returns the list of all of the data file formats supported by the BdsServer's data convertors.

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

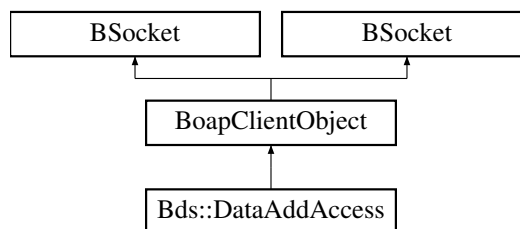
- [BdsC.h](#)
- [BdsC.cc](#)
- [BdsLib.dox](#)
- [/src/blacknest/bds/bds/doc/BdsPython.dox](#)
- [/src/blacknest/bds/bds/doc/bdsApi-extra.dox](#)

7.28 Bds::DataAddAccess Class Reference

This is the DataAdd Access API interface.

```
#include <BdsC.h>
```

Inheritance diagram for Bds::DataAddAccess:



Public Member Functions

- [DataAddAccess](#) (**BString** name="")
BDS RPC access class.
- **BError** [connect](#) (**BString** user, **BString** password)
Provides user/password information.
- **BError** [validateUser](#) (**BString** user, **BString** email)
Checks the user given name or email.
- **BError** [setUser](#) (**BString** user, **BString** email)
Sets user to given name or email.
- **BError** [setUserReal](#) ()
Sets user back to real user.
- **BError** [getVersion](#) (**BString** &version, **BString** &name)
Gets the software version and server name.
- **BError** [userGetFromId](#) (**BUInt32** id, **User** &user)
Get user info given user ID.
- **BError** [userGet](#) (**User** &user)
Get user info of the currently loogged in user.
- **BError** [userSet](#) (**User** user)

- Set user info of the currently loogged in user.*

 - **BError** `userGetGroups` (**BList**< **BString** > &groups)

Get list of groups the user belongs to.
- **BError** `userGetOptions` (**BDict**< **BString** > &items)

Get users options.
- **BError** `userSetOptions` (**BDict**< **BString** > &items)

Set users options.
- **BError** `groupGetList` (**BList**< **Group** > &groups)

Get list of Groups.
- **BError** `networkGetList` (**BList**< **Network** > &networks)

Get list of Networks.
- **BError** `stationGetList` (**Selection** sel, **BList**< **Station** > &stations)

Get list of Stations.
- **BError** `channelGetList` (**Selection** sel, **BList**< **Channel** > &channels)

Get list of Channels.
- **BError** `sourceGetList` (**BList**< **Source** > &sources)

Get list of Sources.
- **BError** `sourcePriorityGetList` (**BList**< **SourcePriority** > &sourcePriorities)

Get list of SourcePriorities.
- **BError** `dataFileGetList` (**Selection** sel, **BList**< **DataFileInfo** > &dataFile)

Get list of DataFiles.
- **BError** `dataChannelGetList` (**Selection** sel, **BList**< **DataChannel** > &dataChannel)

Get list of DataChannels.
- **BError** `channellInstrumentGetList` (**Selection** sel, **BList**< **ChannellInstrument** > &channellInstruments)

Get list of Instruments.
- **BError** `digitiserGetList` (**Selection** sel, **BList**< **Digitiser** > &digitisers)

Get list of Channels.
- **BError** `digitiserGet` (**BUInt32** id, **Digitiser** &digitiser)

*Get a **Digitiser** object given its ID.*
- **BError** `sensorGetList` (**Selection** sel, **BList**< **Sensor** > &sensors)

Get list of Sensors.
- **BError** `sensorGet` (**BUInt32** id, **Sensor** &sensor)

*Get a **Sensor** object given its ID.*
- **BError** `calibrationGetList` (**Selection** sel, **BList**< **Calibration** > &calibrations)

Get list of Calibrations.
- **BError** `responseGetList` (**Selection** sel, **BList**< **Response** > &responses)

Get list of Responses.
- **BError** `locationGetList` (**Selection** sel, **BList**< **Location** > &locations)

*Get list of **Station**, **Channel** or both **Locations** depending on the sel.locationSelect parameter.*
- **BError** `eventGetList` (**Selection** sel, **BList**< **Event** > &events)

Get list of Events.
- **BError** `eventUpdate` (**BInt32** append, **Event** event, **BUInt32** &id)

*Add or update a **Event** entry.*
- **BError** `eventDelete` (**BUInt32** id)

*Delete a **Event** entry.*
- **BError** `specialChannelGetList` (**Selection** sel, **BList**< **SpecialChannel** > &specialChannels)

Get list of Special Channels.
- **BError** `metadataGetChannellInfo` (**Selection** sel, **ChannellInfos** &channellInfos)

Return the channel MetaData in structured form.
- **BError** `metadataGetFormatted` (**Selection** sel, **BString** format, **BArray**< **BUInt8** > & data)

Return the channel MetaData in a particular format.

- **BError** [getSelectionInfo](#) ([SelectionGroup](#) group, [Selection](#) selectionIn, [SelectionInfo](#) &selectionInfo)
Get information on possible selections. Use in GUI programs to list options available.
- **BError** [getSelections](#) ([SelectionGroup](#) group, [Selection](#) selectionIn, [Selection](#) &selectionOut)
Get selection list.
- **BError** [dataAvailability](#) ([Selection](#) selection, **BUInt32** num, **BArray**< [DataAvailChan](#) > &dataAvailChans)
Return availability for data matching the given selection parameters. If num > 0 segment ito this number of fixed time segments.
- **BError** [dataSearch](#) ([Selection](#) selection, [DataInfo](#) &dataInfo)
Search for data matching the given selection parameters.
- **BError** [dataGetChannelInfo](#) ([DataInfo](#) dataInfo, [ChannelInfos](#) &channelInfos)
Return the channel MetaData in structured form.
- **BError** [dataOpen](#) ([DataInfo](#) dataInfo, **BString** mode, **BString** format, **BUInt32** flags, [DataHandle](#) &data↵
Handle)
Open a data file.
- **BError** [dataGetInfo](#) ([DataHandle](#) dataHandle, **BUInt32** infoExtra, [DataInfo](#) &dataInfo)
Get information on the data file.
- **BError** [dataGetNotes](#) ([DataHandle](#) dataHandle, **BList**< [Note](#) > ¬es)
Get notes on the data file.
- **BError** [dataGetWarnings](#) ([DataHandle](#) dataHandle, **BList**< **BString** > &warnings)
Get information on the data file.
- **BError** [dataSeekBlock](#) ([DataHandle](#) dataHandle, **BUInt32** channel, **BUInt32** segment, **BTimeStamp** time, **BUInt32** &blockNumber)
Searches for a data block matching the time given.
- **BError** [dataGetBlock](#) ([DataHandle](#) dataHandle, **BUInt32** channel, **BUInt32** segment, **BUInt32** block↵
Number, [DataBlock](#) & data)
Return a block of data.
- **BError** [dataSetInfo](#) ([DataHandle](#) dataHandle, [DataInfo](#) dataInfo)
Set the info when writing to a file.
- **BError** [dataPutBlock](#) ([DataHandle](#) dataHandle, [DataBlock](#) data)
Send a block of data.
- **BError** [dataClose](#) ([DataHandle](#) dataHandle, **BError** error, **BInt32** del)
Close a file.
- **BError** [dataFormattedRead](#) ([DataHandle](#) dataHandle, **BUInt32** number, **BArray**< **BUInt8** > & data)
Read the raw data from the stream.
- **BError** [dataFormattedGetLength](#) ([DataHandle](#) dataHandle, **BUInt64** & length)
The total length in bytes of the formated data.
- **BError** [dataRealtimeConfig](#) (**BInt32** enable, [Selection](#) sel)
Configures the sending of real-time data blocks.
- **BError** [dataRealtimeGet](#) (**BUInt32** numBlocks, **BUInt32** &numBlocksAvailable, **BList**< [DataBlockChannel](#)
> &dataBlocks)
Returns the number of data blocks available and up to numBlocks of these.
- **BError** [noteGetList](#) ([Selection](#) sel, **BList**< [Note](#) > ¬es)
Return a list of Notes.
- **BError** [noteUpdate](#) (**BInt32** append, [Note](#) note, **BUInt32** &id)
Add or update a [Note](#).
- **BError** [noteWriteDocument](#) (**BUInt32** id, **BString** format, **BArray**< **BUInt8** > data)
Given a [Note](#) write a document associated with it.
- **BError** [noteReadDocument](#) (**BUInt32** id, **BString** & format, **BArray**< **BUInt8** > & data)
Read a document associated with a [Note](#).
- **BError** [logUpdate](#) (**BInt32** append, [Log](#) log, **BUInt32** &id)
*Append a log item ///
Add or update a [Log](#) item.*

- **BError** [logAppend](#) (**BString** type, **BUInt32** priority, **BString** subSystem, **BString** title, **BString** description)
Append a log item.
- **BError** [modeSet](#) (**Mode** mode, **Mode** &previousMode)
Changes the system mode from Master to slave.
- **BError** [modeSnapshotPause](#) (**BInt32** on)
Enables/disables backup synchronisation pause.
- **BError** [clean](#) (**CleanOptions** cleanOptions)
Cleans the system logs and Changes information.
- **BError** [databaseBackup](#) (**BString** &ref)
Backup the database.
- **BError** [statisticsGet](#) (**BDict**< **BString** > &info)
Get a list of system statistics.
- **BError** [serverConfigurationGet](#) (**BDict**< **BString** > &items)
Get server configuration parameters.
- **BError** [dataFormatGetList](#) (**BList**< **DataFormat** > &formats)
Get list of data formats.

Additional Inherited Members

7.28.1 Detailed Description

This is the DataAdd Access API interface.

This object provides the set of API functions that make RPC network calls to a BdsServer. It provides the basic and restricted user orientated read only, data access API along with the ability to import data. It will normally be used by data import client programs.

To use this BdsServer access object you would create one like "DataAccess bds;". Then you would connect it to a particular BdsServer running on a particular host using the `connectService(in String)` function. Once connected you need to login to the BdsServer as a particular user using the `connect(in String user, in String password)`. After your have successfully logged in you can use the various objects methods. These result in a remote procedure call (RPC) to the BdsServer which Will imlement the function and return any data and normally an errors status (**BError**).

7.28.2 Constructor & Destructor Documentation

7.28.2.1 DataAddAccess()

```
Bds::DataAddAccess::DataAddAccess (
    BString name = "" )
```

BDS RPC access class.

Parameters

<i>name</i>	The URL of the remote BOAP object name to connect to.
-------------	---

This object constructor takes the name of the remote object to connect to. However this is normally passed as "" as the **connectService()** function is normally used to perform the actual remote connection.

7.28.3 Member Function Documentation

7.28.3.1 connect()

```
DataAddAccess::connect (
    BString user,
    BString password )
```

Provides user/password information.

Python: (**BError** err) = connect(PythonStr user, PythonStr password);

7.28.3.2 validateUser()

```
BError Bds::DataAddAccess::validateUser (
    BString user,
    BString email )
```

Checks the user given name or email.

Python: (**BError** err) = validateUser(PythonStr user, PythonStr email);

Parameters

<i>user</i>	The user's name
<i>email</i>	The users email address

Checks to see if the user as named exists or if just the email address is provided, if any user with that email address exists.

7.28.3.3 setUser()

```
BError Bds::DataAddAccess::setUser (
    BString user,
    BString email )
```

Sets user to given name or email.

Python: (**BError** err) = setUser(PythonStr user, PythonStr email);

Parameters

<i>user</i>	The user's name
<i>email</i>	The users email address

Changes the connection to be logged in as the given user by their user name or email address. This function will only be allowed if the current user is in the userSet group. The bdsAutodrm user normally has this set so it can change the BdsServer's connection to be that of the BdsAutodrm user for example.

7.28.3.4 setUserReal()

```
DataAddAccess::setUserReal ( )
```

Sets user back to real user.

Python: (**BError** err) = [setUserReal\(\)](#);

7.28.3.5 getVersion()

```
DataAddAccess::getVersion (
    BString & version,
    BString & name )
```

Gets the software version and server name.

Python: (**BError** err, PythonStr version, PythonStr name) = [getVersion\(\)](#);

7.28.3.6 userGetFromId()

```
DataAddAccess::userGetFromId (
    BUInt32 id,
    User & user )
```

Get user info given user ID.

Python: (**BError** err, [User](#) user) = userGetFromId(PythonInt id);

7.28.3.7 userGet()

```
DataAddAccess::userGet (
    User & user )
```

Get user info of the currently loogged in user.

Python: (**BError** err, [User](#) user) = [userGet\(\)](#);

7.28.3.8 userSet()

```
DataAddAccess::userSet (
    User user )
```

Set user info of the currently loogged in user.

Python: (**BError** err) = [userSet\(User user\)](#);

7.28.3.9 userGetGroups()

```
DataAddAccess::userGetGroups (
    BList< BString > & groups )
```

Get list of groups the user belongs to.

Python: (**BError** err, BList<BString > groups) = [userGetGroups\(\)](#);

7.28.3.10 userGetOptions()

```
DataAddAccess::userGetOptions (
    BDict< BString > & items )
```

Get users options.

Python: (**BError** err, BDict<BString > items) = [userGetOptions\(\)](#);

7.28.3.11 userSetOptions()

```
DataAddAccess::userSetOptions (
    BDict< BString > & items )
```

Set users options.

Python: (**BError** err, BDict<BString > items) = [userSetOptions\(\)](#);

7.28.3.12 groupGetList()

```
DataAddAccess::groupGetList (
    BList< Group > & groups )
```

Get list of Groups.

Python: (**BError** err, BList<Group > groups) = [groupGetList\(\)](#);

7.28.3.13 networkGetList()

```
DataAddAccess::networkGetList (
    BList< Network > & networks )
```

Get list of Networks.

Python: (**BError** err, BList<Network > networks) = [networkGetList\(\)](#);

7.28.3.14 stationGetList()

```
BError Bds::DataAddAccess::stationGetList (
    Selection sel,
    BList< Station > & stations )
```

Get list of Stations.

Python: (**BError** err, BList<Station > stations) = [stationGetList\(Selection sel\)](#);

Parameters

<i>sel</i>	The Selection
<i>stations</i>	The list of Stations matching the Selection .

This uses the [Network:Station](#) parts of the [Selection](#) object to return a list of matching Stations.

7.28.3.15 channelGetList()

```
BError Bds::DataAddAccess::channelGetList (
    Selection sel,
    BList< Channel > & channels )
```

Get list of Channels.

Python: (**BError** err, **BList**<Channel > channels) = channelGetList([Selection](#) sel);

Parameters

<i>sel</i>	The Selection
<i>channels</i>	The list of Channels matching the Selection .

If the selection.channelId is set to a value other than 0, this will return a list of one [Channel](#) which matches that id. Otherwise it will return a list of Channels matchin the Network:Station:[Channel](#) regular expression fields of the selection.

7.28.3.16 sourceGetList()

```
DataAddAccess::sourceGetList (
    BList< Source > & sources )
```

Get list of Sources.

Python: (**BError** err, **BList**<Source > sources) = [sourceGetList\(\)](#);

7.28.3.17 sourcePriorityGetList()

```
DataAddAccess::sourcePriorityGetList (
    BList< SourcePriority > & sourcePrioritys )
```

Get list of SourcePriorities.

Python: (**BError** err, **BList**<SourcePriority > sourcePrioritys) = [sourcePriorityGetList\(\)](#);

7.28.3.18 dataFileGetList()

```
BError Bds::DataAddAccess::dataFileGetList (
    Selection sel,
    BList< DataFileInfo > & dataFile )
```

Get list of DataFiles.

Python: (**BError** err, **BList**<DataFileInfo > dataFile) = dataFileGetList([Selection](#) sel);

Parameters

<i>sel</i>	Channels selection
<i>dataFiles</i>	The returned DataFiles

If sel.id is set, then returns the single [DataFile](#) matching this id. Otherwise uses the [Selection](#)'s Network:Station:↔ Channel:Source fields to select a list of DataFiles.

7.28.3.19 dataChannelGetList()

```
BError Bds::DataAddAccess::dataChannelGetList (
    Selection sel,
    BList< DataChannel > & dataChannel )
```

Get list of DataChannels.

Python: (**BError** err, BList<DataChannel > dataChannel) = dataChannelGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>dataChannels</i>	The returned DataChannel

Uses the [Selection](#)'s Network:Station:Channel:Source fields to select a list of DataChannels.

7.28.3.20 channelInstrumentGetList()

```
BError Bds::DataAddAccess::channelInstrumentGetList (
    Selection sel,
    BList< ChannelInstrument > & channelInstruments )
```

Get list of Instruments.

Python: (**BError** err, BList<ChannelInstrument > channelInstruments) = channelInstrumentGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>channelInstruments</i>	The list of ChannelInstruments matching the Selection .

This uses the Network:Station:Channel:Source fields of the [Selection](#) to return a list of matching [Channel](#) Instruments.

7.28.3.21 digitiserGetList()

```
BError Bds::DataAddAccess::digitiserGetList (
    Selection sel,
    BList< Digitiser > & digitisers )
```

Get list of Channels.

Python: (**BError** err, BList<Digitiser > digitisers) = digitiserGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>digitisers</i>	The list of Digitisers matching the Selection .

If selection.digitiserId is set, then return the single matching [Digitiser](#). Otherwise use the [Selection](#)'s Network:↔ Station:Channel:Source fields to return a list of matching Digitisers.

7.28.3.22 digitiserGet()

```
DataAddAccess::digitiserGet (
    BUInt32 id,
    Digitiser & digitiser )
```

Get a [Digitiser](#) object given its ID.

Python: (**BError** err, [Digitiser](#) digitiser) = digitiserGet(PythonInt id);

7.28.3.23 sensorGetList()

```
BError Bds::DataAddAccess::sensorGetList (
    Selection sel,
    BList< Sensor > & sensors )
```

Get list of Sensors.

Python: (**BError** err, BList<Sensor > sensors) = sensorGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>sensors</i>	The list of Sensors matching the Selection .

If selection.sensorId is set, then return the single matching [Sensor](#). Otherwise use the [Selection](#)'s Network:↔ Station:Channel:Source fields to return a list of matching Sensors.

7.28.3.24 sensorGet()

```
DataAddAccess::sensorGet (
    BUInt32 id,
    Sensor & sensor )
```

Get a [Sensor](#) object given its ID.

Python: (**BError** err, [Sensor](#) sensor) = sensorGet(PythonInt id);

7.28.3.25 calibrationGetList()

```
DataAddAccess::calibrationGetList (
    Selection sel,
    BList< Calibration > & calibrations )
```

Get list of Calibrations.

Python: (**BError** err, BList<Calibration > calibrations) = calibrationGetList(Selection sel);

7.28.3.26 responseGetList()

```
BError Bds::DataAddAccess::responseGetList (
    Selection sel,
    BList< Response > & responses )
```

Get list of Responses.

Python: (**BError** err, BList<Response > responses) = responseGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>responses</i>	The list of Responses matching the Selection .

Use the [Selection](#)'s Network:Station:Channel:[Source](#) fields to return a list of matching Responses.

7.28.3.27 locationGetList()

```
BError Bds::DataAddAccess::locationGetList (
    Selection sel,
    BList< Location > & locations )
```

Get list of [Station](#), [Channel](#) or both Locations depending on the sel.locationSelect parameter.

Python: (**BError** err, BList<Location > locations) = locationGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>locations</i>	The list of Locations matching the Selection .

This uses the [Network:Station](#) and optionally the [Channel](#) parts of the [Selection](#) object to return a list of matching Stations. The [Selection.locationSelect](#) parameter can be set to LocationSelectAll, LocationSelectStation or LocationSelectChannel. If set to LocationSelectStation will return the locations of the Stations matched. If set to LocationSelectChannel will return the locations of the Channels matched. If set to LocationSelectAll (The default) will return the locations of the Channels matched if they have Locations or otherwise that of the Stations.

7.28.3.28 eventGetList()

```
BError Bds::DataAddAccess::eventGetList (
    Selection sel,
    BList< Event > & events )
```

Get list of Events.

Python: (**BError** err, BList<Event > events) = eventGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>events</i>	The list of Events matching the Selection .

If sel.id or sel.eventId is set then return the [Event](#) with this ID. Otherwise if sel.name is set return an [Event](#) with a title matching this regular expression between sel.startTime and sel.endTime.. Otherwise match any events between sel.startTime and sel.endTime.

7.28.3.29 eventUpdate()

```
DataAddAccess::eventUpdate (
    BInt32 append,
    Event event,
    BUInt32 & id )
```

Add or update a [Event](#) entry.

Python: (**BError** err, PythonInt id) = eventUpdate(BInt32 append, Event event);

7.28.3.30 eventDelete()

```
DataAddAccess::eventDelete (
    BUInt32 id )
```

Delete a [Event](#) entry.

Python: (**BError** err) = eventDelete(PythonInt id);

7.28.3.31 specialChannelGetList()

```
BError Bds::DataAddAccess::specialChannelGetList (
    Selection sel,
    BList< SpecialChannel > & specialChannels )
```

Get list of Special Channels.

Python: (**BError** err, BList<SpecialChannel > specialChannels) = specialChannelGetList(Selection sel);

Parameters

<i>sel</i>	The Selection
<i>specialChannels</i>	The list of SpecialChannels matching the Selection .

This will return a list of SpecialChannels matchin the Network:Station:[Channel](#) regular expression fields of the selection.

7.28.3.32 metadataGetChannelInfo()

```
BError Bds::DataAddAccess::metadataGetChannelInfo (
    Selection sel,
    ChannelInfos & channelInfos )
```

Return the channel MetaData in structured form.

Python: (**BError** err, [ChannellInfos](#) channellInfos) = metadataGetChannelInfo(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>channellInfos</i>	Metadata information for the channels

This function is designed to help when exporting data from the BDS. It will attempt to gather all of the Metadata present for the Channels selected. Depending on the dataType of the selected Channels it will check that the expected Metadata is present and if not return an Error. For Channels of dataType "seismic" it will check that the following Metadata information exists: [Station](#), [Channel](#), [Location](#). [Calibration](#), [Response](#), [ChannellInstrument](#). For each of the selected Channels there will be an array of [ChannellInfo](#) objects. The BdsServer will segment the Metadata in time periods at each point an of this Metadata is changed. This each [ChannellInfo](#) object will return a set of Metadata over a time period that has no change in Metadata. When exporting data it can be time segmented to match this.

7.28.3.33 metadataGetFormatted()

```
BError Bds::DataAddAccess::metadataGetFormatted (
    Selection sel,
    BString format,
    BArray< BUInt8 > & data )
```

Return the channel MetaData in a particular format.

Python: (**BError** err, **BArray**<**BUInt8** > data) = metadataGetFormatted(Selection sel, PythonStr format);

Parameters

<i>sel</i>	Channels selection
<i>format</i>	The requested Metadata output format. This should match a format specified by any of the BDS data convertors.
<i>data</i>	The raw file data for the format given in bytes.

This function gathers all of the Metadata for the channels matching the [Selection](#) and then finds a suitable BDS data format conertor to export this Metadata in the chosen format. The resultion array of Byte data can be written to a suitable file.

7.28.3.34 getSelectionInfo()

```
BError Bds::DataAddAccess::getSelectionInfo (
    SelectionGroup group,
    Selection selectionIn,
    SelectionInfo & selectionInfo )
```

Get information on possible selections. Use in GUI programs to list options available.

Python: (**BError** err, [SelectionInfo](#) selectionInfo) = getSelectionInfo(SelectionGroup group, Selection selectionIn);

Parameters

<i>group</i>	The type of selection info requested. Can be SelectionGroupData to get selections for Sensor data, SelectionGroupMetaData for Metadata selectrions and SelectionGroupDataWithCount as for SelectionGroupData but alos returns the total number of DataChannels.
<i>selectionIn</i>	The current selection
<i>selectionInfo</i>	The returned selection information

Used for selection systems such as that within a Gui to restrict the options available given partial selection parameters. Given a basic [Selection](#) returns all of the [Selection](#) options available based on the [Sensor](#) data or Metadata in the BDS. For example if the selectionIn defines one particular [Network](#) and one particular [Station](#), the get↵ SelectionInfo function will return a list of Channels and Sources for that base selection.

7.28.3.35 getSelections()

```
BError Bds::DataAddAccess::getSelections (
    SelectionGroup group,
    Selection selectionIn,
    Selection & selectionOut )
```

Get selection list.

Python: (**BError** err, [Selection](#) selectionOut) = getSelections(SelectionGroup group, Selection selectionIn);

Parameters

<i>group</i>	The type of selection info requested. Currently ignored.
<i>selectionIn</i>	The selection which may include regular expressions and Station arrays.
<i>selectionOut</i>	The full Sensor data selection after regular expressions have been expanded and any Array's expanded into a list of Stations.

This is used to expand a [Sensor](#) data [Selection](#) to a full [Selection](#) list.

7.28.3.36 dataAvailability()

```
BError Bds::DataAddAccess::dataAvailability (
```

```

Selection selection,
BUInt32 num,
BArray< DataAvailChan > & dataAvailChans )

```

Return availability for data matching the given selection parameters. If num > 0 segment into this number of fixed time segments.

Python: (**BError** err, BArray<DataAvailChan > dataAvailChans) = dataAvailability(Selection selection, PythonInt num);

Parameters

<i>sel</i>	The selection information
<i>num</i>	The number of availability time segments to return.
<i>dataAvailChans</i>	Information on the availability of sensor data for the selected channel

This function will return a list of [Sensor](#) data availability information for the selected Channels. If num > 0 then it will restrict the number of time segments to that number. The time segments will be linearly spaced in time. If num ==0 then the system will return a data availability segment for each [DataChannel](#) continuous segment. This can be slow and return a very large number of segments.

7.28.3.37 dataSearch()

```

BError Bds::DataAddAccess::dataSearch (
    Selection selection,
    DataInfo & dataInfo )

```

Search for data matching the given selection parameters.

Python: (**BError** err, [DataInfo](#) dataInfo) = dataSearch(Selection selection);

Parameters

<i>selection</i>	Channel Selection information
<i>dataInfo</i>	Sensor data selection information with data segmented by actual data file segments with extra information if available

This will perform a search for [Sensor](#) data matching the [Selection](#). Be careful with the selection as their could be a lot of data matching that will then hit resource limits. There are restrictions on the maximum time range allowed (MaxTimePeriod) and the maximum [DataChannel](#) segments returned (MaxNumChannels). The return [DataInfo](#) object can be interdated for information on the data, such as sampleRates and/or used in the [dataOpen\(\)](#) function for actual access to the [Sensor](#) data.

7.28.3.38 dataGetChannelInfo()

```

BError Bds::DataAddAccess::dataGetChannelInfo (
    DataInfo dataInfo,
    ChannelInfos & channelInfos )

```

Return the channel MetaData in structured form.

Python: (**BError** err, [ChannelInfos](#) channelInfos) = dataGetChannelInfo(DataInfo dataInfo);

Parameters

<i>dataInfo</i>	Data selection
<i>channelInfos</i>	Metadata information for the channels

This function returns all of the Metadata for the given Channels specified in the *dataInfo* selection. It uses the [metadataGetChannelInfo\(\)](#) function converting the [DataInfo](#) selection to a more general [Selection](#) first.

7.28.3.39 dataOpen()

```
BError Bds::DataAddAccess::dataOpen (
    DataInfo dataInfo,
    BString mode,
    BString format,
    BUInt32 flags,
    DataHandle & dataHandle )
```

Open a data file.

Parameters

in	<i>dataInfo</i>	The sensor data to open selection
in	<i>mode</i>	The open format. The mode should be set to "w" for writing data, "a" when appending data and "r" for reading data.
in	<i>format</i>	What format to open the data stream as. This can be API for raw BDS API access or one of the supported formats that the BDS is able to convert to.
in	<i>flags</i>	A bitset of flags as defined by Bds::DataFlags .
out	<i>dataHandle</i>	The handle for the open data set

Python: (**BError** err, [DataHandle](#) dataHandle) = dataOpen(DataInfo dataInfo, PythonStr mode, PythonStr format, PythonInt flags);

Parameters

<i>dataInfo</i>	Data selection defining the Sensor data Channels
<i>mode</i>	The open mode. This can be "r" for read access, "w" for write access and "aday" to append to a day file.
<i>format</i>	This is the format to use. It can be "API-SM" for multiplexed by sample or "API-CM" for multiplexed by channel. For reads it can also be one of the BDS data convertors supported formats.
<i>flags</i>	Bit set of DataFlags. Can include: DataFlagClipDataToTime, DataFlagClipDataToChannels, DataFlagMergeSegments and DataFlagNoMetadata
<i>dataHandle</i>	The returned data handle to access this data set

This function opens a set of [Sensor](#) data files in the BdsServer for read, write or append access. When the format is "API-SM" or "API-CM" then the raw data blocks can be accessed using the [dataSeekBlock\(\)](#) and [dataGetBlock\(\)](#) functions. **Note** that the "API-SM" format requires the Channels selected to be synchronously sampled and so can only be used for certain import/export data formats where this is the case. The flags are used when reading data. Normally whole contiguous DataBlocks are returned. The actual starttime and endtime of the overall data may thus be before and after the selection. The DataFlagClipDataToTime "clips" the first and last blocks, removing samples, so the time match the actual selection. The DataFlagClipDataToChannels "clips" the blocks so that all

of the Channels contain data over the selection period as some Channels otherwise could be missing data for a portion of the [Selection](#). DataFlagMergeSegments merges data segments that have matching end and start times into one continuous segment. DataFlagNoMetadata ignores Metadata when exporting data.

7.28.3.40 dataGetInfo()

```
BError Bds::DataAddAccess::dataGetInfo (
    DataHandle dataHandle,
    BUInt32 infoExtra,
    DataInfo & dataInfo )
```

Get information on the data file.

Python: (**BError** err, [DataInfo](#) dataInfo) = dataGetInfo(DataHandle dataHandle, PythonInt infoExtra);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>infoExtra</i>	A boolean flag to say to include the extra Metadata contained within the files
<i>dataInfo</i>	The resulting fully filled out DataInfo describing the data.

This functions returns fuller, more detailed, information on the [Sensor](#) data using Metadata stored within the data files themselves. The BDS data format files can store an extensive amount of Metadata from the various import formats supported. A fair amount of this Metadata is freeform and this is for general information purposes only. The returned sampleRate is the samplerate calculated on the actual data samples taking into account start and end times as well as actual numbers of continuous samples. The [DataInfo](#) returned has a set of time segmented information per [Channel](#). Each of the [DataInfo](#) segments defines a contiguous set of data with no time discontinuities.

7.28.3.41 dataGetNotes()

```
BError Bds::DataAddAccess::dataGetNotes (
    DataHandle dataHandle,
    BList< Note > & notes )
```

Get notes on the data file.

Python: (**BError** err, BList<Note > notes) = dataGetNotes(DataHandle dataHandle);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>notes</i>	A list of Notes on the data

Returns a list of system and user added notes for the data from the database.

7.28.3.42 dataGetWarnings()

```
BError Bds::DataAddAccess::dataGetWarnings (
    DataHandle dataHandle,
    BList< BString > & warnings )
```

Get information on the data file.

Python: (**BError** err, **BList**<**BString** > warnings) = dataGetWarnings(DataHandle dataHandle);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>warnings</i>	A list of warning strings from the datafiles

Returns a list of warning strings contained within the data files, typically from the data file import convertors.

7.28.3.43 dataSeekBlock()

```
BError Bds::DataAddAccess::dataSeekBlock (
    DataHandle dataHandle,
    BUInt32 channel,
    BUInt32 segment,
    BTimeStamp time,
    BUInt32 & blockNumber )
```

Searches for a data block matching the time given.

Python: (**BError** err, PythonInt blockNumber) = dataSeekBlock(DataHandle dataHandle, PythonInt channel, PythonInt segment, BTimeStamp time);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>time</i>	The time of a sample to search for
<i>blockNumber</i>	The returned block number

This seeks for the a [DataBlock](#) in the set of files that contains a sample for the time provided. If channel is other than 0 then it seeks for a [DataBlock](#) for the given channel. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return the location of a dataBlock containing all of the Channels. If Segment is 0 then the block number returned is referenced to the start of the set of files. If a particular segment number is given then the seek and the returned block number is within that segment.

7.28.3.44 dataGetBlock()

```
BError Bds::DataAddAccess::dataGetBlock (
    DataHandle dataHandle,
    BUInt32 channel,
    BUInt32 segment,
    BUInt32 blockNumber,
    DataBlock & data )
```

Return a block of data.

Python: (**BError** err, [DataBlock](#) data) = dataGetBlock(DataHandle dataHandle, PythonInt channel, PythonInt segment, PythonInt blockNumber);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>blockNumber</i>	The block number to fetch
<i>dataBlock</i>	The returned data block

This returns the given block number from the set of opened data files. If channel is other than 0 then it will return the data for the given channel number. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return a [DataBlock](#) with [Sensor](#) data for all of the channels. If Segment is 0 then the block number is referenced to the start of the set of files. If a particular segment number is given then the returned block number is from within that segment.

7.28.3.45 dataSetInfo()

```
DataAddAccess::dataSetInfo (
    DataHandle dataHandle,
    DataInfo dataInfo )
```

Set the info when writing to a file.

Python: (**BError** err) = [dataSetInfo\(DataHandle dataHandle, DataInfo dataInfo\)](#);

7.28.3.46 dataPutBlock()

```
DataAddAccess::dataPutBlock (
    DataHandle dataHandle,
    DataBlock data )
```

Send a block of data.

Python: (**BError** err) = [dataPutBlock\(DataHandle dataHandle, DataBlock data\)](#);

7.28.3.47 dataClose()

```
DataAddAccess::dataClose (
    DataHandle dataHandle,
    BError error,
    BInt32 del )
```

Close a file.

Python: (**BError** err) = [dataClose\(DataHandle dataHandle, BError error, BInt32 del\)](#);

7.28.3.48 dataFormattedRead()

```
BError Bds::DataAddAccess::dataFormattedRead (
    DataHandle dataHandle,
    BUInt32 number,
    BArray< BUInt8 > & data )
```

Read the raw data from the stream.

Python: (**BError** err, BArray<BUInt8 > data) = [dataFormattedRead\(DataHandle dataHandle, PythonInt number\)](#);

Parameters

<i>dataHandle</i>	The opened data set's handle
<i>number</i>	The maximum number of Bytes to read
<i>data</i>	The set of data bytes in the requested format

Given a handle to a set of data files opened for read in an external data format supported by the BDS data converters, read the raw formatted data bytes from the converted file.

7.28.3.49 dataFormattedGetLength()

```
DataAddAccess::dataFormattedGetLength (
    DataHandle dataHandle,
    BUInt64 & length )
```

The total length in bytes of the formatted data.

Python: (**BError** err, PythonInt length) = dataFormattedGetLength(DataHandle dataHandle);

7.28.3.50 dataRealtimeConfig()

```
BError Bds::DataAddAccess::dataRealtimeConfig (
    BInt32 enable,
    Selection sel )
```

Configures the sending of real-time data blocks.

Python: (**BError** err) = dataRealtimeConfig(BInt32 enable, Selection sel);

Parameters

<i>enable</i>	Enable/disable the sending of a real-time data stream
<i>sel</i>	The selection of channels to send

This function has yet to be implemented. It is designed to enable or disable the sending of a stream of data from the BdsServer to a data viewing client.

7.28.3.51 dataRealtimeGet()

```
BError Bds::DataAddAccess::dataRealtimeGet (
    BUInt32 numBlocks,
    BUInt32 & numBlocksAvailable,
    BList< DataBlockChannel > & dataBlocks )
```

Returns the number of data blocks available and up to numBlocks of these.

Python: (**BError** err, PythonInt numBlocksAvailable, BList<DataBlockChannel > dataBlocks) = dataRealtimeGet(PythonInt numBlocks);

Parameters

<i>numBlocks</i>	How many blocks to return at a time
<i>numBlocksAvailable</i>	The number of blocks available in this stream
<i>dataBlocks</i>	A list of the DataBlocks

This function has yet to be implemented. It is designed to return a set of real-time DataBlocks from the BdsServer to a data viewing client.

7.28.3.52 noteGetList()

```
BError Bds::DataAddAccess::noteGetList (
    Selection sel,
    BList< Note > & notes )
```

Return a list of Notes.

Python: (**BError** err, BList<Note > notes) = noteGetList(Selection sel);

Parameters

<i>sel</i>	Channels selection
<i>notes</i>	The returned Notes

Uses the [Selection](#)'s Network:Station:Channel:Source fields to select a list of Notes.

7.28.3.53 noteUpdate()

```
DataAddAccess::noteUpdate (
    BInt32 append,
    Note note,
    BUInt32 & id )
```

Add or update a [Note](#).

Python: (**BError** err, PythonInt id) = noteUpdate(BInt32 append, Note note);

7.28.3.54 noteWriteDocument()

```
DataAddAccess::noteWriteDocument (
    BUInt32 id,
    BString format,
    BArray< BUInt8 > data )
```

Given a [Note](#) write a document associated with it.

Python: (**BError** err) = noteWriteDocument(PythonInt id, PythonStr format, BArray<BUInt8 > data);

7.28.3.55 noteReadDocument()

```
DataAddAccess::noteReadDocument (
    BUInt32 id,
    BString & format,
    BArray< BUInt8 > & data )
```

Read a document associated with a [Note](#).

Python: (**BError** err, PythonStr format, BArray<BUInt8 > data) = noteReadDocument(PythonInt id);

7.28.3.56 logUpdate()

```
DataAddAccess::logUpdate (
    BInt32 append,
    Log log,
    BUInt32 & id )
```

Append a log item ///< Add or update a [Log](#) item.

Python: (**BError** err, PythonInt id) = logUpdate(BInt32 append, Log log);

7.28.3.57 logAppend()

```
BError Bds::DataAddAccess::logAppend (
    BString type,
    BUInt32 priority,
    BString subSystem,
    BString title,
    BString description )
```

Append a log item.

Python: (**BError** err) = logAppend(PythonStr type, PythonInt priority, PythonStr subSystem, PythonStr title, PythonStr description);

Parameters

<i>type</i>	The log entries type: error, warning, notice, debug or all.
<i>priority</i>	The log events priority from 0 to 5 with 5 being the highest priority
<i>subSystem</i>	The subsystem the log entry is from
<i>title</i>	A title for the log entry
<i>description</i>	A more detailed description of the log entry

This allows BDS xclients, perhaps data import clients, to add BDS system log entries.

7.28.3.58 modeSet()

```
BError Bds::DataAddAccess::modeSet (
    Mode mode,
    Mode & previousMode )
```

Changes the system mode from Master to slave.

Python: (**BError** err, Mode previousMode) = modeSet(Mode mode);

Parameters

<i>mode</i>	The new server mode
<i>previousMode</i>	The current server mode returned

Changes the mode of a BdsServer's operation. A BdsServer can be in ModeMaster or ModeSlave modes. In ModeSlave it will act as a read only BdsServer which can be used for resilience, data protection or for peerformance purposes.

7.28.3.59 modeSnapshotPause()

```
BError Bds::DataAddAccess::modeSnapshotPause (
    BInt32 on )
```

Enables/disables backup synchronisation pause.

Python: (**BError** err) = modeSnapshotPause(BInt32 on);

Parameters

<i>on</i>	Enabvle/disable pause
-----------	-----------------------

Preforms a snapshot pause. When in this mode all changes to the database and data files will be blocked untill the end of the pause. This allows a backup to be performed with consistant files.

7.28.3.60 clean()

```
BError Bds::DataAddAccess::clean (
    CleanOptions cleanOptions )
```

Cleans the system logs and Changes information.

Python: (**BError** err) = clean(CleanOptions cleanOptions);

Parameters

<i>cleanOptions</i>	Clean options. This has the options logs, changes and deletedFiles
---------------------	--

This function can be used to clean up old data. It should be called once in a while to perform this work. It will clean the given items that are over a year old.

7.28.3.61 databaseBackup()

```
BError Bds::DataAddAccess::databaseBackup (
    BString & ref )
```


Backup the database.

Python: (**BError** err, PythonStr ref) = [databaseBackup\(\)](#);

Parameters

<i>ref</i>	Reference and file name component
------------	-----------------------------------

This will perform a complete BDS database backup into a file with the given reference name.

7.28.3.62 statisticsGet()

```
BError Bds::DataAddAccess::statisticsGet (
    BDict< BString > & info )
```

Get a list of system statistics.

Python: (**BError** err, BDict<BString > info) = [statisticsGet\(\)](#);

Parameters

<i>info</i>	A dictionary of BdsServer statistics
-------------	--------------------------------------

This returns runtime statistics from the BdsServer

7.28.3.63 serverConfigurationGet()

```
BError Bds::DataAddAccess::serverConfigurationGet (
    BDict< BString > & items )
```

Get server configuration parameters.

Python: (**BError** err, BDict<BString > items) = [serverConfigurationGet\(\)](#);

Parameters

<i>info</i>	A dictionary of BdsServer configuration entries
-------------	---

This returns the list of server configuration parameters.

7.28.3.64 dataFormatGetList()

```
BError Bds::DataAddAccess::dataFormatGetList (
    BList< DataFormat > & formats )
```

Get list of data formats.

Python: (**BError** err, BList<DataFormat > formats) = [dataFormatGetList\(\)](#);

Parameters

<i>formats</i>	A list of all data file formats
----------------	---------------------------------

This returns the list of all of the data file formats supported by the BdsServer's data convertors.

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

- [BdsC.h](#)
- [BdsC.cc](#)
- [BdsLib.dox](#)
- [/src/blacknest/bds/bds/doc/BdsPython.dox](#)
- [/src/blacknest/bds/bds/doc/bdsApi-extra.dox](#)

7.29 Bds::DataAvail Class Reference

This class provides availability information on a particular period of data.

```
#include <BdsD.h>
```

Public Member Functions

- [DataAvail](#) ([BTimeStamp](#) *startTime*= [BTimeStamp](#)(), [BTimeStamp](#) *endTime*= [BTimeStamp](#)(), [AvailType](#) *availType*=[AvailType](#)())

Public Attributes

- [BTimeStamp](#) *startTime*
The Start Time.
- [BTimeStamp](#) *endTime*
The End Time.
- [AvailType](#) *availType*
The availability type. Can be: AvailNone, AvailPartial or AvailFull.

7.29.1 Detailed Description

This class provides availability information on a particular period of data.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 DataAvail()

```
Bds::DataAvail::DataAvail (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    AvailType availType = AvailType() )
```

7.29.3 Member Data Documentation

7.29.3.1 startTime

```
BTimeStamp Bds::DataAvail::startTime
```

The Start Time.

7.29.3.2 endTime

```
BTimeStamp Bds::DataAvail::endTime
```

The End Time.

7.29.3.3 availType

```
AvailType Bds::DataAvail::availType
```

The availability type. Can be: AvailNone, AvailPartial or AvailFull.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.30 Bds::DataAvailChan Class Reference

This class defines availability information on a set of data.

```
#include <BdsD.h>
```

Public Member Functions

- **DataAvailChan** (**BTimeStamp** startTime= **BTimeStamp**(), **BTimeStamp** endTime= **BTimeStamp**(), **BString** network= **BString**(), **BString** station= **BString**(), **BString** channel= **BString**(), **BString** source= **BString**(), **BArray**< **DataAvail** > segments= **BArray**< **DataAvail** >())

Public Attributes

- **BTimeStamp** startTime
The Start Time.
- **BTimeStamp** endTime
The End Time.
- **BString** network
The Network Name.
- **BString** station
The Station name.
- **BString** channel
The Channels name.
- **BString** source
The Data Source.
- **BArray**< **DataAvail** > segments
Data time segment availability info.

7.30.1 Detailed Description

This class defines availability information on a set of data.

Availability information is provided for a particular channel network:station:channel:source. For a particular time period this will be broken down into time period segments. The time granularity of the segments is dependent of the type of data availability search and options in the requests. Normally the granularity is that defined by the [DataChannel](#) information in the database. The actual seismic data files may not contains some data over the given periods due to missing blocks etc. An in-depth data availability search could provide more fuller data availability information but with much more data processing, this has yet to be implemented.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 DataAvailChan()

```
Bds::DataAvailChan::DataAvailChan (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString(),
    BArray< DataAvail > segments = BArray<DataAvail >() )
```

7.30.3 Member Data Documentation

7.30.3.1 startTime

BTimeStamp Bds::DataAvailChan::startTime

The Start Time.

7.30.3.2 endTime

BTimeStamp Bds::DataAvailChan::endTime

The End Time.

7.30.3.3 network

BString Bds::DataAvailChan::network

The [Network](#) Name.

7.30.3.4 station

BString Bds::DataAvailChan::station

The [Station](#) name.

7.30.3.5 channel

BString Bds::DataAvailChan::channel

The Channels name.

7.30.3.6 source

BString Bds::DataAvailChan::source

The Data [Source](#).

7.30.3.7 segments

BArray<DataAvail > Bds::DataAvailChan::segments

Data time segment availability info.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.31 Bds::DataBlock Class Reference

This class provides the actual [Sensor](#) data values contained within a single data block.

```
#include <BdsD.h>
```

Public Member Functions

- **DataBlock** (**BTimeStamp** startTime= **BTimeStamp**(), **BTimeStamp** endTime= **BTimeStamp**(), **BUInt32** channelNumber=0, **BUInt32** segmentNumber=0, **BArray< BArray< BFloat64 > >** channelData= **BArray< BArray< BFloat64 > >**(), **BDict< BString >** info= **BDict< BString >**())

Public Attributes

- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time the channel was available.
- **BUInt32** [channelNumber](#)
The first channel number. (1, 2, 3 ...)
- **BUInt32** [segmentNumber](#)
The segment number. (1, 2, 3, ...)
- **BArray< BArray< BFloat64 > >** [channelData](#)
The raw channel data in a 2 dimensional array, ordered as per channel information in dataInfo.
- **BDict< BString >** [info](#)
Extra information on data or ASCII data.

7.31.1 Detailed Description

This class provides the actual [Sensor](#) data values contained within a single data block.

All [Sensor](#) data is segmented into blocks that have a start and end timestamp and perhaps other metadata. The [DataBlock](#) contains an array of actual data samples in a 64bit floating pointer format for one or more channels. If the samples are for multiple channels it is assumed that these are synchronously sampled and are normally from a data file that is in the sample multiplexed format (ie. For each point in time there is a set of samples one for each channel). The startTime comes from the original blocks start time information. The endTime may come from the original blocks end time information, if available in the original data format that the data was imported from or is generated from the startTime and the calculated sample rate of the data. For some data types where the sampling rate is a bit variable, the endTime fields may be lined up with the next blocks startTime field to ensure contiguous data segments when exporting data. The info field contains extra, free string format, metadata on the block if available. This could be quality information from the TapeDigitiser system for example.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 DataBlock()

```
Bds::DataBlock::DataBlock (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BUInt32 channelNumber = 0,
    BUInt32 segmentNumber = 0,
    BArray< BArray< BFloat64 > > channelData = BArray< BArray< BFloat64 > >(),
    BDict< BString > info = BDict< BString >() )
```

7.31.3 Member Data Documentation

7.31.3.1 startTime

```
BTimeStamp Bds::DataBlock::startTime
```

The Start Time.

7.31.3.2 endTime

```
BTimeStamp Bds::DataBlock::endTime
```

The End Time the channel was available.

7.31.3.3 channelNumber

```
BUInt32 Bds::DataBlock::channelNumber
```

The first channel number. (1, 2, 3 ...)

7.31.3.4 segmentNumber

```
BUInt32 Bds::DataBlock::segmentNumber
```

The segment number. (1, 2, 3, ...)

7.31.3.5 channelData

```
BArray< BArray< BFloat64 > > Bds::DataBlock::channelData
```

The raw channel data in a 2 dimensional array, ordered as per channel information in dataInfo.

7.31.3.6 info

```
BDict< BString > Bds::DataBlock::info
```

Extra information on data or ASCII data.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.32 Bds::DataBlockChannel Class Reference

This class provides the actual seismic data values contained within a single data block along with the network↵:station:channel:source information.

```
#include <BdsD.h>
```

Public Member Functions

- [DataBlockChannel](#) ([BString](#) network= [BString](#)(), [BString](#) station= [BString](#)(), [BString](#) channel= [BString](#)(), [BString](#) source= [BString](#)())

Public Attributes

- **BString** [network](#)
The [Network](#) Name or names.
- **BString** [station](#)
The [Station](#) name or Stations.
- **BString** [channel](#)
The Channels name or Channels.
- **BString** [source](#)
The Data [Source](#) or Sources.

7.32.1 Detailed Description

This class provides the actual seismic data values contained within a single data block along with the network↵:station:channel:source information.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 DataBlockChannel()

```
Bds::DataBlockChannel::DataBlockChannel (
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString() )
```

7.32.3 Member Data Documentation

7.32.3.1 network

```
BString Bds::DataBlockChannel::network
```

The [Network](#) Name or names.

7.32.3.2 station

```
BString Bds::DataBlockChannel::station
```

The [Station](#) name or Stations.

7.32.3.3 channel

BString Bds::DataBlockChannel::channel

The Channels name or Channels.

7.32.3.4 source

BString Bds::DataBlockChannel::source

The Data [Source](#) or Sources.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.33 Bds::DataBlockPos Class Reference

This defines the position of a data block in a file. It is used by the BDS data converters to order blocks by time.

```
#include <BdsDataFile.h>
```

Public Member Functions

- [DataBlockPos](#) (**BTimeStamp** startTime=0, **BTimeStamp** endTime=0, **BUInt64** position=0, **BUInt** order=0, int ref=0, **BUInt** numSamples=0)
- int [operator<](#) (const [DataBlockPos](#) &b) const

Public Attributes

- **BTimeStamp** [startTime](#)
- **BTimeStamp** [endTime](#)
- **BUInt64** [position](#)
- **BUInt** [order](#)
- int [ref](#)
- **BUInt** [numSamples](#)

7.33.1 Detailed Description

This defines the position of a data block in a file. It is used by the BDS data converters to order blocks by time.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 DataBlockPos()

```
Bds::DataBlockPos::DataBlockPos (
    BTimeStamp startTime = 0,
    BTimeStamp endTime = 0,
    BUInt64 position = 0,
    BUInt order = 0,
    int ref = 0,
    BUInt numSamples = 0 ) [inline]
```

7.33.3 Member Function Documentation

7.33.3.1 operator<()

```
int Bds::DataBlockPos::operator< (
    const DataBlockPos & b ) const [inline]
```

7.33.4 Member Data Documentation

7.33.4.1 startTime

BTimeStamp Bds::DataBlockPos::startTime

7.33.4.2 endTime

BTimeStamp Bds::DataBlockPos::endTime

7.33.4.3 position

BUInt64 Bds::DataBlockPos::position

7.33.4.4 order

BUInt Bds::DataBlockPos::order

7.33.4.5 ref

```
int Bds::DataBlockPos::ref
```

7.33.4.6 numSamples

```
BUInt Bds::DataBlockPos::numSamples
```

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

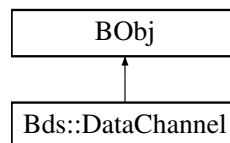
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h](#)

7.34 Bds::DataChannel Class Reference

This class defines information on a single channel's set of data stored in a file.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::DataChannel:



Public Member Functions

- [DataChannel](#) ([BUInt32 id](#)=0, [BTimeStamp startTime](#)= [BTimeStamp\(\)](#), [BTimeStamp endTime](#)= [BTimeStamp\(\)](#), [BString network](#)= [BString\(\)](#), [BString station](#)= [BString\(\)](#), [BString channel](#)= [BString\(\)](#), [BString source](#)= [BString\(\)](#), [BUInt32 numBlocks](#)=0, [BUInt64 numSamples](#)=0, [BFloat64 sampleRate](#)=0, [BUInt32 sampleFormat](#)=0, [BUInt32 dataFileId](#)=0, [BUInt32 dataFileChannel](#)=0, [BString importFormat](#)= [BString\(\)](#), [BString importFilename](#)= [BString\(\)](#), [BTimeStamp importStartTime](#)= [BTimeStamp\(\)](#), [BDict< BString > info](#)= [BDict< BString >\(\)](#))
- [BString getType](#) ()
- [BError setMembers](#) ([BDictString &members](#))
- [BError setMember](#) ([BString name](#), [BString value](#))
- [BError getMembers](#) ([BDictString &members](#))
- [BError getMember](#) ([BString name](#), [BString &value](#))

Public Attributes

- **BUInt32** [id](#)
Unique ID when stored in a database or for other uses.
- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time.
- **BString** [network](#)
The [Network](#) Name.
- **BString** [station](#)
The [Station](#) name.
- **BString** [channel](#)
The Channels name.
- **BString** [source](#)
The Data [Source](#).
- **BUInt32** [numBlocks](#)
The total number of blocks per channel if known, 0 otherwise.
- **BUInt64** [numSamples](#)
The total number of samples per channel if known, 0 otherwise.
- **BFloat64** [sampleRate](#)
The data's sample rate.
- **BUInt32** [sampleFormat](#)
The data sample format.
- **BUInt32** [dataFileId](#)
The Data File Id. This links to the particular [DataFileInfo](#) where the data is stored.
- **BUInt32** [dataFileChannel](#)
The Data File [Channel](#) number. The channel number within the data file. (1, 2, 3 ...)
- **BString** [importFormat](#)
The original data format.
- **BString** [importFilename](#)
The original data file name.
- **BTimeStamp** [importStartTime](#)
The original import files start time.
- **BDict**< **BString** > [info](#)
Extra info on the channel.

7.34.1 Detailed Description

This class defines information on a single channel's set of data stored in a file.

This provides information on actual [Sensor](#) data for a channel that is stored in the BDS system. The data will be stored in a particular file perhaps with other data channels ([DataFileInfo](#)). When known, information on the channels numBlocks, numSamples and sampleRate will be provided. Generally this information will only be known if a data file has been imported rather than a live real-time data stream. Generally the [Sensor](#) data file itself should be interrogated to find the definitive information. The info field provides extra details on the data contents which might come from one of the specific data import formats.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 DataChannel()

```
Bds::DataChannel::DataChannel (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString(),
    BUInt32 numBlocks = 0,
    BUInt64 numSamples = 0,
    BFloat64 sampleRate = 0,
    BUInt32 sampleFormat = 0,
    BUInt32 dataFileId = 0,
    BUInt32 dataFileChannel = 0,
    BString importFormat = BString(),
    BString importFilename = BString(),
    BTimeStamp importStartTime = BTimeStamp(),
    BDict< BString > info = BDict< BString >() )
```

7.34.3 Member Function Documentation

7.34.3.1 getType()

```
BString Bds::DataChannel::getType ( )
```

7.34.3.2 setMembers()

```
BError Bds::DataChannel::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.34.3.3 setMember()

```
BError Bds::DataChannel::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.34.3.4 getMembers()

```
BError Bds::DataChannel::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.34.3.5 getMember()

```
BError Bds::DataChannel::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.34.4 Member Data Documentation

7.34.4.1 id

```
BUInt32 Bds::DataChannel::id
```

Unique ID when stored in a database or for other uses.

7.34.4.2 startTime

```
BTimeStamp Bds::DataChannel::startTime
```

The Start Time.

7.34.4.3 endTime

```
BTimeStamp Bds::DataChannel::endTime
```

The End Time.

7.34.4.4 network

BString Bds::DataChannel::network

The [Network](#) Name.

7.34.4.5 station

BString Bds::DataChannel::station

The [Station](#) name.

7.34.4.6 channel

BString Bds::DataChannel::channel

The Channels name.

7.34.4.7 source

BString Bds::DataChannel::source

The Data [Source](#).

7.34.4.8 numBlocks

BUInt32 Bds::DataChannel::numBlocks

The total number of blocks per channel if known, 0 otherwise.

7.34.4.9 numSamples

BUInt64 Bds::DataChannel::numSamples

The total number of samples per channel if known, 0 otherwise.

7.34.4.10 sampleRate

BFloat64 Bds::DataChannel::sampleRate

The data's sample rate.

7.34.4.11 sampleFormat

BUInt32 Bds::DataChannel::sampleFormat

The data sample format.

7.34.4.12 dataFileId

BUInt32 Bds::DataChannel::dataFileId

The Data File Id. This links to the particular [DataFileInfo](#) where the data is stored.

7.34.4.13 dataFileChannel

BUInt32 Bds::DataChannel::dataFileChannel

The Data File [Channel](#) number. The channel number within the data file. (1, 2, 3 ...)

7.34.4.14 importFormat

BString Bds::DataChannel::importFormat

The original data format.

7.34.4.15 importFilename

BString Bds::DataChannel::importFilename

The original data file name.

7.34.4.16 importStartTime

BTimeStamp Bds::DataChannel::importStartTime

The original import files start time.

7.34.4.17 info

BDict< BString > Bds::DataChannel::info

Extra info on the channel.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.35 Bds::DataCollate Class Reference

Not sure if this is used or what it does.

```
#include <BdsDataCollate.h>
```

Public Member Functions

- [DataCollate](#) ()
- [~DataCollate](#) ()
- **BError** [addSource](#) ([DataFile](#) &dataFile, **BUInt** channel)
- **BError** [readData](#) (**BUInt32** blockNumber, [DataBlock](#) & data)

7.35.1 Detailed Description

Not sure if this is used or what it does.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 DataCollate()

```
Bds::DataCollate::DataCollate ( )
```

7.35.2.2 ~DataCollate()

```
Bds::DataCollate::~~DataCollate ( )
```

7.35.3 Member Function Documentation

7.35.3.1 addSource()

```
BError Bds::DataCollate::addSource (
    DataFile & dataFile,
    BUInt channel )
```

7.35.3.2 readData()

```
BError Bds::DataCollate::readData (
    BUInt32 blockNumber,
    DataBlock & data )
```

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.cpp](#)

7.36 Bds::DataError Class Reference

This stores a data error. It includes an error number and a string as well as information on what seismic channel it is for.

```
#include <BdsLib.h>
```

Public Member Functions

- [DataError](#) ()
- [DataError](#) (int errorNumber, **BString** title, **BString** filename, **BTimeStamp** startTime, **BTimeStamp** endTime, [DataInfo](#) &dataInfo, **BUInt** channel, **BString** description, **BString** user="")
- [DataError](#) & [set](#) (int errorNumber, **BString** title, **BString** importFilename, **BTimeStamp** startTime, **BTimeStamp** endTime, [DataInfo](#) &dataInfo, **BUInt** channel, **BString** description, **BString** user="")
- void [mergeDataInfo](#) (const [DataInfo](#) &dataInfo, **BUInt** channel)
- int [getErrorNumber](#) () const
Get The error number.
- **BString** [getTitle](#) () const
Get the title.
- **BError** [setString](#) (**BString** str)
Set from string.
- **BError** [setStringUser](#) (**BString** str, **BString** user)
Set from string given by user on command line.
- **BString** [getString](#) () const
Get error message.
- int [num](#) () const
Get The error number.
- const char * [str](#) () const
Return a char string.*
- [operator int](#) () const
Return error number.

Public Attributes

- **Blnt32** [oerrorNumber](#)
Error number.
- **BString** [otitle](#)
The title.
- **BString** [odescription](#)
The description.
- **BString** [ofilename](#)
The import filename;.
- **BTimeStamp** [ostartTime](#)
The start Time.
- **BTimeStamp** [oendTime](#)
The end Time.
- **BString** [onetwork](#)
The network Name.
- **BString** [ostation](#)
The station/array name.
- **BString** [ochannel](#)
The channel name.
- **BString** [osource](#)
The data [Source](#).
- **BString** [ouser](#)
The user.

7.36.1 Detailed Description

This stores a data error. It includes an error number and a string as well as information on what seismic channel it is for.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 DataError() [1/2]

```
Bds::DataError::DataError ( )
```

7.36.2.2 DataError() [2/2]

```
Bds::DataError::DataError (
    int errorNumber,
    BString title,
    BString filename,
    BTimeStamp startTime,
    BTimeStamp endTime,
    DataInfo & dataInfo,
    BUInt channel,
    BString description,
    BString user = "" )
```

7.36.3 Member Function Documentation

7.36.3.1 set()

```
DataError & Bds::DataError::set (
    int errorNumber,
    BString title,
    BString importFilename,
    BTimeStamp startTime,
    BTimeStamp endTime,
    DataInfo & dataInfo,
    BUInt channel,
    BString description,
    BString user = "" )
```

7.36.3.2 mergeDataInfo()

```
void Bds::DataError::mergeDataInfo (
    const DataInfo & dataInfo,
    BUInt channel )
```

7.36.3.3 getErrorNumber()

```
int Bds::DataError::getErrorNumber ( ) const
```

Get The error number.

7.36.3.4 getTitle()

```
BString Bds::DataError::getTitle ( ) const
```

Get the title.

7.36.3.5 setString()

```
BError Bds::DataError::setString (
    BString str )
```

Set from string.

7.36.3.6 setStringUser()

```
BError Bds::DataError::setStringUser (
    BString str,
    BString user )
```

Set from string given by user on command line.

7.36.3.7 getString()

```
BString Bds::DataError::getString ( ) const
```

Get error message.

7.36.3.8 num()

```
int Bds::DataError::num ( ) const
```

Get The error number.

7.36.3.9 str()

```
const char * Bds::DataError::str ( ) const
```

Return a char* string.

7.36.3.10 operator int()

```
Bds::DataError::operator int ( ) const
```

Return error number.

7.36.4 Member Data Documentation

7.36.4.1 oerrorNumber

```
BInt32 Bds::DataError::oerrorNumber
```

Error number.

7.36.4.2 otitle

```
BString Bds::DataError::otitle
```

The title.

7.36.4.3 odescription

```
BString Bds::DataError::odescription
```

The description.

7.36.4.4 ofilename

```
BString Bds::DataError::ofilename
```

The import filename;.

7.36.4.5 ostartTime

```
BTimeStamp Bds::DataError::ostartTime
```

The start Time.

7.36.4.6 oendTime

```
BTimeStamp Bds::DataError::oendTime
```

The end Time.

7.36.4.7 onetwork

```
BString Bds::DataError::onetwork
```

The network Name.

7.36.4.8 ostation

```
BString Bds::DataError::ostation
```

The station/array name.

7.36.4.9 ochannel

```
BString Bds::DataError::ochannel
```

The channel name.

7.36.4.10 osource

BString Bds::DataError::osource

The data [Source](#).

7.36.4.11 ouser

BString Bds::DataError::ouser

The user.

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

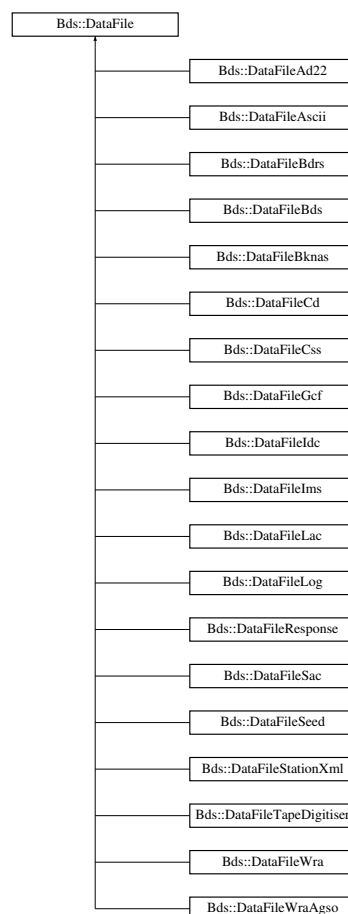
- [BdsLib.h](#)
- [BdsLib.cpp](#)

7.37 Bds::DataFile Class Reference

This class defines the interface for generic data file access that all of the BDS data converters share.

```
#include <BdsDataFile.h>
```

Inheritance diagram for Bds::DataFile:



Public Types

- enum [DataOrder](#) { [DataOrderUnknown](#) , [DataOrderAll](#) , [DataOrderSample](#) , [DataOrderChannel](#) }
- enum [Features](#) { [FeatureNone](#) = 0x00 , [FeatureCanWrite](#) = 0x01 , [FeatureCanRead](#) = 0x02 }
- enum [WriteOptionsList](#) { [WriteOptionNone](#) = 0x00 , [WriteOptionSensorData](#) = 0x01 , [WriteOptionNoMetadata](#) = 0x02 }
- enum [ReadOptionsList](#) {
[ReadOptionNone](#) = 0x00 , [ReadOptionValidate](#) = 0x01 , [ReadOptionFileNameProcess](#) = 0x02 ,
[ReadOptionFixCorruptions](#) = 0x04 ,
[ReadOptionReorder](#) = 0x08 , [ReadOptionDeleteDuplicates](#) = 0x10 , [ReadOptionInfoExtra](#) = 0x20 ,
[ReadOptionIgnoreSamplerate](#) = 0x40 ,
[ReadOptionPrintBlocks](#) = 0x80 , [ReadOptionFixSampleRate](#) = 0x100 }

Public Member Functions

- [DataFile](#) ()
- virtual [~DataFile](#) ()
- virtual void [init](#) ()
Initialise.
- virtual **BError** [open](#) (**BString** fileName, **BString** mode)
Open the file for read or write.
- virtual **BError** [close](#) ()
Close the file.
- virtual **BError** [setFormat](#) (**BString** format)
Set the sub-format.
- virtual **BString** [getFileName](#) ()
Return the file name.
- virtual [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- virtual int [getFeatures](#) ()
Get bitmask of supported features.
- virtual **BString** [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- virtual **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannelInfos](#) &channelInfos, [WriteOptionsList](#) options=[WriteOptionNone](#))
Set information on data for write.
- virtual **BError** [start](#) (**BUInt** channel, **BUInt** segment)
Start writing next segment of data.
- virtual **BError** [writeData](#) (const [DataBlock](#) & data)
Write a block of data.
- virtual **BError** [end](#) ()
End write segment.
- virtual **BError** [flush](#) ()
Flush data to disk.
- virtual **BError** [fileNameProcess](#) ()
Parse the file name for a date/time.
- virtual **BError** [getFormat](#) (**BString** & format)
Get sub-format.
- virtual **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- virtual **BError** [seekBlock](#) (**BUInt32** channel, **BUInt** segment, **BTimeStamp** time, **BUInt32** &blockNumber, **BUInt64** &sampleNumber, [DataBlock](#) & data)

- Find requested block on given channel given a time.*
- virtual **Error** [readData](#) (**UInt32** channel, **UInt** segment, **UInt32** blockNumber, [DataBlock](#) &dataBlock)
- Read a block.*
- virtual **Error** [getMetaData](#) ([ChannelInfos](#) &channelInfos, **UInt32** options, **List**< [DataError](#) > &errors)
- Return all known MetaData in the file.*
- void [dataErrorFixup](#) (const [DataInfo](#) &dataInfo, **List**< [DataError](#) > &errors)
- Fixup data errors, mainly start/end times to be within data.*
- **Int64** [timeCompare](#) (**TimeStamp** t1, **TimeStamp** t2, **UInt** diff)
- Compare timestamps with a margin.*
- int [duplicateCheck](#) (const [DataBlock](#) &data1, const [DataBlock](#) &data2, **UInt** channel=0)
- Check if blocks are duplicates.*
- **UInt64** [getFilePosition](#) ()

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()
- Get list of supported formats.*

Protected Attributes

- **String** [ofilename](#)
- **String** [omode](#)
- **TimeStamp** [ofilenameTime](#)
- **File** [ofile](#)
- **String** [oformat](#)

7.37.1 Detailed Description

This class defines the interface for generic data file access that all of the BDS data converters share.

7.37.2 Member Enumeration Documentation

7.37.2.1 DataOrder

```
enum Bds::DataFile::DataOrder
```

Enumerator

DataOrderUnknown	
DataOrderAll	
DataOrderSample	
DataOrderChannel	

7.37.2.2 Features

```
enum Bds::DataFile::Features
```

Enumerator

FeatureNone	
FeatureCanWrite	
FeatureCanRead	

7.37.2.3 WriteOptionsList

```
enum Bds::DataFile::WriteOptionsList
```

Enumerator

WriteOptionNone	
WriteOptionSensorData	
WriteOptionNoMetadata	

7.37.2.4 ReadOptionsList

```
enum Bds::DataFile::ReadOptionsList
```

Enumerator

ReadOptionNone	
ReadOptionValidate	
ReadOptionFileNameProcess	
ReadOptionFixCorruptions	
ReadOptionReorder	
ReadOptionDeleteDuplicates	
ReadOptionInfoExtra	
ReadOptionIgnoreSamplerate	
ReadOptionPrintBlocks	
ReadOptionFixSampleRate	

7.37.3 Constructor & Destructor Documentation

7.37.3.1 DataFile()

```
Bds::DataFile::DataFile ( )
```

7.37.3.2 ~DataFile()

```
Bds::DataFile::~~DataFile ( ) [virtual]
```

7.37.4 Member Function Documentation

7.37.4.1 init()

```
void Bds::DataFile::init ( ) [virtual]
```

Initialise.

7.37.4.2 open()

```
BError Bds::DataFile::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for read or write.

Parameters

<i>fileName</i>	The filename to open. Can be a full path or relative path to the current working directory
<i>mode</i>	The open mode. Can be "r" or "w".

Opens a file of the gievn file format

Reimplemented in [Bds::DataFileAscii](#), [Bds::DataFileBds](#), [Bds::DataFileBknas](#), [Bds::DataFileIms](#), [Bds::DataFileLog](#), and [Bds::DataFileTapeDigitiser](#).

7.37.4.3 close()

```
BError Bds::DataFile::close ( ) [virtual]
```

Close the file.

Reimplemented in [Bds::DataFileBds](#), [Bds::DataFileIms](#), and [Bds::DataFileSeed](#).

7.37.4.4 setFormat()

```
BError Bds::DataFile::setFormat (
    BString format ) [virtual]
```

Set the sub-format.

Parameters

<i>format</i>	The data files format
---------------	-----------------------

When a data converter support multiple sub-format, this chooses the one to use.

Reimplemented in [Bds::DataFileAscii](#), [Bds::DataFileBds](#), [Bds::DataFileLog](#), [Bds::DataFileWra](#), and [Bds::DataFileSeed](#).

7.37.4.5 getFileName()

```
BString Bds::DataFile::getFileName ( ) [virtual]
```

Return the file name.

7.37.4.6 getDataOrder()

```
DataFile::DataOrder Bds::DataFile::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented in [Bds::DataFileAd22](#), [Bds::DataFileAscii](#), [Bds::DataFileBdrs](#), [Bds::DataFileBds](#), [Bds::DataFileCd](#), [Bds::DataFileCss](#), [Bds::DataFileGcf](#), [Bds::DataFileIms](#), [Bds::DataFileLac](#), [Bds::DataFileLog](#), [Bds::DataFileWra](#), [Bds::DataFileWraAgso](#), and [Bds::DataFileSeed](#).

7.37.4.7 getFeatures()

```
BError Bds::DataFile::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Returns a bit mask of the data converters featur from the set FeatureCanWrite = 0x01 and FeatureCanRead = 0x02

Reimplemented in [Bds::DataFileAd22](#), [Bds::DataFileAscii](#), [Bds::DataFileBdrs](#), [Bds::DataFileCd](#), [Bds::DataFileCss](#), [Bds::DataFileGcf](#), [Bds::DataFileIdc](#), [Bds::DataFileIms](#), [Bds::DataFileLac](#), [Bds::DataFileLog](#), [Bds::DataFileResponse](#), [Bds::DataFileSac](#), [Bds::DataFileStationXml](#), [Bds::DataFileWra](#), [Bds::DataFileWraAgso](#), and [Bds::DataFileSeed](#).

7.37.4.8 getFixesInfo()

```
BError Bds::DataFile::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

returns a human readable string describing the fixes to corrupted files this data converter can implement when the ReadOptionFixCorruptions option is used

Reimplemented in [Bds::DataFileAd22](#), [Bds::DataFileBdrs](#), [Bds::DataFileCd](#), [Bds::DataFileGcf](#), [Bds::DataFileLac](#), [Bds::DataFileWra](#), and [Bds::DataFileSeed](#).

7.37.4.9 setInfo()

```
BError Bds::DataFile::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionNone ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented in [Bds::DataFileIdc](#), [Bds::DataFileResponse](#), [Bds::DataFileStationXml](#), [Bds::DataFileIms](#), [Bds::DataFileSac](#), [Bds::DataFileAscii](#), [Bds::DataFileBds](#), [Bds::DataFileBknas](#), [Bds::DataFileLog](#), and [Bds::DataFileSeed](#).

7.37.4.10 start()

```
BError Bds::DataFile::start (
    BUInt channel,
    BUInt segment ) [virtual]
```

Start writing next segment of data.

Parameters

<i>channel</i>	The Channel number, 0 for all channels if multiple channel
<i>segment</i>	The data segment number

This starts writing a contiguous time segment of data to the file. It might write a section header etc depending of the file format

Reimplemented in [Bds::DataFileAscii](#), [Bds::DataFileIms](#), [Bds::DataFileLog](#), and [Bds::DataFileSeed](#).

7.37.4.11 writeData()

```
BError Bds::DataFile::writeData (
    const DataBlock & data ) [virtual]
```

Write a block of data.

Parameters

<i>data</i>	A block of data
-------------	-----------------

This writes a block of data to the file

Reimplemented in [Bds::DataFileAscii](#), [Bds::DataFileBds](#), [Bds::DataFileBknas](#), [Bds::DataFileIms](#), [Bds::DataFileLog](#), and [Bds::DataFileSeed](#).

7.37.4.12 end()

```
BError Bds::DataFile::end ( ) [virtual]
```

End write segment.

This defines the end of a data segment.

Reimplemented in [Bds::DataFileAscii](#), [Bds::DataFileIms](#), [Bds::DataFileLog](#), and [Bds::DataFileSeed](#).

7.37.4.13 flush()

```
BError Bds::DataFile::flush ( ) [virtual]
```

Flush data to disk.

Makes sure all of the files contents is flushed from RAM to the disk.

Reimplemented in [Bds::DataFileBds](#).

7.37.4.14 fileNameProcess()

```
BError Bds::DataFile::fileNameProcess ( ) [virtual]
```

Parse the file name for a date/time.

7.37.4.15 getFormat()

```
BError Bds::DataFile::getFormat (
    BString & format ) [virtual]
```

Get sub-format.

Parameters

<i>format</i>	The returned format
---------------	---------------------

Returns the name of the sub-format in use

7.37.4.16 getInfo()

```
BError Bds::DataFile::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fixe corruptions in the samplerate gieven

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented in [Bds::DataFileAd22](#), [Bds::DataFileBdrs](#), [Bds::DataFileBds](#), [Bds::DataFileCd](#), [Bds::DataFileCss](#), [Bds::DataFileGcf](#), [Bds::DataFileLac](#), [Bds::DataFileLog](#), [Bds::DataFileTapeDigitiser](#), [Bds::DataFileWra](#), [Bds::DataFileWraAgso](#), and [Bds::DataFileSeed](#).

7.37.4.17 seekBlock()

```
BError Bds::DataFile::seekBlock (
    BUInt32 channel,
    BUInt segment,
    BTimeStamp time,
    BUInt32 & blockNumber,
    BUInt64 & sampleNumber,
    DataBlock & data ) [virtual]
```

Find requested block on given channel given a time.

Parameters

<i>channel</i>	The channel number. 0 for all channels.
<i>segment</i>	The segment number. 0 for all segments
<i>time</i>	The time of a sample to search for
<i>blockNumber</i>	The returned block number
<i>sampleNumber</i>	The sample number inside the block (based on sampleRate)
<i>data</i>	The matched DataBlock

This seeks for the a [DataBlock](#) in the file that contains a sample for the time provided. If channel is other than 0 then it seeks for a [DataBlock](#) for the given channel. If the data is sample multiplexed a [Channel](#) number of 0 can be given which will return the location of a dataBlock containing all of the Channels. If Segment is 0 then the block number returned is referenced to the start of the set of files. If a particular segment number is given then the seek and the returned block number is within that segment.

Reimplemented in [Bds::DataFileBds](#).

7.37.4.18 readData()

```
BError Bds::DataFile::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented in [Bds::DataFileAd22](#), [Bds::DataFileBdrs](#), [Bds::DataFileCd](#), [Bds::DataFileCss](#), [Bds::DataFileGcf](#), [Bds::DataFileLac](#), [Bds::DataFileLog](#), [Bds::DataFileTapeDigitiser](#), [Bds::DataFileWra](#), [Bds::DataFileWraAgso](#), [Bds::DataFileSeed](#), and [Bds::DataFileBds](#).

7.37.4.19 getMetaData()

```
BError Bds::DataFile::getMetaData (
    ChannelInfos & channelInfos,
    BUInt32 options,
    BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Parameters

<i>channelInfos</i>	The Metadata in the file is returned in this
<i>options</i>	This operators similarly to the getInfo options although some features are not relevant
<i>errors</i>	Returns a list of errors and/or warnings

This function is similar to getInfo except that it returns the list of [Channel](#) metadata in the file if any. Only certain formats can provide this.

Reimplemented in [Bds::DataFileIdc](#), [Bds::DataFileIms](#), [Bds::DataFileResponse](#), [Bds::DataFileSac](#), [Bds::DataFileStationXml](#), and [Bds::DataFileSeed](#).

7.37.4.20 dataErrorFixup()

```
void Bds::DataFile::dataErrorFixup (
    const DataInfo & dataInfo,
    BList< DataError > & errors )
```

Fixup data errors, mainly start/end times to be within data.

7.37.4.21 timeCompare()

```
BInt64 Bds::DataFile::timeCompare (
    BTimeStamp t1,
    BTimeStamp t2,
    BUInt diff )
```

Compare timestamps with a margin.

7.37.4.22 duplicateCheck()

```
int Bds::DataFile::duplicateCheck (
    const DataBlock & data1,
    const DataBlock & data2,
    BUInt channel = 0 )
```

Check if blocks are duplicates.

7.37.4.23 getFilePosition()

```
BUInt64 Bds::DataFile::getFilePosition ( )
```

7.37.4.24 getFormats()

```
DataFormats Bds::DataFile::getFormats ( ) [static]
```

Get list of supported formats.

7.37.5 Member Data Documentation

7.37.5.1 offileName

```
BString Bds::DataFile::offileName [protected]
```

7.37.5.2 omode

```
BString Bds::DataFile::omode [protected]
```

7.37.5.3 offileNameTime

```
BTimeStamp Bds::DataFile::offileNameTime [protected]
```

7.37.5.4 ofile

BFile Bds::DataFile::ofile [protected]

7.37.5.5 oformat

BString Bds::DataFile::oformat [protected]

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

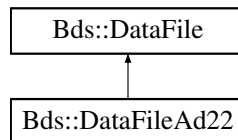
- /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h
- /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.cpp
- /src/blacknest/bds/bds/doc/bdsApi-extra.dox

7.38 Bds::DataFileAd22 Class Reference

Data file convertor for AD22 format files.

```
#include <BdsDataFileAd22.h>
```

Inheritance diagram for Bds::DataFileAd22:



Public Member Functions

- [DataFileAd22](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- **BString** [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUInt32** channel, **BUInt** segment, **BUInt32** blockNumber, [DataBlock](#) &data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.38.1 Detailed Description

Data file convertor for AD22 format files.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 DataFileAd22()

```
Bds::DataFileAd22::DataFileAd22 ( )
```

7.38.3 Member Function Documentation

7.38.3.1 getFeatures()

```
int Bds::DataFileAd22::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.38.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileAd22::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.38.3.3 getFixesInfo()

```
BString Bds::DataFileAd22::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.38.3.4 getInfo()

```
BError Bds::DataFileAd22::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fix corruptions in the samplerate gieven

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.38.3.5 readData()

```
BError Bds::DataFileAd22::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.38.3.6 getFormats()

```
DataFormats Bds::DataFileAd22::getFormats ( ) [static]
```

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

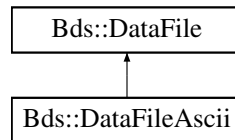
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileAd22.h](#)
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileAd22.cpp](#)

7.39 Bds::DataFileAscii Class Reference

Data file convertor for ASCII format files.

```
#include <BdsDataFileAscii.h>
```

Inheritance diagram for Bds::DataFileAscii:



Public Member Functions

- [DataFileAscii](#) ()
- **BError** [open](#) (**BString** fileName, **BString** mode)
Open the file for read or write.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [setFormat](#) (**BString** format)
Set the sub-format.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionSensorData](#))
Set information on data for write.
- **BError** [start](#) (**BUInt** channel, **BUInt** segment)
Start writing next segment of data.
- **BError** [writeData](#) (const [DataBlock](#) & data)
Write a block of data.
- **BError** [end](#) ()
End write segment.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.39.1 Detailed Description

Data file convertor for ASCII format files.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 DataFileAscii()

```
Bds::DataFileAscii::DataFileAscii ( )
```

7.39.3 Member Function Documentation

7.39.3.1 open()

```
BError Bds::DataFileAscii::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for read or write.

Parameters

<i>fileName</i>	The filename to open. Can be a full path or relative path to the current working directory
<i>mode</i>	The open mode. Can be "r" or "w".

Opens a file of the gievn file format

Reimplemented from [Bds::DataFile](#).

7.39.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileAscii::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.39.3.3 getFeatures()

```
int Bds::DataFileAscii::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Returns a bit mask of the data converters featur from the set FeatureCanWrite = 0x01 and FeatureCanRead = 0x02

Reimplemented from [Bds::DataFile](#).

7.39.3.4 setFormat()

```
BError Bds::DataFileAscii::setFormat (
    BString format ) [virtual]
```

Set the sub-format.

Parameters

<i>format</i>	The data files format
---------------	-----------------------

When a data converter support multiple sub-format, this chooses the one to use.

Reimplemented from [Bds::DataFile](#).

7.39.3.5 setInfo()

```
BError Bds::DataFileAscii::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionSensorData ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.39.3.6 start()

```
BError Bds::DataFileAscii::start (
    BUInt channel,
    BUInt segment ) [virtual]
```

Start writing next segment of data.

Parameters

<i>channel</i>	The Channel number, 0 for all channels if multiple channel
<i>segment</i>	The data segment number

This starts writing a contiguous time segment of data to the file. It might write a section header etc depending of the file format

Reimplemented from [Bds::DataFile](#).

7.39.3.7 writeData()

```
BError Bds::DataFileAscii::writeData (
    const DataBlock & data ) [virtual]
```

Write a block of data.

Parameters

<i>data</i>	A block of data
-------------	-----------------

This writes a block of data to the file

Reimplemented from [Bds::DataFile](#).

7.39.3.8 end()

```
BError Bds::DataFileAscii::end ( ) [virtual]
```

End write segment.

This defines the end of a data segment.

Reimplemented from [Bds::DataFile](#).

7.39.3.9 getFormats()

```
DataFormats Bds::DataFileAscii::getFormats ( ) [static]
```

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

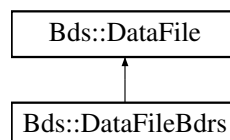
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.cpp](#)

7.40 Bds::DataFileBdrs Class Reference

Data file convertor for BDRS format files.

```
#include <BdsDataFileBdrs.h>
```

Inheritance diagram for Bds::DataFileBdrs:



Public Member Functions

- [DataFileBdrs](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- [BString](#) [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- [BError](#) [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, [BList](#)< [DataError](#) > &errors)
Get info on data.
- [BError](#) [readData](#) ([BUInt32](#) channel, [BUInt](#) segment, [BUInt32](#) blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.40.1 Detailed Description

Data file convertor for BDRS format files.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 DataFileBdrs()

```
Bds::DataFileBdrs::DataFileBdrs ( )
```

7.40.3 Member Function Documentation

7.40.3.1 getFeatures()

```
int Bds::DataFileBdrs::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.40.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileBdrs::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.40.3.3 getFixesInfo()

```
BString Bds::DataFileBdrs::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.40.3.4 getInfo()

```
BError Bds::DataFileBdrs::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fixe corruptions in the samplerate gieven

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.40.3.5 readData()

```
BError Bds::DataFileBdrs::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.40.3.6 getFormats()

```
DataFormats Bds::DataFileBdrs::getFormats ( ) [static]
```

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

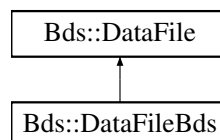
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.cpp](#)

7.41 Bds::DataFileBds Class Reference

This class implements the BDS Data File/Stream access system.

```
#include <BdsDataFileBds.h>
```

Inheritance diagram for Bds::DataFileBds:



Public Types

- enum { [StreamsMax](#) = 256 }
- enum { [DefaultBlockSize](#) = 65536 }
- enum [PackFormat](#) { [PackFormat_Unknown](#) = 0 , [PackFormat_SM](#) = 1 , [PackFormat_CM](#) = 2 , [PackFormat_SM_CC](#) = 3 }

Public Member Functions

- [DataFileBds](#) ()
- [~DataFileBds](#) ()
- **BError** [open](#) (**BString** fileName, **BString** mode)
Open the file for reading or writing.
- **BError** [flush](#) ()
Flush any data to disk even if blocks are not full.
- **BError** [close](#) ()
Close file.
- **BError** [setFormat](#) (**BString** format)
Sets the sub-format.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionSensorData](#))
Sets the information.
- **BError** [writeData](#) (const [DataBlock](#) &data)
Writes a data block to the file.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get information on open file.
- **BError** [seekBlock](#) (**BUint32** channel, **BUint** segment, **BTimeStamp** time, **BUint32** &blockNumber, **BUint64** &sampleNumber, [DataBlock](#) &dataBlock)
Find the block that contains the samples for the time requested.
- **BError** [readData](#) (**BUint32** channel, **BUint** segment, **BUint32** blockNumber, [DataBlock](#) &dataBlock)
Read the data block for the given channel or all channels if blockNumber is 0.
- **BError** [setDiskBlockSize](#) (**BUint32** blockSize)
Sets up file/stream block size.
- **BUint32** [getDiskBlockSize](#) ()
Returns the data block size in bytes.
- **BError** [streamletToChannel](#) (**BUint** streamlet, **BUint** &channel)
Find streamlet given channel.
- **BError** [setWritePositionForAppend](#) ()
Sets the next packet write position.
- **BError** [setReadPositionToStart](#) ()
- **BError** [packetRead](#) ([BdsDataPacket](#) &packet)
Reads a packet from the file.
- **BError** [packetWrite](#) ([BdsDataPacket](#) &packet)
Writes a packet to the file.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()
Get the names of the supported formats.

Additional Inherited Members

7.41.1 Detailed Description

This class implements the BDS Data File/Stream access system.

7.41.2 Member Enumeration Documentation

7.41.2.1 anonymous enum

anonymous enum

Enumerator

StreamsMax	
------------	--

7.41.2.2 anonymous enum

anonymous enum

Enumerator

DefaultBlockSize	
------------------	--

7.41.2.3 PackFormat

enum `Bds::DataFileBds::PackFormat`

Enumerator

PackFormat_Unknown	
PackFormat_SM	
PackFormat_CM	
PackFormat_SM_CC	

7.41.3 Constructor & Destructor Documentation

7.41.3.1 DataFileBds()

`Bds::DataFileBds::DataFileBds ()`

7.41.3.2 ~DataFileBds()

```
Bds::DataFileBds::~~DataFileBds ( )
```

7.41.4 Member Function Documentation

7.41.4.1 open()

```
BError Bds::DataFileBds::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for reading or writing.

Reimplemented from [Bds::DataFile](#).

7.41.4.2 flush()

```
BError Bds::DataFileBds::flush ( ) [virtual]
```

Flush any data to disk even if blocks are not full.

Reimplemented from [Bds::DataFile](#).

7.41.4.3 close()

```
BError Bds::DataFileBds::close ( ) [virtual]
```

Close file.

Reimplemented from [Bds::DataFile](#).

7.41.4.4 setFormat()

```
BError Bds::DataFileBds::setFormat (
    BString format ) [virtual]
```

Sets the sub-format.

Reimplemented from [Bds::DataFile](#).

7.41.4.5 setInfo()

```
BError Bds::DataFileBds::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionSensorData ) [virtual]
```

Sets the information.

Reimplemented from [Bds::DataFile](#).

7.41.4.6 writeData()

```
BError Bds::DataFileBds::writeData (
    const DataBlock & data ) [virtual]
```

Writes a data block to the file.

Reimplemented from [Bds::DataFile](#).

7.41.4.7 getDataOrder()

```
DataFile::DataOrder Bds::DataFileBds::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.41.4.8 getInfo()

```
BError Bds::DataFileBds::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get information on open file.

Reimplemented from [Bds::DataFile](#).

7.41.4.9 seekBlock()

```
BError Bds::DataFileBds::seekBlock (
    BUInt32 channel,
    BUInt segment,
    BTimeStamp time,
    BUInt32 & blockNumber,
    BUInt64 & sampleNumber,
    DataBlock & dataBlock ) [virtual]
```

Find the block that contains the samples for the time requested.

Reimplemented from [Bds::DataFile](#).

7.41.4.10 readData()

```
BError Bds::DataFileBds::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read the data block for the given channel or all channels if blockNumber is 0.

Reimplemented from [Bds::DataFile](#).

7.41.4.11 getFormats()

```
DataFormats Bds::DataFileBds::getFormats ( ) [static]
```

Get the names of the supported formats.

7.41.4.12 setDiskBlockSize()

```
BError Bds::DataFileBds::setDiskBlockSize (
    BUInt32 blockSize )
```

Sets up file/stream block size.

7.41.4.13 getDiskBlockSize()

```
uint32_t Bds::DataFileBds::getDiskBlockSize ( )
```

Returns the data block size in bytes.

7.41.4.14 streamletToChannel()

```
BEError Bds::DataFileBds::streamletToChannel (
    BUInt streamlet,
    BUInt & channel )
```

Find streamlet given channel.

7.41.4.15 setWritePositionForAppend()

```
BEError Bds::DataFileBds::setWritePositionForAppend ( )
```

Sets the next packet write position.

7.41.4.16 setReadPositionToStart()

```
BEError Bds::DataFileBds::setReadPositionToStart ( )
```

7.41.4.17 packetRead()

```
BEError Bds::DataFileBds::packetRead (
    BdsDataPacket & packet )
```

Reads a packet from the file.

7.41.4.18 packetWrite()

```
BEError Bds::DataFileBds::packetWrite (
    BdsDataPacket & packet )
```

Writes a packet to the file.

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

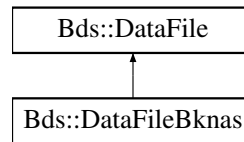
- /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h
- /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.cpp

7.42 Bds::DataFileBknas Class Reference

Data file convertor for BKNAS format files.

```
#include <BdsDataFileBknas.h>
```

Inheritance diagram for Bds::DataFileBknas:



Public Member Functions

- [DataFileBknas](#) ()
- **BError** [open](#) (**BString** fileName, **BString** mode)
Open the file for read or write.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionSensorData](#))
Set information on data for write.
- **BError** [writeData](#) (const [DataBlock](#) &data)
Write a block of data.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.42.1 Detailed Description

Data file convertor for BKNAS format files.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 DataFileBknas()

```
Bds::DataFileBknas::DataFileBknas ( )
```

7.42.3 Member Function Documentation

7.42.3.1 open()

```
BError Bds::DataFileBknas::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for read or write.

Parameters

<i>fileName</i>	The filename to open. Can be a full path or relative path to the current working directory
<i>mode</i>	The open mode. Can be "r" or "w".

Opens a file of the gievn file format

Reimplemented from [Bds::DataFile](#).

7.42.3.2 setInfo()

```
BError Bds::DataFileBknas::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionSensorData ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channellInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.42.3.3 writeData()

```
BError Bds::DataFileBknas::writeData (
    const DataBlock & data ) [virtual]
```

Write a block of data.

Parameters

<i>data</i>	A block of data
-------------	-----------------

This writes a block of data to the file

Reimplemented from [Bds::DataFile](#).

7.42.3.4 getFormats()

`DataFormats` `Bds::DataFileBknas::getFormats ()` [static]

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

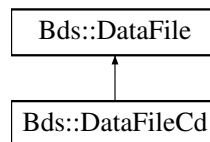
- `/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.h`
- `/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.cpp`

7.43 Bds::DataFileCd Class Reference

Data file convertor for CD1.0 and CD1.1 file formats.

```
#include <BdsDataFileCd.h>
```

Inheritance diagram for `Bds::DataFileCd`:



Public Member Functions

- `DataFileCd ()`
- `int getFeatures ()`
Get bitmask of supported features.
- `DataOrder getDataOrder ()`
Get the expected order of writing data, by sample or by channel.
- `BString getFixesInfo ()`
Get readable list of fixes that can be applied to faulty data files.
- `BError getInfo (DataInfo &dataInfo, DataFileOptions options, BList< DataError > &errors)`
Get info on data.
- `BError readData (BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock & data)`
Read a block.

Static Public Member Functions

- static `DataFormats getFormats ()`

Additional Inherited Members

7.43.1 Detailed Description

Data file convertor for CD1.0 and CD1.1 file formats.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 DataFileCd()

```
Bds::DataFileCd::DataFileCd ( )
```

7.43.3 Member Function Documentation

7.43.3.1 getFeatures()

```
int Bds::DataFileCd::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.43.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileCd::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.43.3.3 getFixesInfo()

```
BString Bds::DataFileCd::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.43.3.4 getInfo()

```
BError Bds::DataFileCd::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fixe corruptions in the samplerate gieven

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.43.3.5 readData()

```
BError Bds::DataFileCd::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.43.3.6 getFormats()

```
DataFormats Bds::DataFileCd::getFormats ( ) [static]
```

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

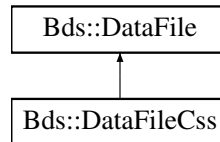
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileCd.h](#)
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileCd.cpp](#)

7.44 Bds::DataFileCss Class Reference

Data file convertor for CSS format files.

```
#include <BdsDataFileCss.h>
```

Inheritance diagram for Bds::DataFileCss:



Public Member Functions

- [DataFileCss](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUInt32** channel, **BUInt** segment, **BUInt32** blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.44.1 Detailed Description

Data file convertor for CSS format files.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 DataFileCss()

```
Bds::DataFileCss::DataFileCss ( )
```

7.44.3 Member Function Documentation

7.44.3.1 getFeatures()

```
int Bds::DataFileCss::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.44.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileCss::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.44.3.3 getInfo()

```
BError Bds::DataFileCss::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fixe corruptions in the samplerate given

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.44.3.4 readData()

```
BError Bds::DataFileCss::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.44.3.5 getFormats()

```
DataFormats Bds::DataFileCss::getFormats ( ) [static]
```

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

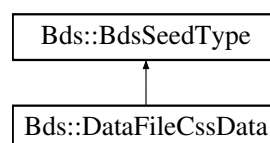
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.cpp](#)

7.45 Bds::DataFileCssData Class Reference

[DataFileCss](#) internal CSS data type.

```
#include <BdsDataFileCss.h>
```

Inheritance diagram for Bds::DataFileCssData:



Public Member Functions

- [DataFileCssData](#) ()
- [~DataFileCssData](#) ()
- **BError** [set](#) ([BString](#) line)

Public Attributes

- **BString** [sta](#)
- **BString** [chan](#)
- double [startTime](#)
- int [wfid](#)
- int [chanid](#)
- int [jdate](#)
- double [endTime](#)
- int [nsamp](#)
- double [sampleRate](#)
- double [calibrationFactor](#)
- double [calibrationFreq](#)
- **BString** [instType](#)
- **BString** [segtype](#)
- **BString** [datatype](#)
- **BString** [clip](#)
- **BString** [dirName](#)
- **BString** [fileName](#)
- **BUInt32** [fileOffset](#)
- int [commId](#)
- **BString** [loadDate](#)
- **BFile** * [file](#)
- **BUInt32** [sampleFormat](#)
- **BUInt32** [sampleSize](#)
- int [sampleBigEndian](#)

7.45.1 Detailed Description

[DataFileCss](#) internal CSS data type.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 DataFileCssData()

```
Bds::DataFileCssData::DataFileCssData ( )
```

7.45.2.2 ~DataFileCssData()

```
Bds::DataFileCssData::~~DataFileCssData ( )
```

7.45.3 Member Function Documentation

7.45.3.1 set()

```
BError Bds::DataFileCssData::set (
    BString line )
```

7.45.4 Member Data Documentation

7.45.4.1 sta

```
BString Bds::DataFileCssData::sta
```

7.45.4.2 chan

```
BString Bds::DataFileCssData::chan
```

7.45.4.3 startTime

```
double Bds::DataFileCssData::startTime
```

7.45.4.4 wfid

```
int Bds::DataFileCssData::wfid
```

7.45.4.5 chanid

```
int Bds::DataFileCssData::chanid
```

7.45.4.6 jdate

```
int Bds::DataFileCssData::jdate
```

7.45.4.7 endTime

```
double Bds::DataFileCssData::endTime
```

7.45.4.8 nsamp

```
int Bds::DataFileCssData::nsamp
```

7.45.4.9 sampleRate

```
double Bds::DataFileCssData::sampleRate
```

7.45.4.10 calibrationFactor

```
double Bds::DataFileCssData::calibrationFactor
```

7.45.4.11 calibrationFreq

```
double Bds::DataFileCssData::calibrationFreq
```

7.45.4.12 instType

```
BString Bds::DataFileCssData::instType
```

7.45.4.13 segtype

BString Bds::DataFileCssData::segtype

7.45.4.14 datatype

BString Bds::DataFileCssData::datatype

7.45.4.15 clip

BString Bds::DataFileCssData::clip

7.45.4.16 dirName

BString Bds::DataFileCssData::dirName

7.45.4.17 fileName

BString Bds::DataFileCssData::fileName

7.45.4.18 fileOffset

BUInt32 Bds::DataFileCssData::fileOffset

7.45.4.19 commId

int Bds::DataFileCssData::commId

7.45.4.20 loadDate

BString Bds::DataFileCssData::loadDate

7.45.4.21 file

```
BFile* Bds::DataFileCssData::file
```

7.45.4.22 sampleFormat

```
BUInt32 Bds::DataFileCssData::sampleFormat
```

7.45.4.23 sampleSize

```
BUInt32 Bds::DataFileCssData::sampleSize
```

7.45.4.24 sampleBigEndian

```
int Bds::DataFileCssData::sampleBigEndian
```

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

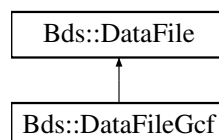
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.cpp](#)

7.46 Bds::DataFileGcf Class Reference

Data file convertor for GCF format files.

```
#include <BdsDataFileGcf.h>
```

Inheritance diagram for Bds::DataFileGcf:



Public Member Functions

- [DataFileGcf](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- [BString](#) [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- [BError](#) [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, [BList](#)< [DataError](#) > &errors)
Get info on data.
- [BError](#) [readData](#) ([BUInt32](#) channel, [BUInt](#) segment, [BUInt32](#) blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.46.1 Detailed Description

Data file convertor for GCF format files.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 DataFileGcf()

```
Bds::DataFileGcf::DataFileGcf ( )
```

7.46.3 Member Function Documentation

7.46.3.1 getFeatures()

```
int Bds::DataFileGcf::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.46.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileGcf::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.46.3.3 getFixesInfo()

```
BString Bds::DataFileGcf::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.46.3.4 getInfo()

```
BError Bds::DataFileGcf::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix known file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: For specific formats fix corruptions in the samplerate given

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.46.3.5 readData()

```
BEError Bds::DataFileGcf::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.46.3.6 getFormats()

```
DataFormats Bds::DataFileGcf::getFormats ( ) [static]
```

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

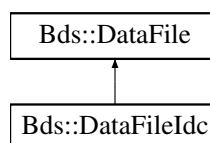
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.cpp](#)

7.47 Bds::DataFileIdc Class Reference

This class defines the interface for IDC response data file access.

```
#include <BdsDataFileIdc.h>
```

Inheritance diagram for Bds::DataFileIdc:



Public Member Functions

- [DataFileIdc](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [getMetaData](#) ([ChannelInfos](#) &channelInfos, **BUInt32** options, **BList**< [DataError](#) > &errors)
Return all known MetaData in the file.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannelInfos](#) &channelInfos, [WriteOptionsList](#) options)
Set information on data for write.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.47.1 Detailed Description

This class defines the interface for IDC response data file access.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 DataFileIdc()

```
Bds::DataFileIdc::DataFileIdc ( )
```

7.47.3 Member Function Documentation

7.47.3.1 getFeatures()

```
int Bds::DataFileIdc::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.47.3.2 getMetaData()

```
BError Bds::DataFileIdc::getMetaData (
    ChannelInfos & channelInfos,
    BUInt32 options,
    BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Reimplemented from [Bds::DataFile](#).

7.47.3.3 setInfo()

```
BError Bds::DataFileIdc::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.47.3.4 getFormats()

```
DataFormats Bds::DataFileIdc::getFormats ( ) [static]
```

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

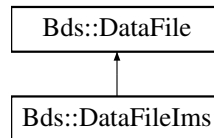
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.cpp](#)

7.48 Bds::DataFileIms Class Reference

Data file convertor for IMS format files.

```
#include <BdsDataFileIms.h>
```

Inheritance diagram for Bds::DataFileIms:



Public Member Functions

- [DataFileIms](#) ()
- **BError** [open](#) ([BString](#) fileName, [BString](#) mode)
Open the file for read or write.
- **BError** [close](#) ()
Close the file.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionNone](#))
Set information on data for write.
- **BError** [start](#) ([BUInt](#) channel, [BUInt](#) segment)
Start writing next segment of data.
- **BError** [writeData](#) (const [DataBlock](#) & data)
Write a block of data.
- **BError** [end](#) ()
End write segment.
- **BError** [getMetaData](#) ([ChannellInfos](#) &channellInfos, [BUInt32](#) options, [BList](#)< [DataError](#) > &errors)
Return all known MetaData in the file.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.48.1 Detailed Description

Data file convertor for IMS format files.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 DataFileImgs()

```
Bds::DataFileImgs::DataFileImgs ( )
```

7.48.3 Member Function Documentation

7.48.3.1 open()

```
BError Bds::DataFileImgs::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for read or write.

Parameters

<i>fileName</i>	The filename to open. Can be a full path or relative path to the current working directory
<i>mode</i>	The open mode. Can be "r" or "w".

Opens a file of the gieven file format

Reimplemented from [Bds::DataFile](#).

7.48.3.2 close()

```
BError Bds::DataFileImgs::close ( ) [virtual]
```

Close the file.

Reimplemented from [Bds::DataFile](#).

7.48.3.3 getDataOrder()

```
DataFile::DataOrder Bds::DataFileImgs::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.48.3.4 getFeatures()

```
int Bds::DataFileIms::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Returns a bit mask of the data converters features from the set FeatureCanWrite = 0x01 and FeatureCanRead = 0x02

Reimplemented from [Bds::DataFile](#).

7.48.3.5 setInfo()

```
BError Bds::DataFileIms::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionNone ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.48.3.6 start()

```
BError Bds::DataFileIms::start (
    BUInt channel,
    BUInt segment ) [virtual]
```

Start writing next segment of data.

Parameters

<i>channel</i>	The Channel number, 0 for all channels if multiple channel
<i>segment</i>	The data segment number

This starts writing a contiguous time segment of data to the file. It might write a section header etc depending of the file format

Reimplemented from [Bds::DataFile](#).

7.48.3.7 writeData()

```
BError Bds::DataFileIms::writeData (
    const DataBlock & data ) [virtual]
```

Write a block of data.

Parameters

<i>data</i>	A block of data
-------------	-----------------

This writes a block of data to the file

Reimplemented from [Bds::DataFile](#).

7.48.3.8 end()

```
BError Bds::DataFileIms::end ( ) [virtual]
```

End write segment.

This defines the end of a data segment.

Reimplemented from [Bds::DataFile](#).

7.48.3.9 getMetaData()

```
BError Bds::DataFileIms::getMetaData (
    ChannelInfos & channelInfos,
    BUInt32 options,
    BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Parameters

<i>channelInfos</i>	The Metadata in the file is returned in this
<i>options</i>	This operators similarly to the getInfo options although some features are not relevant
<i>errors</i>	Returns a list of errors and/or warnings

This function is similar to getInfo except that it returns the list of [Channel](#) metadata in the file if any. Only certain formats can provide this.

Reimplemented from [Bds::DataFile](#).

7.48.3.10 getFormats()

```
DataFormats Bds::DataFileImgs::getFormats ( ) [static]
```

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

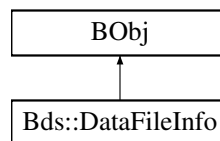
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileImgs.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileImgs.cpp](#)

7.49 Bds::DataFileInfo Class Reference

This class defines information on a [Sensor](#) data file.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::DataFileInfo:



Public Member Functions

- [DataFileInfo](#) ([BUInt32](#) [id](#)=0, [BTimeStamp](#) [startTime](#)= [BTimeStamp](#)(), [BTimeStamp](#) [endTime](#)= [BTimeStamp](#)(), [BString](#) [location](#)= [BString](#)(), [BString](#) [format](#)= [BString](#)(), [BString](#) [url](#)= [BString](#)(), [BString](#) [stream](#)= [BString](#)(), [BString](#) [comment](#)= [BString](#)(), [BUInt32](#) [importUserId](#)=0, [BTimeStamp](#) [importTime](#)= [BTimeStamp](#)(), [BString](#) [state](#)= [BString](#)())
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- **BUInt32** [id](#)
Unique ID when stored in a database or for other uses.
- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time.
- **BString** [location](#)
The storage location.
- **BString** [format](#)
The data format.
- **BString** [url](#)
The URL for file access.
- **BString** [stream](#)
The real-time data stream.
- **BString** [comment](#)
A comment on the file.
- **BUInt32** [importUserId](#)
The user ID of the importing user.
- **BTimeStamp** [importTime](#)
The Time the data was imported.
- **BString** [state](#)
Status info on the import (importing, realtime, failed, ok etc)

7.49.1 Detailed Description

This class defines information on a [Sensor](#) data file.

The raw sensor data for a seismic channel is stored in files in the BDS system. This class defines the database entry that describes this file, its storage location and status. A single file can store one or more channels of seismic data in different formats although normally the BDS data file format is used.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 DataFileInfo()

```
Bds::DataFileInfo::DataFileInfo (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString location = BString(),
    BString format = BString(),
    BString url = BString(),
    BString stream = BString(),
    BString comment = BString(),
    BUInt32 importUserId = 0,
    BTimeStamp importTime = BTimeStamp(),
    BString state = BString() )
```

7.49.3 Member Function Documentation

7.49.3.1 getType()

```
BString Bds::DataFileInfo::getType ( )
```

7.49.3.2 setMembers()

```
BError Bds::DataFileInfo::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.49.3.3 setMember()

```
BError Bds::DataFileInfo::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.49.3.4 getMembers()

```
BError Bds::DataFileInfo::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.49.3.5 getMember()

```
BError Bds::DataFileInfo::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.49.4 Member Data Documentation

7.49.4.1 id

BUInt32 Bds::DataFileInfo::id

Unique ID when stored in a database or for other uses.

7.49.4.2 startTime

BTimeStamp Bds::DataFileInfo::startTime

The Start Time.

7.49.4.3 endTime

BTimeStamp Bds::DataFileInfo::endTime

The End Time.

7.49.4.4 location

BString Bds::DataFileInfo::location

The storage location.

7.49.4.5 format

BString Bds::DataFileInfo::format

The data format.

7.49.4.6 url

BString Bds::DataFileInfo::url

The URL for file access.

7.49.4.7 stream

BString Bds::DataFileInfo::stream

The real-time data stream.

7.49.4.8 comment

BString Bds::DataFileInfo::comment

A comment on the file.

7.49.4.9 importUserId

BUInt32 Bds::DataFileInfo::importUserId

The user ID of the importing user.

7.49.4.10 importTime

BTimeStamp Bds::DataFileInfo::importTime

The Time the data was imported.

7.49.4.11 state

BString Bds::DataFileInfo::state

Status info on the import (importing, realtime, failed, ok etc)

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

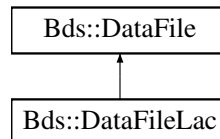
- [BdsD.h](#)
- [BdsD.cc](#)

7.50 Bds::DataFileLac Class Reference

Data file convertor for LAC format files.

```
#include <BdsDataFileLac.h>
```

Inheritance diagram for Bds::DataFileLac:



Public Member Functions

- [DataFileLac](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- **BString** [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUint32** channel, **BUint** segment, **BUint32** blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.50.1 Detailed Description

Data file convertor for LAC format files.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 DataFileLac()

```
Bds::DataFileLac::DataFileLac ( )
```


7.50.3 Member Function Documentation

7.50.3.1 getFeatures()

```
int Bds::DataFileLac::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.50.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileLac::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.50.3.3 getFixesInfo()

```
BString Bds::DataFileLac::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.50.3.4 getInfo()

```
BError Bds::DataFileLac::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation
Generated by Doxygen	
	ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Sample rate is not used to determine block boundaries ReadOptionPrintBlocks: Print blocks of data

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.50.3.5 readData()

```
BError Bds::DataFileLac::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.50.3.6 getFormats()

```
DataFormats Bds::DataFileLac::getFormats ( ) [static]
```

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

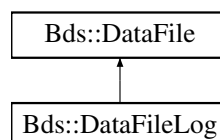
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.cpp](#)

7.51 Bds::DataFileLog Class Reference

Data file convertor for LOG format files.

```
#include <BdsDataFileLog.h>
```

Inheritance diagram for Bds::DataFileLog:



Public Member Functions

- [DataFileLog](#) ()
- **BError** [open](#) ([BString](#) fileName, [BString](#) mode)
Open the file for read or write.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, [BList](#)< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) ([BUInt32](#) channel, [BUInt](#) segment, [BUInt32](#) blockNumber, [DataBlock](#) & data)
Read a block.
- **BError** [setFormat](#) ([BString](#) format)
Set the sub-format.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionSensorData](#))
Set information on data for write.
- **BError** [start](#) ([BUInt](#) channel, [BUInt](#) segment)
Start writing next segment of data.
- **BError** [writeData](#) (const [DataBlock](#) & data)
Write a block of data.
- **BError** [end](#) ()
End write segment.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.51.1 Detailed Description

Data file convertor for LOG format files.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 DataFileLog()

```
Bds::DataFileLog::DataFileLog ( )
```

7.51.3 Member Function Documentation

7.51.3.1 open()

```
BError Bds::DataFileLog::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for read or write.

Parameters

<i>fileName</i>	The filename to open. Can be a full path or relative path to the current working directory
<i>mode</i>	The open mode. Can be "r" or "w".

Opens a file of the gievn file format

Reimplemented from [Bds::DataFile](#).

7.51.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileLog::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.51.3.3 getFeatures()

```
int Bds::DataFileLog::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Returns a bit mask of the data converters featur from the set FeatureCanWrite = 0x01 and FeatureCanRead = 0x02

Reimplemented from [Bds::DataFile](#).

7.51.3.4 getInfo()

```
BError Bds::DataFileLog::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats
	ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks in time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug information for file format ReadOptionFixSampleRate: Fix sample rate differences

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.51.3.5 readData()

```
BError Bds::DataFileLog::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.51.3.6 setFormat()

```
BError Bds::DataFileLog::setFormat (
    BString format ) [virtual]
```

Set the sub-format.

Parameters

<i>format</i>	The data files format
---------------	-----------------------

When a data converter support multiple sub-format, this chooses the one to use.

Reimplemented from [Bds::DataFile](#).

7.51.3.7 setInfo()

```
BError Bds::DataFileLog::setInfo (
    const DataInfo & dataInfo,
```

```
const ChannelInfos & channelInfos,
WriteOptionsList options = WriteOptionSensorData ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.51.3.8 start()

```
BError Bds::DataFileLog::start (
    BUInt channel,
    BUInt segment ) [virtual]
```

Start writing next segment of data.

Parameters

<i>channel</i>	The Channel number, 0 for all channels if multiple channel
<i>segment</i>	The data segment number

This starts writing a contiguous time segment of data to the file. It might write a section header etc depending of the file format

Reimplemented from [Bds::DataFile](#).

7.51.3.9 writeData()

```
BError Bds::DataFileLog::writeData (
    const DataBlock & data ) [virtual]
```

Write a block of data.

Parameters

<i>data</i>	A block of data
-------------	-----------------

This writes a block of data to the file

Reimplemented from [Bds::DataFile](#).

7.51.3.10 end()

```
BError Bds::DataFileLog::end ( ) [virtual]
```

End write segment.

This defines the end of a data segment.

Reimplemented from [Bds::DataFile](#).

7.51.3.11 getFormats()

```
DataFormats Bds::DataFileLog::getFormats ( ) [static]
```

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

- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.cpp](#)

7.52 Bds::DataFileOptions Class Reference

This defines a list of BDS data converter options.

```
#include <BdsDataFile.h>
```

Public Member Functions

- [DataFileOptions](#) (int options=0)
- [operator int](#) ()
- [DataFileOptions](#) & [operator|=](#) (int o)

Public Attributes

- int [optionList](#)
- **BArray**< **BUInt** > [oignoreBlockList](#)

7.52.1 Detailed Description

This defines a list of BDS data converter options.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 DataFileOptions()

```
Bds::DataFileOptions::DataFileOptions (
    int options = 0 ) [inline]
```

7.52.3 Member Function Documentation

7.52.3.1 operator int()

```
Bds::DataFileOptions::operator int ( ) [inline]
```

7.52.3.2 operator" |=()

```
DataFileOptions & Bds::DataFileOptions::operator|= (
    int o ) [inline]
```

7.52.4 Member Data Documentation

7.52.4.1 ooptionList

```
int Bds::DataFileOptions::ooptionList
```

7.52.4.2 oignoreBlockList

```
BArray< BUInt> Bds::DataFileOptions::oignoreBlockList
```

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

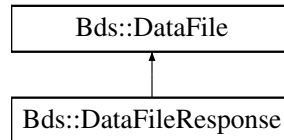
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h](#)

7.53 Bds::DataFileResponse Class Reference

This class defines the interface for generic response data file access.

```
#include <BdsDataFileResponse.h>
```

Inheritance diagram for Bds::DataFileResponse:



Public Member Functions

- [DataFileResponse](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [getMetaData](#) ([ChannellInfos](#) &channellInfos, **BUInt32** options, **BList**< [DataError](#) > &errors)
Return all known MetaData in the file.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options)
Set information on data for write.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.53.1 Detailed Description

This class defines the interface for generic response data file access.

7.53.2 Constructor & Destructor Documentation

7.53.2.1 DataFileResponse()

```
Bds::DataFileResponse::DataFileResponse ( )
```

7.53.3 Member Function Documentation

7.53.3.1 getFeatures()

```
int Bds::DataFileResponse::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.53.3.2 getMetaData()

```
BError Bds::DataFileResponse::getMetaData (
    ChannelInfos & channelInfos,
    BUInt32 options,
    BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Reimplemented from [Bds::DataFile](#).

7.53.3.3 setInfo()

```
BError Bds::DataFileResponse::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.53.3.4 getFormats()

```
DataFormats Bds::DataFileResponse::getFormats ( ) [static]
```

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

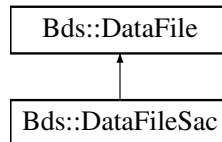
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileResponse.h](#)
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileResponse.cpp](#)

7.54 Bds::DataFileSac Class Reference

Data file convertor for SAC format files.

```
#include <BdsDataFileSac.h>
```

Inheritance diagram for Bds::DataFileSac:



Public Member Functions

- [DataFileSac](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [getMetaData](#) ([ChannellInfos](#) &channellInfos, **BUInt32** options, **BList**< [DataError](#) > &errors)
Return all known MetaData in the file.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionNone](#))
Set information on data for write.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.54.1 Detailed Description

Data file convertor for SAC format files.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 DataFileSac()

```
Bds::DataFileSac::DataFileSac ( )
```

7.54.3 Member Function Documentation

7.54.3.1 getFeatures()

```
int Bds::DataFileSac::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Returns a bit mask of the data converters features from the set FeatureCanWrite = 0x01 and FeatureCanRead = 0x02

Reimplemented from [Bds::DataFile](#).

7.54.3.2 getMetaData()

```
BError Bds::DataFileSac::getMetaData (
    ChannelInfos & channelInfos,
    BUInt32 options,
    BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Reimplemented from [Bds::DataFile](#).

7.54.3.3 setInfo()

```
BError Bds::DataFileSac::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionNone ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.54.3.4 getFormats()

```
DataFormats Bds::DataFileSac::getFormats ( ) [static]
```

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

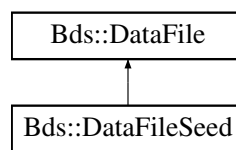
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.cpp](#)

7.55 Bds::DataFileSeed Class Reference

Data file convertor for SEED file formats.

```
#include <BdsDataFileSeed.h>
```

Inheritance diagram for Bds::DataFileSeed:



Public Member Functions

- [DataFileSeed](#) ()
- [~DataFileSeed](#) ()
- **BError** [close](#) ()
Close the file.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BString** [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- **BError** [setFormat](#) (**BString** format)
Set the sub-format.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUint32** channel, **BUint** segment, **BUint32** blockNumber, [DataBlock](#) &data)
Read a block.
- **BError** [getMetaData](#) ([ChannellInfos](#) &channellInfos, **BUint32** options, **BList**< [DataError](#) > &errors)
Return all known MetaData in the file.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options=[WriteOptionSensorData](#))
Set information on data for write.
- **BError** [start](#) (**BUint** channel, **BUint** segment)
Start writing next segment of data.
- **BError** [writeData](#) (const [DataBlock](#) &data)
Write a block of data.
- **BError** [end](#) ()
End write segment.
- void [msrFileWrite](#) (void * data, int len)

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Static Public Attributes

- static **BError** [omsrErr](#)
MSR processing error.
- static int [onoLock](#)
Disable libmseed locking.

Additional Inherited Members

7.55.1 Detailed Description

Data file convertor for SEED file formats.

7.55.2 Constructor & Destructor Documentation

7.55.2.1 DataFileSeed()

```
Bds::DataFileSeed::DataFileSeed ( )
```

7.55.2.2 ~DataFileSeed()

```
Bds::DataFileSeed::~~DataFileSeed ( )
```

7.55.3 Member Function Documentation

7.55.3.1 close()

```
BError Bds::DataFileSeed::close ( ) [virtual]
```

Close the file.

Reimplemented from [Bds::DataFile](#).

7.55.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileSeed::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.55.3.3 getFeatures()

```
int Bds::DataFileSeed::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.55.3.4 getFixesInfo()

```
BString Bds::DataFileSeed::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.55.3.5 setFormat()

```
BError Bds::DataFileSeed::setFormat (
    BString format ) [virtual]
```

Set the sub-format.

Parameters

<i>format</i>	The data files format
---------------	-----------------------

When a data converter support multiple sub-format, this chooses the one to use.

Reimplemented from [Bds::DataFile](#).

7.55.3.6 getInfo()

```
BError Bds::DataFileSeed::getInfo (
    DataInfo & dataInfo,
```

```

    DataFileOptions options,
    BList< DataError > & errors ) [virtual]

```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fixe corruptions in the samplerate given

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.55.3.7 readData()

```

BError Bds::DataFileSeed::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]

```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.55.3.8 getMetaData()

```

BError Bds::DataFileSeed::getMetaData (
    ChannelInfos & channelInfos,

```



```
BUInt32 options,
BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Reimplemented from [Bds::DataFile](#).

7.55.3.9 setInfo()

```
BError Bds::DataFileSeed::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options = WriteOptionSensorData ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format.

Reimplemented from [Bds::DataFile](#).

7.55.3.10 start()

```
BError Bds::DataFileSeed::start (
    BUInt channel,
    BUInt segment ) [virtual]
```

Start writing next segment of data.

Parameters

<i>channel</i>	The Channel number, 0 for all channels if multiple channel
<i>segment</i>	The data segment number

This starts writing a contiguous time segment of data to the file. It might write a section header etc depending of the file format

Reimplemented from [Bds::DataFile](#).

7.55.3.11 writeData()

```
BError Bds::DataFileSeed::writeData (
    const DataBlock & data ) [virtual]
```

Write a block of data.

Parameters

<i>data</i>	A block of data
-------------	-----------------

This writes a block of data to the file

Reimplemented from [Bds::DataFile](#).

7.55.3.12 end()

```
BError Bds::DataFileSeed::end ( ) [virtual]
```

End write segment.

This defines the end of a data segment.

Reimplemented from [Bds::DataFile](#).

7.55.3.13 msrFileWrite()

```
void Bds::DataFileSeed::msrFileWrite (
    void * data,
    int len )
```

7.55.3.14 getFormats()

```
DataFormats Bds::DataFileSeed::getFormats ( ) [static]
```

7.55.4 Member Data Documentation

7.55.4.1 omsrErr

BError Bds::DataFileSeed::omsrErr [static]

MSR processing error.

7.55.4.2 onoLock

int Bds::DataFileSeed::onoLock [static]

Disable libmseed locking.

Disable libmseed lock, for sequential programs.

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

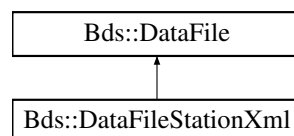
- /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.h
- /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.cpp

7.56 Bds::DataFileStationXml Class Reference

This class defines the interface for generic response data file access.

```
#include <BdsDataFileStationXml.h>
```

Inheritance diagram for Bds::DataFileStationXml:



Public Member Functions

- [DataFileStationXml](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- **BError** [setInfo](#) (const [DataInfo](#) &dataInfo, const [ChannellInfos](#) &channellInfos, [WriteOptionsList](#) options)
Set information on data for write.
- **BError** [getMetaData](#) ([ChannellInfos](#) &channellInfos, **BUInt32** options, **BList**< [DataError](#) > &errors)
Return all known MetaData in the file.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.56.1 Detailed Description

This class defines the interface for generic response data file access.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 DataFileStationXml()

```
Bds::DataFileStationXml::DataFileStationXml ( )
```

7.56.3 Member Function Documentation

7.56.3.1 getFeatures()

```
int Bds::DataFileStationXml::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.56.3.2 setInfo()

```
BError Bds::DataFileStationXml::setInfo (
    const DataInfo & dataInfo,
    const ChannelInfos & channelInfos,
    WriteOptionsList options ) [virtual]
```

Set information on data for write.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>channelInfos</i>	The metadata for the data channels
<i>options</i>	A bitmask of options including: WriteOptionSensorData, WriteOptionNoMetadata WriteOptionSensorData: Writes Sensor data to the file WriteOptionNoMetadata: Disable the writing of Metadata

This function provides the information to create and write to a data file of this given format. *** ???

Reimplemented from [Bds::DataFile](#).

7.56.3.3 getMetaData()

```
BError Bds::DataFileStationXml::getMetaData (
    ChannelInfos & channelInfos,
    BUInt32 options,
    BList< DataError > & errors ) [virtual]
```

Return all known MetaData in the file.

Reimplemented from [Bds::DataFile](#).

7.56.3.4 getFormats()

```
DataFormats Bds::DataFileStationXml::getFormats ( ) [static]
```

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

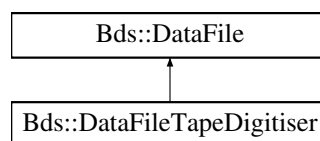
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.cpp](#)

7.57 Bds::DataFileTapeDigitiser Class Reference

This class implements the TapeDigitiser's file output conversion and storing system.

```
#include <BdsDataFileTapeDigitiser.h>
```

Inheritance diagram for Bds::DataFileTapeDigitiser:



Public Member Functions

- [DataFileTapeDigitiser](#) ()
- **BError** [open](#) (**BString** fileName, **BString** mode)
Open the file for reading or writing.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUInt32** channel, **BUInt** segment, **BUInt32** blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.57.1 Detailed Description

This class implements the TapeDigitiser's file output conversion and storing system.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 DataFileTapeDigitiser()

```
Bds::DataFileTapeDigitiser::DataFileTapeDigitiser ( )
```

7.57.3 Member Function Documentation

7.57.3.1 open()

```
BError Bds::DataFileTapeDigitiser::open (
    BString fileName,
    BString mode ) [virtual]
```

Open the file for reading or writing.

Reimplemented from [Bds::DataFile](#).

7.57.3.2 getInfo()

```
BError Bds::DataFileTapeDigitiser::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSampleRate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate : performs data file validation

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.57.3.3 readData()

```
BError Bds::DataFileTapeDigitiser::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.57.3.4 getFormats()

```
DataFormats Bds::DataFileTapeDigitiser::getFormats ( ) [static]
```

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

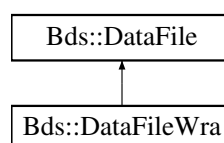
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.h](#)
- [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.cpp](#)

7.58 Bds::DataFileWra Class Reference

Data file convertor for WRA format files.

```
#include <BdsDataFileWra.h>
```

Inheritance diagram for Bds::DataFileWra:



Public Member Functions

- [DataFileWra](#) ()
- **BError** [setFormat](#) (**BString** format)
Set the sub-format.
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- **BString** [getFixesInfo](#) ()
Get readable list of fixes that can be applied to faulty data files.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUint32** channel, **BUint** segment, **BUint32** blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.58.1 Detailed Description

Data file convertor for WRA format files.

7.58.2 Constructor & Destructor Documentation

7.58.2.1 DataFileWra()

```
Bds::DataFileWra::DataFileWra ( )
```

7.58.3 Member Function Documentation

7.58.3.1 setFormat()

```
BError Bds::DataFileWra::setFormat (
    BString format ) [virtual]
```

Set the sub-format.

Parameters

<i>format</i>	The data files format
---------------	-----------------------

When a data converter support multiple sub-format, this chooses the one to use.

Reimplemented from [Bds::DataFile](#).

7.58.3.2 getFeatures()

```
int Bds::DataFileWra::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.58.3.3 getDataOrder()

```
DataFile::DataOrder Bds::DataFileWra::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.58.3.4 getFixesInfo()

```
BString Bds::DataFileWra::getFixesInfo ( ) [virtual]
```

Get readable list of fixes that can be applied to faulty data files.

Reimplemented from [Bds::DataFile](#).

7.58.3.5 getInfo()

```
BError Bds::DataFileWra::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file corruptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fix corruptions in the samplerate given

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.58.3.6 readData()

```
BError Bds::DataFileWra::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.58.3.7 getFormats()

```
DataFormats Bds::DataFileWra::getFormats ( ) [static]
```

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

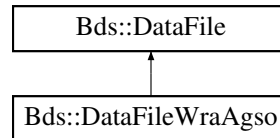
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileWra.h](#)
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileWra.cpp](#)

7.59 Bds::DataFileWraAgso Class Reference

Data file convertor for WRA AGSO format files.

```
#include <BdsDataFileWraAgso.h>
```

Inheritance diagram for Bds::DataFileWraAgso:



Public Member Functions

- [DataFileWraAgso](#) ()
- int [getFeatures](#) ()
Get bitmask of supported features.
- [DataOrder](#) [getDataOrder](#) ()
Get the expected order of writing data, by sample or by channel.
- **BError** [getInfo](#) ([DataInfo](#) &dataInfo, [DataFileOptions](#) options, **BList**< [DataError](#) > &errors)
Get info on data.
- **BError** [readData](#) (**BUint32** channel, **BUint** segment, **BUint32** blockNumber, [DataBlock](#) & data)
Read a block.

Static Public Member Functions

- static [DataFormats](#) [getFormats](#) ()

Additional Inherited Members

7.59.1 Detailed Description

Data file convertor for WRA AGSO format files.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 DataFileWraAgso()

```
Bds::DataFileWraAgso::DataFileWraAgso ( )
```

7.59.3 Member Function Documentation

7.59.3.1 getFeatures()

```
int Bds::DataFileWraAgso::getFeatures ( ) [virtual]
```

Get bitmask of supported features.

Reimplemented from [Bds::DataFile](#).

7.59.3.2 getDataOrder()

```
DataFile::DataOrder Bds::DataFileWraAgso::getDataOrder ( ) [virtual]
```

Get the expected order of writing data, by sample or by channel.

Reimplemented from [Bds::DataFile](#).

7.59.3.3 getInfo()

```
BError Bds::DataFileWraAgso::getInfo (
    DataInfo & dataInfo,
    DataFileOptions options,
    BList< DataError > & errors ) [virtual]
```

Get info on data.

Parameters

<i>dataInfo</i>	Info on the Sensor data channels to be written
<i>options</i>	A bitmask of options including: ReadOptionValidate = 0x01, ReadOptionFileNameProcess = 0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10, ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80, ReadOptionFixSampleRate = 0x100
<i>errors</i>	Returns a list of errors and/or warnings ReadOptionValidate: performs data file validation ReadOptionFileNameProcess: Validate the filename for particular file formats ReadOptionFixCorruptions: Fix know file currptions ReadOptionReorder: Re-order blocks into time order ReadOptionDeleteDuplicates: Delete duplicate blocks where they are completely duplicate in startTime and data. ReadOptionInfoExtra: Read the extra string Metadata information specific to the format ReadOptionIgnoreSamplerate: Ignore samplerate differences ReadOptionPrintBlocks: debug option to print information on the blocks to stdout ReadOptionFixSampleRate: Fors specific formats fixe currptions in the samplerate given

This function scans the file and returns detailed Metadata from the blocks in the file.

Reimplemented from [Bds::DataFile](#).

7.59.3.4 readData()

```
BError Bds::DataFileWraAgso::readData (
    BUInt32 channel,
    BUInt segment,
    BUInt32 blockNumber,
    DataBlock & dataBlock ) [virtual]
```

Read a block.

Parameters

<i>channel</i>	The channel number. 0 Means all channels
<i>segment</i>	The segment number. 0 Means all segments
<i>blockNumber</i>	The block number. This starts from 0.
<i>dataBlock</i>	The returned DataBlock

This function is used to read a data block from the file.

Reimplemented from [Bds::DataFile](#).

7.59.3.5 getFormats()

```
DataFormats Bds::DataFileWraAgso::getFormats ( ) [static]
```

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

- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileWraAgso.h](#)
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileWraAgso.cpp](#)

7.60 Bds::DataFormat Class Reference

This holds information on a [Sensor](#) data format.

```
#include <BdsD.h>
```

Public Member Functions

- [DataFormat](#) ([BList](#)< [BString](#) > names= [BList](#)< [BString](#) >(), [BInt32](#) dataRead=0, [BInt32](#) dataWrite=0, [BInt32](#) metadataRead=0, [BInt32](#) metadataWrite=0, [BString](#) extension= [BString](#)(), [BString](#) description= [BString](#)())

Public Attributes

- **BList**< **BString** > [names](#)
The format names.
- **BInt32** [dataRead](#)
Ability to read [Sensor](#) data.
- **BInt32** [dataWrite](#)
Ability to write [Sensor](#) data.
- **BInt32** [metadataRead](#)
MetaData read supported.
- **BInt32** [metadataWrite](#)
MetaData write supported.
- **BString** [extension](#)
Default filename extension.
- **BString** [description](#)
The description.

7.60.1 Detailed Description

This holds information on a [Sensor](#) data format.

It is used by the BDS data convertors to define which data formats they support.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 DataFormat()

```
Bds::DataFormat::DataFormat (
    BList< BString > names = BList< BString >(),
    BInt32 dataRead = 0,
    BInt32 dataWrite = 0,
    BInt32 metadataRead = 0,
    BInt32 metadataWrite = 0,
    BString extension = BString(),
    BString description = BString() )
```

7.60.3 Member Data Documentation

7.60.3.1 names

```
BList< BString > Bds::DataFormat::names
```

The format names.

7.60.3.2 dataRead

UInt32 Bds::DataFormat::dataRead

Ability to read [Sensor](#) data.

7.60.3.3 dataWrite

UInt32 Bds::DataFormat::dataWrite

Ability to write [Sensor](#) data.

7.60.3.4 metadataRead

UInt32 Bds::DataFormat::metadataRead

MetaData read supported.

7.60.3.5 metadataWrite

UInt32 Bds::DataFormat::metadataWrite

MetaData write supported.

7.60.3.6 extension

String Bds::DataFormat::extension

Default filename extension.

7.60.3.7 description

String Bds::DataFormat::description

The description.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.61 Bds::DataFormatAll Class Reference

This class defines the interface for generic data file access.

```
#include <BdsDataLib.h>
```

Public Member Functions

- [DataFormatAll](#) ()
- [~DataFormatAll](#) ()
- **BError** [formatList](#) ([DataFormats](#) &formats)
- **BError** [formatGet](#) (**BString** **format**, [DataFile](#) *&dataFile, [DataFormatSet](#) formatSet=[DataFormatSetNone](#))
Returns a list of all data converters.
- **BString** [formatGetExtension](#) (**BString** **format**)
Searches all of the BDS data converters to find out if any supports the given format. If one is found the appropriate data converter is initialised and a pointer to it returned in file.

Protected Member Functions

- int [findFormat](#) ([DataFormats](#) dataFormats, **BString** **string**, [DataFormatSet](#) formatSet)
Returns the default file extension for the given format.

7.61.1 Detailed Description

This class defines the interface for generic data file access.

It allows programs to get a list of all of the supported data formats and to get a suitable BDS data convertor for accessing the file.

7.61.2 Constructor & Destructor Documentation

7.61.2.1 DataFormatAll()

```
Bds::DataFormatAll::DataFormatAll ( )
```

7.61.2.2 ~DataFormatAll()

```
Bds::DataFormatAll::~~DataFormatAll ( )
```

7.61.3 Member Function Documentation

7.61.3.1 formatList()

```
BError Bds::DataFormatAll::formatList (
    DataFormats & formats )
```

7.61.3.2 formatGet()

```
BError Bds::DataFormatAll::formatGet (
    BString format,
    DataFile *& dataFile,
    DataFormatSet formatSet = DataFormatSetNone )
```

Returns a list of all data converters.

7.61.3.3 formatGetExtension()

```
BString Bds::DataFormatAll::formatGetExtension (
    BString format )
```

Searches all of the BDS data converters to find out if any supports the given format. If one is found the appropriate data converter is initialised and a pointer to it returned in file.

7.61.3.4 findFormat()

```
int Bds::DataFormatAll::findFormat (
    DataFormats dataFormats,
    BString string,
    DataFormatSet formatSet ) [protected]
```

Returns the default file extension for the given format.

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

- /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.h
- /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.cpp

7.62 Bds::DataHandle Class Reference

This defines a handle to a sensor data stream/file when opened for read or write.

```
#include <BdsD.h>
```

Public Member Functions

- [DataHandle](#) ([BUInt32](#) [handle](#)=0, [BUInt32](#) [dataFileId](#)=0)

Public Attributes

- [BUInt32](#) [handle](#)
Opaque file handle.
- [BUInt32](#) [dataFileId](#)
The data file ID if opened for write.

7.62.1 Detailed Description

This defines a handle to a sensor data stream/file when opened for read or write.

7.62.2 Constructor & Destructor Documentation

7.62.2.1 DataHandle()

```
Bds::DataHandle::DataHandle (
    BUInt32 handle = 0,
    BUInt32 dataFileId = 0 )
```

7.62.3 Member Data Documentation

7.62.3.1 handle

```
BUInt32 Bds::DataHandle::handle
```

Opaque file handle.

7.62.3.2 dataFileId

```
BUInt32 Bds::DataHandle::dataFileId
```

The data file ID if opened for write.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.63 Bds::DataInfo Class Reference

This class defines information on a set of data.

```
#include <BdsD.h>
```

Public Member Functions

- **DataInfo** (**BTimeStamp** [startTime](#)= **BTimeStamp**(), **BTimeStamp** [endTime](#)= **BTimeStamp**(), **BString** [array](#)= **BString**(), **BString** [description](#)= **BString**(), **BUInt32** [synchronous](#)=0, **BArray**< **BArray**< **DataChannel** > > [channels](#)= **BArray**< **BArray**< **DataChannel** > >(), **BDict**< **BString** > [info](#)= **BDict**< **BString** >(), **BDict**< **BString** > [infoExtra](#)= **BDict**< **BString** >(), **BList**< **BString** > [warnings](#)= **BList**< **BString** >())

Public Attributes

- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time.
- **BString** [array](#)
The Seismic Array that all of the channels are from, if just one.
- **BString** [description](#)
The Comment.
- **BUInt32** [synchronous](#)
The channels are synchronously sampled.
- **BArray**< **BArray**< **DataChannel** > > [channels](#)
The Data channels. Each channel can have multiple segments of data.
- **BDict**< **BString** > [info](#)
Info on the set of channels.
- **BDict**< **BString** > [infoExtra](#)
Extra Info on the set of channels. Used for extended error/logging information.
- **BList**< **BString** > [warnings](#)
Warnings on the data set.

7.63.1 Detailed Description

This class defines information on a set of data.

This describes a set of [Sensor](#) data. It is used to returning basic [Sensor](#) data information when performing a selection of data or detailed information when enquiring information from an actual [Sensor](#) data file. All of the detailed information comes from the [Sensor](#) data files themselves. This includes the info, infoExtra and warnings information. The channels array contains an array of data segments per channel split at time discontinuities. For basic information this could be a single segment over a time period. However, when equiring detailed information from a file it will contain an entry per contiguous data segemnt in the file.

7.63.2 Constructor & Destructor Documentation

7.63.2.1 DataInfo()

```
Bds::DataInfo::DataInfo (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString array = BString(),
    BString description = BString(),
    BUInt32 synchronous = 0,
    BArray< BArray< DataChannel > > channels = BArray< BArray<DataChannel > >(),
    BDict< BString > info = BDict< BString >(),
    BDict< BString > infoExtra = BDict< BString >(),
    BList< BString > warnings = BList< BString >() )
```

7.63.3 Member Data Documentation

7.63.3.1 startTime

BTimeStamp Bds::DataInfo::startTime

The Start Time.

7.63.3.2 endTime

BTimeStamp Bds::DataInfo::endTime

The End Time.

7.63.3.3 array

BString Bds::DataInfo::array

The Seismic Array that all of the channels are from, if just one.

7.63.3.4 description

BString Bds::DataInfo::description

The Comment.

7.63.3.5 synchronous

```
BUInt32 Bds::DataInfo::synchronous
```

The channels are synchronously sampled.

7.63.3.6 channels

```
BArray< BArray<DataChannel > > Bds::DataInfo::channels
```

The Data channels. Each channel can have multiple segments of data.

7.63.3.7 info

```
BDict< BString > Bds::DataInfo::info
```

Info on the set of channels.

7.63.3.8 infoExtra

```
BDict< BString > Bds::DataInfo::infoExtra
```

Extra Info on the set of channels. Used for extended error/logging information.

7.63.3.9 warnings

```
BList< BString > Bds::DataInfo::warnings
```

Warnings on the data set.

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

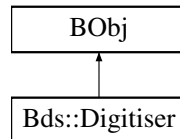
- [BdsD.h](#)
- [BdsD.cc](#)

7.64 Bds::Digitiser Class Reference

This class defines a seismic [Digitiser](#).

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Digitiser:



Public Member Functions

- [Digitiser](#) ([BUInt32](#) id=0, [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTime](#)↵
Stamp(), [BString](#) name= [BString](#)(), [BString](#) type= [BString](#)(), [BString](#) serialNumber= [BString](#)(),
[BUInt32](#) numberChannels=0, [BFloat64](#) baseSamplingFrequency=0, [BFloat64](#) initialSamplingFrequency=0,
[BFloat64](#) gain=0, [BInt32](#) shared=0)
- [BString](#) getType ()
- [BError](#) setMembers ([BDictString](#) &members)
- [BError](#) setMember ([BString](#) name, [BString](#) value)
- [BError](#) getMembers ([BDictString](#) &members)
- [BError](#) getMember ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) id
The ID.
- [BTimeStamp](#) startTime
The Start Time.
- [BTimeStamp](#) endTime
The End Time the channel was available.
- [BString](#) name
The Digitisers name.
- [BString](#) type
The Digitisers type.
- [BString](#) serialNumber
The digitisers's serial number.
- [BUInt32](#) numberChannels
The number of supported channels.
- [BFloat64](#) baseSamplingFrequency
The base sampling frequency.
- [BFloat64](#) initialSamplingFrequency
The initial pre-decimation sampling frequency.
- [BFloat64](#) gain
The overall gain of the digitiser at the manufacturers calibration frequency. (For information only)
- [BInt32](#) shared
This digitiser is shared.

7.64.1 Detailed Description

This class defines a seismic [Digitiser](#).

This just stores information on the seismic instrument's digitiser. Its contents is generally for information only.

7.64.2 Constructor & Destructor Documentation

7.64.2.1 Digitiser()

```
Bds::Digitiser::Digitiser (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString name = BString(),
    BString type = BString(),
    BString serialNumber = BString(),
    BUInt32 numberChannels = 0,
    BFloat64 baseSamplingFrequency = 0,
    BFloat64 initialSamplingFrequency = 0,
    BFloat64 gain = 0,
    BInt32 shared = 0 )
```

7.64.3 Member Function Documentation

7.64.3.1 getType()

```
BString Bds::Digitiser::getType ( )
```

7.64.3.2 setMembers()

```
BError Bds::Digitiser::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from [BObj](#).

7.64.3.3 setMember()

```
BError Bds::Digitiser::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.64.3.4 getMembers()

```
BError Bds::Digitiser::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.64.3.5 getMember()

```
BError Bds::Digitiser::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.64.4 Member Data Documentation

7.64.4.1 id

```
BUInt32 Bds::Digitiser::id
```

The ID.

7.64.4.2 startTime

```
BTimeStamp Bds::Digitiser::startTime
```

The Start Time.

7.64.4.3 endTime

BTimeStamp Bds::Digitiser::endTime

The End Time the channel was available.

7.64.4.4 name

BString Bds::Digitiser::name

The Digitisers name.

7.64.4.5 type

BString Bds::Digitiser::type

The Digitisers type.

7.64.4.6 serialNumber

BString Bds::Digitiser::serialNumber

The digitisers's serial number.

7.64.4.7 numberChannels

BUInt32 Bds::Digitiser::numberChannels

The number of supported channels.

7.64.4.8 baseSamplingFrequency

BFloat64 Bds::Digitiser::baseSamplingFrequency

The base sampling frequency.

7.64.4.9 initialSamplingFrequency

BFloat64 Bds::Digitiser::initialSamplingFrequency

The initial pre-decimation sampling frequency.

7.64.4.10 gain

BFloat64 Bds::Digitiser::gain

The overall gain of the digitiser at the manufacturers calibration frequency. (For information only)

7.64.4.11 shared

BInt32 Bds::Digitiser::shared

This digitiser is shared.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.65 Bds::Event Class Reference

This class defines a seismic event.

```
#include <BdsD.h>
```

Public Member Functions

- [Event](#) ([BUInt32](#) id=0, [BUInt32](#) userId=0, [BString](#) type= [BString](#)(), [BString](#) title= [BString](#)(), [BString](#) network= [BString](#)(), [BString](#) source= [BString](#)(), [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTimeStamp](#)(), [BTimeStamp](#) eventTime= [BTimeStamp](#)(), [BFloat64](#) longitude=0, [BFloat64](#) latitude=0, [BFloat64](#) elevation=0, [BFloat64](#) waterDepth=0, [BFloat64](#) magnitude=0, [BString](#) magnitudeUnits= [BString](#)(), [BString](#) description= [BString](#)(), [BString](#) notes= [BString](#)(), [BDict](#)< [BString](#) > extra= [BDict](#)< [BString](#) >(), [BList](#)< [SelectionChannel](#) > dataChannels= [BList](#)< [SelectionChannel](#) >()

Public Attributes

- **BUInt32** [id](#)
Unique id defining this event within the BDS system.
- **BUInt32** [userId](#)
User ID for initial creator of this event. This allows them to edit these events.
- **BString** [type](#)
The event type (a hierarchy of types)
- **BString** [title](#)
Some text describing the event.
- **BString** [network](#)
Unique network for this event if a project.
- **BString** [source](#)
Unique source for this event if a project.
- **BTimeStamp** [startTime](#)
The startTime of the event to the nearest microsecond. For data access.
- **BTimeStamp** [endTime](#)
The endTime of the event to the nearest microsecond. For data access.
- **BTimeStamp** [eventTime](#)
The actual time of the event to the nearest microsecond.
- **BFloat64** [longitude](#)
The longitude in degrees using the WGS84 datum.
- **BFloat64** [latitude](#)
The Latitude in degrees using the WGS84 datum.
- **BFloat64** [elevation](#)
The ground level elevation in meters from the WGS84 ellipsoid (Sea level)
- **BFloat64** [waterDepth](#)
Water depth of the event if in water.
- **BFloat64** [magnitude](#)
Magnitude of the event.
- **BString** [magnitudeUnits](#)
Magnitude Units.
- **BString** [description](#)
General description of the Event/Project.
- **BString** [notes](#)
General notes on the Event/project.
- **BDict**< **BString** > [extra](#)
An array of name value pairs for extra metadata specific to particular events.
- **BList**< [SelectionChannel](#) > [dataChannels](#)
List of BDS Channels of associated sensor data files if any.

7.65.1 Detailed Description

This class defines a seismic event.

Each event defines Metadata for a seismic event. This includes a list of all of the data channels relevant to the event. Given this information the BDS is able, given an [Event](#) title, to export all of the Metadata and [Sensor](#) data relevant to this event.

7.65.2 Constructor & Destructor Documentation

7.65.2.1 Event()

```
Bds::Event::Event (
    BUInt32 id = 0,
    BUInt32 userId = 0,
    BString type = BString(),
    BString title = BString(),
    BString network = BString(),
    BString source = BString(),
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BTimeStamp eventTime = BTimeStamp(),
    BFloat64 longitude = 0,
    BFloat64 latitude = 0,
    BFloat64 elevation = 0,
    BFloat64 waterDepth = 0,
    BFloat64 magnitude = 0,
    BString magnitudeUnits = BString(),
    BString description = BString(),
    BString notes = BString(),
    BDict< BString > extra = BDict< BString >(),
    BList< SelectionChannel > dataChannels = BList<SelectionChannel >() )
```

7.65.3 Member Data Documentation

7.65.3.1 id

```
BUInt32 Bds::Event::id
```

Unique id defining this event within the BDS system.

7.65.3.2 userId

```
BUInt32 Bds::Event::userId
```

User ID for initial creator of this event. This allows them to edit these events.

7.65.3.3 type

BString Bds::Event::type

The event type (a hierarchy of types)

7.65.3.4 title

BString Bds::Event::title

Some text describing the event.

7.65.3.5 network

BString Bds::Event::network

Unique network for this event if a project.

7.65.3.6 source

BString Bds::Event::source

Unique source for this event if a project.

7.65.3.7 startTime

BTimeStamp Bds::Event::startTime

The startTime of the event to the nearest microsecond. For data access.

7.65.3.8 endTime

BTimeStamp Bds::Event::endTime

The endTime of the event to the nearest microsecond. For data access.

7.65.3.9 eventTime

BTimeStamp Bds::Event::eventTime

The actual time of the event to the nearest microsecond.

7.65.3.10 longitude

BFloat64 Bds::Event::longitude

The longitude in degrees using the WGS84 datum.

7.65.3.11 latitude

BFloat64 Bds::Event::latitude

The Latitude in degrees using the WGS84 datum.

7.65.3.12 elevation

BFloat64 Bds::Event::elevation

The ground level elevation in meters from the WGS84 ellipsoid (Sea level)

7.65.3.13 waterDepth

BFloat64 Bds::Event::waterDepth

Water depth of the event if in water.

7.65.3.14 magnitude

BFloat64 Bds::Event::magnitude

Magnitude of the event.

7.65.3.15 magnitudeUnits

BString Bds::Event::magnitudeUnits

Magnitude Units.

7.65.3.16 description

BString Bds::Event::description

General description of the Event/Project.

7.65.3.17 notes

BString Bds::Event::notes

General notes on the Event/project.

7.65.3.18 extra

BDict< **BString** > Bds::Event::extra

An array of name value pairs for extra metadata specific to particular events.

7.65.3.19 dataChannels

BList<[SelectionChannel](#) > Bds::Event::dataChannels

List of BDS Channels of associated sensor data files if any.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.66 Bds::Fap Class Reference

This class defines an entry in an Amplitude/Phase [Response](#) table.

```
#include <BdsD.h>
```

Public Member Functions

- [Fap](#) ([BFloat64](#) [frequency](#)=0, [BFloat64](#) [amplitude](#)=0, [BFloat64](#) [phase](#)=0)

Public Attributes

- [BFloat64](#) [frequency](#)
The frequency.
- [BFloat64](#) [amplitude](#)
The Amplitude.
- [BFloat64](#) [phase](#)
The Phase.

7.66.1 Detailed Description

This class defines an entry in an Amplitude/Phase [Response](#) table.

7.66.2 Constructor & Destructor Documentation

7.66.2.1 Fap()

```
Bds::Fap::Fap (
    BFloat64 frequency = 0,
    BFloat64 amplitude = 0,
    BFloat64 phase = 0 )
```

7.66.3 Member Data Documentation

7.66.3.1 frequency

[BFloat64](#) [Bds::Fap::frequency](#)

The frequency.

7.66.3.2 amplitude

[BFloat64](#) [Bds::Fap::amplitude](#)

The Amplitude.

7.66.3.3 phase

BFloat64 Bds::Fap::phase

The Phase.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.67 Bds::Fir Class Reference

This class defines an FIR response table.

```
#include <BdsD.h>
```

Public Member Functions

- [Fir](#) ([BArray](#)< [FirEntry](#) > b= [BArray](#)< [FirEntry](#) >(), [BArray](#)< [FirEntry](#) > a= [BArray](#)< [FirEntry](#) >())

Public Attributes

- [BArray](#)< [FirEntry](#) > [b](#)
FIR Numerator array.
- [BArray](#)< [FirEntry](#) > [a](#)
FIR Denominator array.

7.67.1 Detailed Description

This class defines an FIR response table.

This has an array of the A and B coefficients.

7.67.2 Constructor & Destructor Documentation

7.67.2.1 Fir()

```
Bds::Fir::Fir (
    BArray< FirEntry > b = BArray<FirEntry > (),
    BArray< FirEntry > a = BArray<FirEntry > () )
```

7.67.3 Member Data Documentation

7.67.3.1 **b**

BArray<[FirEntry](#) > Bds::Fir::b

FIR Numerator array.

7.67.3.2 **a**

BArray<[FirEntry](#) > Bds::Fir::a

FIR Denominator array.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.68 Bds::FirEntry Class Reference

This class defines an entry in a FIR coefficient table.

```
#include <BdsD.h>
```

Public Member Functions

- [FirEntry](#) ([BFloat64](#) [coefficient](#)=0, [BFloat64](#) [error](#)=0)

Public Attributes

- [BFloat64](#) [coefficient](#)
FIR Coefficient Value.
- [BFloat64](#) [error](#)
FIR Coefficien Error.

7.68.1 Detailed Description

This class defines an entry in a FIR coefficient table.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 FirEntry()

```
Bds::FirEntry::FirEntry (
    BFloat64 coefficient = 0,
    BFloat64 error = 0 )
```

7.68.3 Member Data Documentation

7.68.3.1 coefficient

```
BFloat64 Bds::FirEntry::coefficient
```

FIR Coefficient Value.

7.68.3.2 error

```
BFloat64 Bds::FirEntry::error
```

FIR Coefficient Error.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.69 Bds::GcfChannel Struct Reference

[DataFileGcf](#) internal GCF channel information.

```
#include <BdsDataFileGcf.h>
```

Public Attributes

- BString [systemId](#)
- BString [streamId](#)
- BUInt [type](#)
- BUInt [sampleRate](#)
- BUInt [format](#)
- BUInt32 [channel](#)

7.69.1 Detailed Description

[DataFileGcf](#) internal GCF channel information.

7.69.2 Member Data Documentation

7.69.2.1 systemId

BString Bds::GcfChannel::systemId

7.69.2.2 streamId

BString Bds::GcfChannel::streamId

7.69.2.3 type

BUInt Bds::GcfChannel::type

7.69.2.4 sampleRate

BUInt Bds::GcfChannel::sampleRate

7.69.2.5 format

BUInt Bds::GcfChannel::format

7.69.2.6 channel

BUInt32 Bds::GcfChannel::channel

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

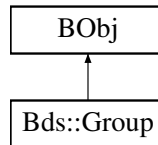
- /src/blacknest/bds/bds/bdsDataLib/[BdsDataFileGcf.h](#)

7.70 Bds::Group Class Reference

This holds information on a [User](#) security group.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Group:



Public Member Functions

- [Group](#) ([BUInt32](#) id=0, [BString](#) group= [BString](#)(), [BString](#) description= [BString](#)())
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) [id](#)
The unique id.
- [BString](#) [group](#)
The [Group](#) name.
- [BString](#) [description](#)
The Groups description.

7.70.1 Detailed Description

This holds information on a [User](#) security group.

The BDS has the concept of a security group that users can belong to. This class defines that security group as stored in the database.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 Group()

```

Bds::Group::Group (
    BUInt32 id = 0,
    BString group = BString(),
    BString description = BString() )
  
```

7.70.3 Member Function Documentation

7.70.3.1 getType()

```
BString Bds::Group::getType ( )
```

7.70.3.2 setMembers()

```
BError Bds::Group::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.70.3.3 setMember()

```
BError Bds::Group::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.70.3.4 getMembers()

```
BError Bds::Group::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.70.3.5 getMember()

```
BError Bds::Group::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.70.4 Member Data Documentation

7.70.4.1 id

BUInt32 Bds::Group::id

The unique id.

7.70.4.2 group

BString Bds::Group::group

The [Group](#) name.

7.70.4.3 description

BString Bds::Group::description

The Groups description.

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

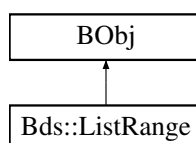
- [BdsD.h](#)
- [BdsD.cc](#)

7.71 Bds::ListRange Class Reference

This class defines an integer based range.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::ListRange:



Public Member Functions

- [ListRange](#) ([BUInt32](#) [start](#)=0, [BUInt32](#) [number](#)=0, [BInt32](#) [reverse](#)=0)
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) [start](#)
The start position.
- [BUInt32](#) [number](#)
The number of items.
- [BInt32](#) [reverse](#)
List from end.

7.71.1 Detailed Description

This class defines an integer based range.

It is used for limit the number of items returned in selections etc.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 ListRange()

```
Bds::ListRange::ListRange (
    BUInt32 start = 0,
    BUInt32 number = 0,
    BInt32 reverse = 0 )
```

7.71.3 Member Function Documentation

7.71.3.1 getType()

```
BString Bds::ListRange::getType ( )
```


7.71.3.2 setMembers()

```
BError Bds::ListRange::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.71.3.3 setMember()

```
BError Bds::ListRange::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.71.3.4 getMembers()

```
BError Bds::ListRange::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.71.3.5 getMember()

```
BError Bds::ListRange::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.71.4 Member Data Documentation

7.71.4.1 start

```
BUInt32 Bds::ListRange::start
```

The start position.

7.71.4.2 number

```
BUInt32 Bds::ListRange::number
```

The number of items.

7.71.4.3 reverse

```
BInt32 Bds::ListRange::reverse
```

List from end.

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

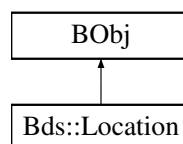
- [BdsD.h](#)
- [BdsD.cc](#)

7.72 Bds::Location Class Reference

This class defines the physical location of a [Station](#).

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Location:



Public Member Functions

- [Location](#) ([BUInt32](#) id=0, [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTime](#)↔[Stamp](#)(), [BString](#) network= [BString](#)(), [BString](#) station= [BString](#)(), [BString](#) channel= [BString](#)(), [BString](#) datum= [BString](#)(), [BFloat64](#) longitude=0, [BFloat64](#) latitude=0, [BFloat64](#) elevation=0, [BFloat64](#) arrayOffsetEast=0, [BFloat64](#) arrayOffsetNorth=0)
- [BString](#) getType ()
- [BError](#) setMembers ([BDictString](#) &members)
- [BError](#) setMember ([BString](#) name, [BString](#) value)
- [BError](#) getMembers ([BDictString](#) &members)
- [BError](#) getMember ([BString](#) name, [BString](#) &value)

Public Attributes

- **BUInt32** [id](#)
The ID.
- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time the channel was available.
- **BString** [network](#)
The Network/Organisation Name.
- **BString** [station](#)
The station this location is for.
- **BString** [channel](#)
The channel this location is for. If blank the location is for the [Station](#) itself.
- **BString** [datum](#)
The locations Datum.
- **BFloat64** [longitude](#)
The longitude in degrees using the WGS84 datum.
- **BFloat64** [latitude](#)
The Latitude in degrees using the WGS84 datum.
- **BFloat64** [elevation](#)
The ground level elevation in meters from the WGS84 ellipsoid (Sea level)
- **BFloat64** [arrayOffsetEast](#)
The Array offset in an array in an easterly direction if array known/given. [Note](#) returns Array's [ArrayChannel](#) offset information.
- **BFloat64** [arrayOffsetNorth](#)
The Array offset in an array in a northerly direction if array known/given. [Note](#) returns Array's [ArrayChannel](#) offset information.

7.72.1 Detailed Description

This class defines the physical location of a [Station](#).

This defines the physical location of the station using WGS84 longitude and latitude parameters. It also defines the stations elevation. If it is part of an array the arrayOffsetEast and arrayOffsetNorth will be filled in from the Array's [ArrayChannel](#) offset information. So although arrayOffsetEast and arrayOffsetNorth are part of a [Location](#) object they are not stored in the database StationLocations table.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 Location()

```
Bds::Location::Location (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString datum = BString(),
    BFloat64 longitude = 0,
    BFloat64 latitude = 0,
    BFloat64 elevation = 0,
    BFloat64 arrayOffsetEast = 0,
    BFloat64 arrayOffsetNorth = 0 )
```

7.72.3 Member Function Documentation

7.72.3.1 getType()

```
BString Bds::Location::getType ( )
```

7.72.3.2 setMembers()

```
BError Bds::Location::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.72.3.3 setMember()

```
BError Bds::Location::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.72.3.4 getMembers()

```
BError Bds::Location::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.72.3.5 getMember()

```
BError Bds::Location::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.72.4 Member Data Documentation

7.72.4.1 id

```
BUInt32 Bds::Location::id
```

The ID.

7.72.4.2 startTime

```
BTimeStamp Bds::Location::startTime
```

The Start Time.

7.72.4.3 endTime

```
BTimeStamp Bds::Location::endTime
```

The End Time the channel was available.

7.72.4.4 network

```
BString Bds::Location::network
```

The Network/Organisation Name.

7.72.4.5 station

```
BString Bds::Location::station
```

The station this location is for.

7.72.4.6 channel

```
BString Bds::Location::channel
```

The channel this location is for. If blank the location is for the [Station](#) itself.

7.72.4.7 datum

```
BString Bds::Location::datum
```

The locations Datum.

7.72.4.8 longitude

```
BFloat64 Bds::Location::longitude
```

The longitude in degrees using the WGS84 datum.

7.72.4.9 latitude

```
BFloat64 Bds::Location::latitude
```

The Latitude in degrees using the WGS84 datum.

7.72.4.10 elevation

```
BFloat64 Bds::Location::elevation
```

The ground level elevation in meters from the WGS84 ellipsoid (Sea level)

7.72.4.11 arrayOffsetEast

BFloat64 Bds::Location::arrayOffsetEast

The Array offset in an array in an easterly direction if array known/given. [Note](#) returns Array's [ArrayChannel](#) offset information.

7.72.4.12 arrayOffsetNorth

BFloat64 Bds::Location::arrayOffsetNorth

The Array offset in an array in a northerly direction if array known/given. [Note](#) returns Array's [ArrayChannel](#) offset information.

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

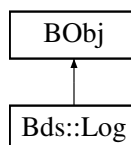
- [BdsD.h](#)
- [BdsD.cc](#)

7.73 Bds::Log Class Reference

This holds information on a [Log](#) entry.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Log:



Public Member Functions

- [Log](#) ([BUInt32](#) id=0, [BTimeStamp](#) time= [BTimeStamp](#)(), [BString](#) type= [BString](#)(), [BUInt32](#) priority=0, [BString](#) subSystem= [BString](#)(), [BString](#) title= [BString](#)(), [BString](#) description= [BString](#)())
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- **BUInt32** [id](#)
The unique id.
- **BTimeStamp** [time](#)
The Time.
- **BString** [type](#)
The Type.
- **BUInt32** [priority](#)
The priority 0 to 5.
- **BString** [subSystem](#)
The SubSystem.
- **BString** [title](#)
The Changes title.
- **BString** [description](#)
The Description of the change.

7.73.1 Detailed Description

This holds information on a [Log](#) entry.

[Log](#) entries are added automatically and manually to the system. A system administrator can view these logs.

7.73.2 Constructor & Destructor Documentation

7.73.2.1 Log()

```
Bds::Log::Log (
    BUInt32 id = 0,
    BTimeStamp time = BTimeStamp(),
    BString type = BString(),
    BUInt32 priority = 0,
    BString subSystem = BString(),
    BString title = BString(),
    BString description = BString() )
```

7.73.3 Member Function Documentation

7.73.3.1 getType()

```
BString Bds::Log::getType ( )
```


7.73.3.2 setMembers()

```
BError Bds::Log::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.73.3.3 setMember()

```
BError Bds::Log::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.73.3.4 getMembers()

```
BError Bds::Log::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.73.3.5 getMember()

```
BError Bds::Log::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.73.4 Member Data Documentation

7.73.4.1 id

```
BUInt32 Bds::Log::id
```

The unique id.

7.73.4.2 time

BTimeStamp Bds::Log::time

The Time.

7.73.4.3 type

BString Bds::Log::type

The Type.

7.73.4.4 priority

BUInt32 Bds::Log::priority

The priority 0 to 5.

7.73.4.5 subSystem

BString Bds::Log::subSystem

The SubSystem.

7.73.4.6 title

BString Bds::Log::title

The Changes title.

7.73.4.7 description

BString Bds::Log::description

The Description of the change.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.74 Bds::LogSelect Class Reference

This defines the selection cirteria when requesting a set of log entries.

```
#include <BdsD.h>
```

Public Member Functions

- **LogSelect** (**BTimeStamp** *startTime*= **BTimeStamp**(), **BString** *type*= **BString**(), **BUInt32** *priority*=0, **BString** *subSystem*= **BString**())

Public Attributes

- **BTimeStamp** *startTime*
The start time.
- **BString** *type*
The Type.
- **BUInt32** *priority*
The priority 0 to 5. Will return log entries at or above the priority given.
- **BString** *subSystem*
The SubSystem.

7.74.1 Detailed Description

This defines the selection cirteria when requesting a set of log entries.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 LogSelect()

```
Bds::LogSelect::LogSelect (
    BTimeStamp startTime = BTimeStamp(),
    BString type = BString(),
    BUInt32 priority = 0,
    BString subSystem = BString() )
```

7.74.3 Member Data Documentation

7.74.3.1 startTime

BTimeStamp Bds::LogSelect::startTime

The start time.

7.74.3.2 type

BString Bds::LogSelect::type

The Type.

7.74.3.3 priority

BUInt32 Bds::LogSelect::priority

The priority 0 to 5. Will return log entries at or above the priority given.

7.74.3.4 subSystem

BString Bds::LogSelect::subSystem

The SubSystem.

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

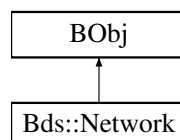
- [BdsD.h](#)
- [BdsD.cc](#)

7.75 Bds::Network Class Reference

This class defines a seismic [Network](#) organisation.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Network:



Public Member Functions

- [Network](#) ([BUInt32](#) id=0, [BString](#) network= [BString](#)(), [BString](#) description= [BString](#)(), [BList](#)< [BString](#) > stations= [BList](#)< [BString](#) >())
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) id
Unique ID when stored in a database or for other uses.
- [BString](#) network
The name.
- [BString](#) description
The organisations description.
- [BList](#)< [BString](#) > stations
The list of arrays/stations the [Network](#) uses. (deprecated)

7.75.1 Detailed Description

This class defines a seismic [Network](#) organisation.

Typical Seismic Networks are "BN", IDC" etc. The stations list is depreciated.

7.75.2 Constructor & Destructor Documentation

7.75.2.1 Network()

```
Bds::Network::Network (
    BUInt32 id = 0,
    BString network = BString(),
    BString description = BString(),
    BList< BString > stations = BList< BString >() )
```

7.75.3 Member Function Documentation

7.75.3.1 getType()

```
BString Bds::Network::getType ( )
```

7.75.3.2 setMembers()

```
BError Bds::Network::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.75.3.3 setMember()

```
BError Bds::Network::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.75.3.4 getMembers()

```
BError Bds::Network::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.75.3.5 getMember()

```
BError Bds::Network::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.75.4 Member Data Documentation

7.75.4.1 id

```
BUInt32 Bds::Network::id
```

Unique ID when stored in a database or for other uses.

7.75.4.2 network

BString Bds::Network::network

The name.

7.75.4.3 description

BString Bds::Network::description

The organisations description.

7.75.4.4 stations

BList< BString > Bds::Network::stations

The list of arrays/stations the [Network](#) uses. (deprecated)

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

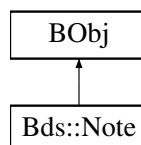
- [BdsD.h](#)
- [BdsD.cc](#)

7.76 Bds::Note Class Reference

This holds information on a [Note](#) for general information.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Note:



Public Member Functions

- [Note](#) (**BUInt32** id=0, **BTimeStamp** startTime= **BTimeStamp**(), **BTimeStamp** endTime= **BTimeStamp**(), **BString** network= **BString**(), **BString** station= **BString**(), **BString** channel= **BString**(), **BString** source= **BString**(), **BString** type= **BString**(), **BString** user= **BString**(), **BTimeStamp** timeAdded= **BTimeStamp**(), **BInt32** errorNumber=0, **BString** title= **BString**(), **BString** description= **BString**(), **BString** docFormat= **BString**(), **BString** docUrl= **BString**(), **BUInt32** dataField=0, **BString** importFilename= **BString**(), **BUInt32** eventId=0)
- **BString** getType ()
- **BError** setMembers (**BDictString** &members)
- **BError** setMember (**BString** name, **BString** value)
- **BError** getMembers (**BDictString** &members)
- **BError** getMember (**BString** name, **BString** &value)

Public Attributes

- **BUInt32** [id](#)
The unique id.
- **BTimeStamp** [startTime](#)
The Start Time note is for.
- **BTimeStamp** [endTime](#)
The End Time note is for.
- **BString** [network](#)
The [Network](#) Name.
- **BString** [station](#)
The Station/Array name.
- **BString** [channel](#)
The Channels name.
- **BString** [source](#)
The Data [Source](#).
- **BString** [type](#)
The Type (note, warning, error ...)
- **BString** [user](#)
The user.
- **BTimeStamp** [timeAdded](#)
The Time Entered.
- **BInt32** [errorNumber](#)
Error number if error.
- **BString** [title](#)
The title.
- **BString** [description](#)
The Description.
- **BString** [docFormat](#)
Document format if any.
- **BString** [docUrl](#)
Document Url if any.
- **BUInt32** [dataFileId](#)
The data file id associated with this note.
- **BString** [importFilename](#)
The import filename.
- **BUInt32** [eventId](#)
The event ID associated with this note.

7.76.1 Detailed Description

This holds information on a [Note](#) for general information.

Normally a [Note](#) can be added for a particular set of data over a particular time period. These notes are sometimes added automatically during a data import process or manually by a user. A [User](#) can then ask for any notes for a particular set of data.

7.76.2 Constructor & Destructor Documentation

7.76.2.1 Note()

```
Bds::Note::Note (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString(),
    BString type = BString(),
    BString user = BString(),
    BTimeStamp timeAdded = BTimeStamp(),
    BInt32 errorNumber = 0,
    BString title = BString(),
    BString description = BString(),
    BString docFormat = BString(),
    BString docUrl = BString(),
    BUInt32 dataFileId = 0,
    BString importFilename = BString(),
    BUInt32 eventId = 0 )
```

7.76.3 Member Function Documentation

7.76.3.1 getType()

```
BString Bds::Note::getType ( )
```

7.76.3.2 setMembers()

```
BError Bds::Note::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.76.3.3 setMember()

```
BError Bds::Note::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.76.3.4 getMembers()

```
BError Bds::Note::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.76.3.5 getMember()

```
BError Bds::Note::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.76.4 Member Data Documentation

7.76.4.1 id

```
BUInt32 Bds::Note::id
```

The unique id.

7.76.4.2 startTime

```
BTimeStamp Bds::Note::startTime
```

The Start Time note is for.

7.76.4.3 endTime

```
BTimeStamp Bds::Note::endTime
```

The End Time note is for.

7.76.4.4 network

BString Bds::Note::network

The [Network](#) Name.

7.76.4.5 station

BString Bds::Note::station

The Station/Array name.

7.76.4.6 channel

BString Bds::Note::channel

The Channels name.

7.76.4.7 source

BString Bds::Note::source

The Data [Source](#).

7.76.4.8 type

BString Bds::Note::type

The Type (note, warning, error ...)

7.76.4.9 user

BString Bds::Note::user

The user.

7.76.4.10 timeAdded

BTimeStamp Bds::Note::timeAdded

The Time Entered.

7.76.4.11 errorNumber

Int32 Bds::Note::errorNumber

Error number if error.

7.76.4.12 title

BString Bds::Note::title

The title.

7.76.4.13 description

BString Bds::Note::description

The Description.

7.76.4.14 docFormat

BString Bds::Note::docFormat

Document format if any.

7.76.4.15 docUrl

BString Bds::Note::docUrl

Document Url if any.

7.76.4.16 dataFileId

BUInt32 Bds::Note::dataFileId

The data file id associated with this note.

7.76.4.17 importFilename

BString Bds::Note::importFilename

The import filename.

7.76.4.18 eventId

BUInt32 Bds::Note::eventId

The event ID associated with this note.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.77 Bds::Point Class Reference

This class defines an X,Y location.

```
#include <BdsD.h>
```

Public Member Functions

- [Point](#) ([BFloat64](#) x=0, [BFloat64](#) y=0)

Public Attributes

- [BFloat64](#) x
- [BFloat64](#) y

7.77.1 Detailed Description

This class defines an X,Y location.

The class simply stores the x and y point values.

7.77.2 Constructor & Destructor Documentation

7.77.2.1 Point()

```
Bds::Point::Point (
    BFloat64 x = 0,
    BFloat64 y = 0 )
```

7.77.3 Member Data Documentation

7.77.3.1 x

```
BFloat64 Bds::Point::x
```

7.77.3.2 y

```
BFloat64 Bds::Point::y
```

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.78 Bds::PoleZero Class Reference

This class defines a Pole/Zero [Response](#).

```
#include <BdsD.h>
```

Public Member Functions

- [PoleZero](#) ([BArray](#)< [BComplex](#) > [poles](#)= [BArray](#)< [BComplex](#) >(), [BArray](#)< [BComplex](#) > [zeros](#)= [BArray](#)< [BComplex](#) >())

Public Attributes

- [BArray](#)< [BComplex](#) > [poles](#)
Poles.
- [BArray](#)< [BComplex](#) > [zeros](#)
Zeros.

7.78.1 Detailed Description

This class defines a Pole/Zero [Response](#).

It consists of an array of Complex poles and an array of Complex zeros.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 PoleZero()

```
Bds::PoleZero::PoleZero (
    BArray< BComplex > poles = BArray< BComplex > (),
    BArray< BComplex > zeros = BArray< BComplex > () )
```

7.78.3 Member Data Documentation

7.78.3.1 poles

```
BArray< BComplex > Bds::PoleZero::poles
```

Poles.

7.78.3.2 zeros

```
BArray< BComplex > Bds::PoleZero::zeros
```

Zeros.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.79 Bds::Polynomial Class Reference

This class defines an [Polynomial](#) response table.

```
#include <BdsD.h>
```

Public Member Functions

- **Polynomial** (**BString** transferType= **BString**(), **BString** approximationType= **BString**(), **BString** validFrequencyUnits= **BString**(), **BFloat64** frequencyLowerBound=0, **BFloat64** frequencyUpperBound=0, **BFloat64** approximationLowerBound=0, **BFloat64** approximationUpperBound=0, **BFloat64** maximumError=0, **BArray**< **PolynomialEntry** > coefficients= **BArray**< **PolynomialEntry** >())

Public Attributes

- **BString** transferType
Transfer Type.
- **BString** approximationType
Polynomial Approximation Type.
- **BString** validFrequencyUnits
Valid Frequency Units.
- **BFloat64** frequencyLowerBound
Lower Valid Frequency Bound.
- **BFloat64** frequencyUpperBound
Upper Valid Frequency Bound.
- **BFloat64** approximationLowerBound
Lower Bound of Approximation.
- **BFloat64** approximationUpperBound
Upper Bound of Approximation.
- **BFloat64** maximumError
Maximum Absolute Error.
- **BArray**< **PolynomialEntry** > coefficients
The Coefficients.

7.79.1 Detailed Description

This class defines an **Polynomial** response table.

This has an array of the polynomial coefficients.

7.79.2 Constructor & Destructor Documentation

7.79.2.1 Polynomial()

```
Bds::Polynomial::Polynomial (
    BString transferType = BString(),
    BString approximationType = BString(),
    BString validFrequencyUnits = BString(),
    BFloat64 frequencyLowerBound = 0,
    BFloat64 frequencyUpperBound = 0,
    BFloat64 approximationLowerBound = 0,
    BFloat64 approximationUpperBound = 0,
    BFloat64 maximumError = 0,
    BArray< PolynomialEntry > coefficients = BArray<PolynomialEntry >() )
```


7.79.3 Member Data Documentation

7.79.3.1 transferType

BString Bds::Polynomial::transferType

Transfer Type.

7.79.3.2 approximationType

BString Bds::Polynomial::approximationType

[Polynomial](#) Approximation Type.

7.79.3.3 validFrequencyUnits

BString Bds::Polynomial::validFrequencyUnits

Valid Frequency Units.

7.79.3.4 frequencyLowerBound

BFloat64 Bds::Polynomial::frequencyLowerBound

Lower Valid Frequency Bound.

7.79.3.5 frequencyUpperBound

BFloat64 Bds::Polynomial::frequencyUpperBound

Upper Valid Frequency Bound.

7.79.3.6 approximationLowerBound

BFloat64 Bds::Polynomial::approximationLowerBound

Lower Bound of Approximation.

7.79.3.7 approximationUpperBound

BFloat64 Bds::Polynomial::approximationUpperBound

Upper Bound of Approximation.

7.79.3.8 maximumError

BFloat64 Bds::Polynomial::maximumError

Maximum Absolute Error.

7.79.3.9 coefficients

BArray<[PolynomialEntry](#) > Bds::Polynomial::coefficients

The Coefficients.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.80 Bds::PolynomialEntry Class Reference

This class defines an entry in a [Polynomial](#) coefficient table.

```
#include <BdsD.h>
```

Public Member Functions

- [PolynomialEntry](#) (**BFloat64** [coefficient](#)=0, **BFloat64** [plusError](#)=0, **BFloat64** [minusError](#)=0, **BString** [measurementMethod](#)=**BString**())

Public Attributes

- **BFloat64** [coefficient](#)
The coefficient.
- **BFloat64** [plusError](#)
Plus uncertainty or error in measured value.
- **BFloat64** [minusError](#)
Minus uncertainty or error in measured value.
- **BString** [measurementMethod](#)
The measurement method.

7.80.1 Detailed Description

This class defines an entry in a [Polynomial](#) coefficient table.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 PolynomialEntry()

```
Bds::PolynomialEntry::PolynomialEntry (
    BFloat64 coefficient = 0,
    BFloat64 plusError = 0,
    BFloat64 minusError = 0,
    BString measurementMethod = BString() )
```

7.80.3 Member Data Documentation

7.80.3.1 coefficient

```
BFloat64 Bds::PolynomialEntry::coefficient
```

The coefficient.

7.80.3.2 plusError

```
BFloat64 Bds::PolynomialEntry::plusError
```

Plus uncertainty or error in measured value.

7.80.3.3 minusError

BFloat64 Bds::PolynomialEntry::minusError

Minus uncertainty or error in measured value.

7.80.3.4 measurementMethod

BString Bds::PolynomialEntry::measurementMethod

The measurement method.

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

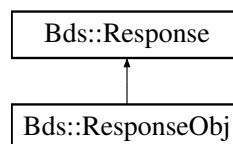
- [BdsD.h](#)
- [BdsD.cc](#)

7.81 Bds::Response Class Reference

This class defines a seismic [Response](#) characteristic.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Response:



Public Member Functions

- [Response](#) (**BUInt32** id=0, **BTimeStamp** startTime= **BTimeStamp**(), **BTimeStamp** endTime= **BTime**↔**Stamp**(), **BString** network= **BString**(), **BString** station= **BString**(), **BString** channel= **BString**(), **BString** source= **BString**(), **BUInt32** stage=0, **BString** name= **BString**(), **BString** type= **BString**(), **PoleZero** poleZeros=**PoleZero**(), **BArray**< **Fap** > faps= **BArray**< **Fap** >(), **Fir** fir=**Fir**(), **Polynomial** polynomial=**Polynomial**(), **BFloat64** gain=0, **BFloat64** gainFrequency=0, **BString** stageType= **BString**(), **BFloat64** decimation=0, **BFloat64** decimationOffset=0, **BFloat64** decimationDelay=0, **BFloat64** decimationCorr=0, **BString** symmetry= **BString**(), **BString** description= **BString**(), **BInt32** measured=0, **BFloat64** sampleRate=0, **BString** inputUnits= **BString**(), **BString** inputUnitsDesc= **BString**(), **BString** outputUnits= **BString**(), **BString** outputUnitsDesc= **BString**())

Public Attributes

- **BUInt32** [id](#)
The ID.
- **BTimeStamp** [startTime](#)
The start time when ret response is valid from.
- **BTimeStamp** [endTime](#)
The end time the response is valid to.
- **BString** [network](#)
The Network/Organisation Name.
- **BString** [station](#)
The station.
- **BString** [channel](#)
The channel.
- **BString** [source](#)
The source.
- **BUInt32** [stage](#)
The stage (0, 1, 2, 3, ...). 0 is a special overall response sometimes used.
- **BString** [name](#)
The response name. (Anti-Aliasing filter, [Digitiser](#), post filter etc)
- **BString** [type](#)
The type of response ([PoleZero](#), FIR, FAP, [Polynomial](#) or blank if no frequency response)
- [PoleZero](#) [poleZeros](#)
The PoleZeros.
- **BArray**< [Fap](#) > [faps](#)
The FAP Frequency/Amplitude/Phase table.
- [Fir](#) [fir](#)
The FIR filters coefficients.
- [Polynomial](#) [polynomial](#)
The polynomial response.
- **BFloat64** [gain](#)
Overall gain at [gainFrequency](#).
- **BFloat64** [gainFrequency](#)
Frequency that gain is valid for.
- **BString** [stageType](#)
The stage type: A - Analog (rad/sec), B - Analog (Hz), C - Composite, D - Digital.
- **BFloat64** [decimation](#)
Decimation factor performed post filter.
- **BFloat64** [decimationOffset](#)
Decimation sample offset.
- **BFloat64** [decimationDelay](#)
Decimation delay in seconds.
- **BFloat64** [decimationCorr](#)
Decimation correction performed.
- **BString** [symmetry](#)
Symmetry for FIR coefficients (A = asymmetric, B = symmetric[odd], C = symmetric[even])
- **BString** [description](#)
Misc description.
- **BInt32** [measured](#)
Boolean if response was a measured response.
- **BFloat64** [sampleRate](#)

- The stage's sample rate.*
- **BString** [inputUnits](#)

The units of the data as input from the perspective of data acquisition. After correcting data for this response, these would be the resulting units.
- **BString** [inputUnitsDesc](#)

The input units description.
- **BString** [outputUnits](#)

The units of the data as output from the perspective of data acquisition. These would be the units of the data prior to correcting for this response.
- **BString** [outputUnitsDesc](#)

The output units description.

7.81.1 Detailed Description

This class defines a seismic [Response](#) characteristic.

For each seismic channel there is a frequency response characteristic. There can be multiple stages in a channels frequency response, this response data describes one of those stages frequencies responses. The stage parameter defines which stage it is for (1, 2, 3, ...) Stage 0 is reserved to store an overall channel response for special puposes. A response can be in the form of an array of poles and zeros, a FAP array, a set of FIR coefficients or a [Polynomial](#). This object contains Metadata for other response characteristics as defined in various seismic [Response](#) formats. In general this extra Metadata is simply passed through the BDS and is used when exporting Metadat in different formats. **Note** that the actual response data is stored in a database Blob and the poleZeros, faps, fir or polynomial fields are filled in based upon the [Response](#) type.

7.81.2 Constructor & Destructor Documentation

7.81.2.1 Response()

```
Bds::Response::Response (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString(),
    BUInt32 stage = 0,
    BString name = BString(),
    BString type = BString(),
    PoleZero poleZeros = PoleZero(),
    BArray< Fap > faps = BArray<Fap >(),
    Fir fir = Fir(),
    Polynomial polynomial = Polynomial(),
    BFloat64 gain = 0,
    BFloat64 gainFrequency = 0,
    BString stageType = BString(),
    BFloat64 decimation = 0,
    BFloat64 decimationOffset = 0,
    BFloat64 decimationDelay = 0,
```

```
BFloat64 decimationCorr = 0,  
BString symmetry = BString(),  
BString description = BString(),  
BInt32 measured = 0,  
BFloat64 sampleRate = 0,  
BString inputUnits = BString(),  
BString inputUnitsDesc = BString(),  
BString outputUnits = BString(),  
BString outputUnitsDesc = BString() )
```

7.81.3 Member Data Documentation

7.81.3.1 id

```
BUInt32 Bds::Response::id
```

The ID.

7.81.3.2 startTime

```
BTimeStamp Bds::Response::startTime
```

The start time when ret response is valid from.

7.81.3.3 endTime

```
BTimeStamp Bds::Response::endTime
```

The end time the response is valid to.

7.81.3.4 network

```
BString Bds::Response::network
```

The Network/Organisation Name.

7.81.3.5 station

BString Bds::Response::station

The station.

7.81.3.6 channel

BString Bds::Response::channel

The channel.

7.81.3.7 source

BString Bds::Response::source

The source.

7.81.3.8 stage

BUInt32 Bds::Response::stage

The stage (0, 1, 2, 3, ...). 0 is a special overall response sometimes used.

7.81.3.9 name

BString Bds::Response::name

The response name. (Anti-Aliasing filter, [Digitiser](#), post filter etc)

7.81.3.10 type

BString Bds::Response::type

The type of response ([PoleZero](#), FIR, FAP, [Polynomial](#) or blank if no frequency response)

7.81.3.11 poleZeros

`PoleZero` Bds::Response::poleZeros

The PoleZeros.

7.81.3.12 faps

`BArray<Fap >` Bds::Response::faps

The FAP Frequency/Amplitude/Phase table.

7.81.3.13 fir

`Fir` Bds::Response::fir

The FIR filters coefficients.

7.81.3.14 polynomial

`Polynomial` Bds::Response::polynomial

The polynomial response.

7.81.3.15 gain

`BFloat64` Bds::Response::gain

Overall gain at gainFrequency.

7.81.3.16 gainFrequency

`BFloat64` Bds::Response::gainFrequency

Frequency that gain is valid for.

7.81.3.17 stageType

BString Bds::Response::stageType

The stage type: A - Analog (rad/sec), B - Analog (Hz), C - Composite, D - Digital.

7.81.3.18 decimation

BFloat64 Bds::Response::decimation

Decimation factor performed post filter.

7.81.3.19 decimationOffset

BFloat64 Bds::Response::decimationOffset

Decimation sample offset.

7.81.3.20 decimationDelay

BFloat64 Bds::Response::decimationDelay

Decimation delay in seconds.

7.81.3.21 decimationCorr

BFloat64 Bds::Response::decimationCorr

Decimation correction performed.

7.81.3.22 symmetry

BString Bds::Response::symmetry

Symmetry for FIR coefficients (A = asymmetric, B = symmetric[odd], C = symmetric[even])

7.81.3.23 description

BString Bds::Response::description

Misc description.

7.81.3.24 measured

Int32 Bds::Response::measured

Boolean if response was a measured response.

7.81.3.25 sampleRate

Float64 Bds::Response::sampleRate

The stage's sample rate.

7.81.3.26 inputUnits

BString Bds::Response::inputUnits

The units of the data as input from the perspective of data acquisition. After correcting data for this response, these would be the resulting units.

7.81.3.27 inputUnitsDesc

BString Bds::Response::inputUnitsDesc

The input units description.

7.81.3.28 outputUnits

BString Bds::Response::outputUnits

The units of the data as output from the perspective of data acquisition. These would be the units of the data prior to correcting for this response.

7.81.3.29 outputUnitsDesc

BString Bds::Response::outputUnitsDesc

The output units description.

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

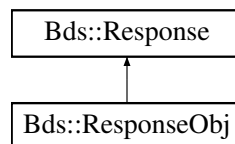
- [BdsD.h](#)
- [BdsD.cc](#)

7.82 Bds::ResponseObj Class Reference

[Response](#) object adding string conversion.

```
#include <BdsLib.h>
```

Inheritance diagram for Bds::ResponseObj:



Public Member Functions

- [ResponseObj](#) (const [Response](#) &response)
- [~ResponseObj](#) ()
- **BString** [getString](#) ()
- void [setString](#) (**BString** str)

Additional Inherited Members

7.82.1 Detailed Description

[Response](#) object adding string conversion.

7.82.2 Constructor & Destructor Documentation

7.82.2.1 ResponseObj()

```
Bds::ResponseObj::ResponseObj (
    const Response & response )
```

7.82.2.2 ~ResponseObj()

```
Bds::ResponseObj::~~ResponseObj ( )
```

7.82.3 Member Function Documentation

7.82.3.1 getString()

```
BString Bds::ResponseObj::getString ( )
```

7.82.3.2 setString()

```
void Bds::ResponseObj::setString (
    BString str )
```

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

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

7.83 Bds::Selection Class Reference

This class defines a generic Metadata or [Sensor](#) data selection.

```
#include <BdsD.h>
```

Public Member Functions

- [Selection](#) ([BUInt32](#) id=0, [ListRange](#) range=[ListRange](#)(), [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTimeStamp](#)(), [BList](#)< [SelectionChannel](#) > channels= [BList](#)< [SelectionChannel](#) >(), [BUInt32](#) channelId=0, [BUInt32](#) digitiserId=0, [BUInt32](#) sensorId=0, [BUInt32](#) sensorOldId=0, [BInt32](#) completeSegments=0, [BString](#) calibrationName= [BString](#)(), [BString](#) array= [BString](#)(), [BUInt32](#) eventId=0, [BString](#) name= [BString](#)(), [LocationSelect](#) locationSelect=[LocationSelect](#)(), [BString](#) dataTypes= [BString](#)(), [BString](#) excludeChannels= [BString](#)())

Public Attributes

- **BUInt32** [id](#)
The ID of the record to return.
- **ListRange** [range](#)
The range of data to return.
- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time.
- **BList**< [SelectionChannel](#) > [channels](#)
The data channels to select.
- **BUInt32** [channelId](#)
The [Channel](#) id.
- **BUInt32** [digitiserId](#)
The [Digitiser](#) id.
- **BUInt32** [sensorId](#)
The [Sensor](#) id.
- **BUInt32** [sensorOldId](#)
The [Sensor](#) old id.
- **BInt32** [completeSegments](#)
Do not clip the segment times to match the required time period.
- **BString** [calibrationName](#)
[Calibration](#) name to use.
- **BString** [array](#)
Channels are based on the given array (for array offsets)
- **BUInt32** [eventId](#)
A particular event's ID.
- **BString** [name](#)
Match the name, title or some other string in the objects to select.
- **LocationSelect** [locationSelect](#)
Which locations to select, those for stations, channels or all of them when used in a [locationGetList\(\)](#) call.
- **BString** [dataTypes](#)
Place holder as yet for: A comma separated list of [Channel](#) [dataTypes](#) to select when returning Channels. If set to null all [Channel](#) [dataTypes](#) are returned.
- **BString** [excludeChannels](#)
Comma separated wildcard list of channels to be excluded.

7.83.1 Detailed Description

This class defines a generic Metadata or [Sensor](#) data selection.

This defines a set of selection criteria when selecting items from the BDS Metadata or [Sensor](#) data sets. The fields, when set, limit the items returned as defined by the setting fields provided. In effect these have an "AND" type of function with the parameters provided. Most of the string parameters can be provided as regular expressions. The core section fields are the [startTime](#), [endTime](#) and [channels](#) fields. As well as these there are some specific parameters like: [channelId](#), [digitiserId](#), [sensorId](#) etc which are used with specific data selection functions.

7.83.2 Constructor & Destructor Documentation

7.83.2.1 Selection()

```

Bds::Selection::Selection (
    BUInt32 id = 0,
    ListRange range = ListRange(),
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BList< SelectionChannel > channels = BList< SelectionChannel >(),
    BUInt32 channelId = 0,
    BUInt32 digitiserId = 0,
    BUInt32 sensorId = 0,
    BUInt32 sensorOldId = 0,
    BInt32 completeSegments = 0,
    BString calibrationName = BString(),
    BString array = BString(),
    BUInt32 eventId = 0,
    BString name = BString(),
    LocationSelect locationSelect = LocationSelect(),
    BString dataTypes = BString(),
    BString excludeChannels = BString() )

```

7.83.3 Member Data Documentation

7.83.3.1 id

BUInt32 Bds::Selection::id

The ID of the record to return.

7.83.3.2 range

ListRange Bds::Selection::range

The range of data to return.

7.83.3.3 startTime

BTimeStamp Bds::Selection::startTime

The Start Time.

7.83.3.4 endTime

`BTimeStamp Bds::Selection::endTime`

The End Time.

7.83.3.5 channels

`BList<SelectionChannel > Bds::Selection::channels`

The data channels to select.

7.83.3.6 channelId

`BUInt32 Bds::Selection::channelId`

The [Channel](#) id.

7.83.3.7 digitiserId

`BUInt32 Bds::Selection::digitiserId`

The [Digitiser](#) id.

7.83.3.8 sensorId

`BUInt32 Bds::Selection::sensorId`

The [Sensor](#) id.

7.83.3.9 sensorOldId

`BUInt32 Bds::Selection::sensorOldId`

The [Sensor](#) old id.

7.83.3.10 completeSegments

BInt32 Bds::Selection::completeSegments

Do not clip the segment times to match the required time period.

7.83.3.11 calibrationName

BString Bds::Selection::calibrationName

Calibration name to use.

7.83.3.12 array

BString Bds::Selection::array

Channels are based on the given array (for array offsets)

7.83.3.13 eventId

BUInt32 Bds::Selection::eventId

A particular event's ID.

7.83.3.14 name

BString Bds::Selection::name

Match the name, title or some other string in the objects to select.

7.83.3.15 locationSelect

LocationSelect Bds::Selection::locationSelect

Which locations to select, those for stations, channels or all of them when used in a locationGetList() call.

7.83.3.16 dataTypes

BString Bds::Selection::dataTypes

Place holder as yet for: A comma separated list of [Channel](#) dataTypes to select when returning Channels. If set to null all [Channel](#) dataTypes are returned.

7.83.3.17 excludeChannels

BString Bds::Selection::excludeChannels

Comma separated wildcard list of channels to be excluded.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.84 Bds::SelectionChannel Class Reference

This class defines an individual channel for selection.

```
#include <BdsD.h>
```

Public Member Functions

- [SelectionChannel](#) (**BString** network= **BString**(), **BString** station= **BString**(), **BString** channel= **BString**(), **BString** source= **BString**())

Public Attributes

- **BString** [network](#)
- **BString** [station](#)
- **BString** [channel](#)
- **BString** [source](#)

7.84.1 Detailed Description

This class defines an individual channel for selection.

It contains the network:station:channel:source names.

7.84.2 Constructor & Destructor Documentation

7.84.2.1 SelectionChannel()

```
Bds::SelectionChannel::SelectionChannel (
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString source = BString() )
```

7.84.3 Member Data Documentation

7.84.3.1 network

```
BString Bds::SelectionChannel::network
```

7.84.3.2 station

```
BString Bds::SelectionChannel::station
```

7.84.3.3 channel

```
BString Bds::SelectionChannel::channel
```

7.84.3.4 source

```
BString Bds::SelectionChannel::source
```

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.85 Bds::SelectionInfo Class Reference

This class defines the set of Metadata or Siesmic sensor data to be selected when `getSelectionInfo()` is use.

```
#include <BdsD.h>
```

Public Member Functions

- **SelectionInfo** (**BTimeStamp** *startTime*= **BTimeStamp**(), **BTimeStamp** *endTime*= **BTimeStamp**(), **BList**< **BString** > *networks*= **BList**< **BString** >(), **BList**< **BString** > *arrays*= **BList**< **BString** >(), **BList**< **BString** > *stations*= **BList**< **BString** >(), **BList**< **BString** > *arraysAndStations*= **BList**< **BString** >(), **BList**< **BString** > *channels*= **BList**< **BString** >(), **BList**< **BString** > *sources*= **BList**< **BString** >(), **BUInt32** *numDataChannels*=0)

Public Attributes

- **BTimeStamp** *startTime*
The Start Time.
- **BTimeStamp** *endTime*
The End Time.
- **BList**< **BString** > *networks*
The list of [Network](#) Names.
- **BList**< **BString** > *arrays*
The list of Array names.
- **BList**< **BString** > *stations*
The list of [Station](#) names.
- **BList**< **BString** > *arraysAndStations*
The list of Array and [Station](#) names.
- **BList**< **BString** > *channels*
The list of Channels.
- **BList**< **BString** > *sources*
The list of Data Sources.
- **BUInt32** *numDataChannels*
The number of sets of data in the system matching the criteria.

7.85.1 Detailed Description

This class defines the set of Metadata or Siesmic sensor data to be selected when `getSelectionInfo()` is use.

This provides information on everything selected by a [Selection](#) object from the BDS Metadata or [Sensor](#) data sets.

7.85.2 Constructor & Destructor Documentation

7.85.2.1 SelectionInfo()

```
Bds::SelectionInfo::SelectionInfo (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BList< BString > networks = BList< BString >(),
    BList< BString > arrays = BList< BString >(),
    BList< BString > stations = BList< BString >(),
    BList< BString > arraysAndStations = BList< BString >(),
    BList< BString > channels = BList< BString >(),
    BList< BString > sources = BList< BString >(),
    BUInt32 numDataChannels = 0 )
```

7.85.3 Member Data Documentation

7.85.3.1 startTime

BTimeStamp Bds::SelectionInfo::startTime

The Start Time.

7.85.3.2 endTime

BTimeStamp Bds::SelectionInfo::endTime

The End Time.

7.85.3.3 networks

BList< **BString** > Bds::SelectionInfo::networks

The list of [Network](#) Names.

7.85.3.4 arrays

BList< **BString** > Bds::SelectionInfo::arrays

The list of Array names.

7.85.3.5 stations

BList< **BString** > Bds::SelectionInfo::stations

The list of [Station](#) names.

7.85.3.6 arraysAndStations

```
BList< BString > Bds::SelectionInfo::arraysAndStations
```

The list of Array and [Station](#) names.

7.85.3.7 channels

```
BList< BString > Bds::SelectionInfo::channels
```

The list of Channels.

7.85.3.8 sources

```
BList< BString > Bds::SelectionInfo::sources
```

The list of Data Sources.

7.85.3.9 numDataChannels

```
BUInt32 Bds::SelectionInfo::numDataChannels
```

The number of sets of data in the system matching the criteria.

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

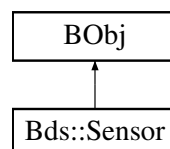
- [BdsD.h](#)
- [BdsD.cc](#)

7.86 Bds::Sensor Class Reference

This class defines a seismic [Sensor](#).

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Sensor:



Public Member Functions

- [Sensor](#) ([BUInt32](#) id=0, [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTimeStamp](#)(), [BString](#) name= [BString](#)(), [BString](#) type= [BString](#)(), [BString](#) serialNumber= [BString](#)(), [BUInt32](#) numberChannels=0, [BString](#) gainUnits= [BString](#)(), [BFloat64](#) gain=0, [BUInt32](#) oldId=0, [BInt32](#) shared=0)
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) [id](#)
The ID.
- [BTimeStamp](#) [startTime](#)
The Start Time.
- [BTimeStamp](#) [endTime](#)
The End Time.
- [BString](#) [name](#)
The Sensors name.
- [BString](#) [type](#)
The type of sensor. (Seismometer, Hydrophone etc)
- [BString](#) [serialNumber](#)
The sensor's serial number. Only used when there is a unique physical sensor.
- [BUInt32](#) [numberChannels](#)
The number of supported channels.
- [BString](#) [gainUnits](#)
The gain units.
- [BFloat64](#) [gain](#)
The overall gain of the sensor at the manufacturers calibration frequency. (For information only)
- [BUInt32](#) [oldId](#)
The Id from the old Autodrm database.
- [BInt32](#) [shared](#)
This sensor is shared.

7.86.1 Detailed Description

This class defines a seismic [Sensor](#).

This just stores information on the seismic instrument's sensor. Its contents is generally for information only.

7.86.2 Constructor & Destructor Documentation

7.86.2.1 Sensor()

```
Bds::Sensor::Sensor (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString name = BString(),
    BString type = BString(),
    BString serialNumber = BString(),
    BUInt32 numberChannels = 0,
    BString gainUnits = BString(),
    BFloat64 gain = 0,
    BUInt32 oldId = 0,
    BInt32 shared = 0 )
```

7.86.3 Member Function Documentation

7.86.3.1 getType()

```
BString Bds::Sensor::getType ( )
```

7.86.3.2 setMembers()

```
BError Bds::Sensor::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.86.3.3 setMember()

```
BError Bds::Sensor::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.86.3.4 getMembers()

```
BError Bds::Sensor::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.86.3.5 getMember()

```
BError Bds::Sensor::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.86.4 Member Data Documentation

7.86.4.1 id

```
BUInt32 Bds::Sensor::id
```

The ID.

7.86.4.2 startTime

```
BTimeStamp Bds::Sensor::startTime
```

The Start Time.

7.86.4.3 endTime

```
BTimeStamp Bds::Sensor::endTime
```

The End Time.

7.86.4.4 name

```
BString Bds::Sensor::name
```

The Sensors name.

7.86.4.5 type

BString Bds::Sensor::type

The type of sensor. (Seismometer, Hydrophone etc)

7.86.4.6 serialNumber

BString Bds::Sensor::serialNumber

The sensor's serial number. Only used when there is a unique physical sensor.

7.86.4.7 numberChannels

BUInt32 Bds::Sensor::numberChannels

The number of supported channels.

7.86.4.8 gainUnits

BString Bds::Sensor::gainUnits

The gain units.

7.86.4.9 gain

BFloat64 Bds::Sensor::gain

The overall gain of the sensor at the manufacturers calibration frequency. (For information only)

7.86.4.10 oldId

BUInt32 Bds::Sensor::oldId

The Id from the old Autodrm database.

7.86.4.11 shared

```
BInt32 Bds::Sensor::shared
```

This sensor is shared.

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

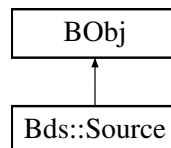
- [BdsD.h](#)
- [BdsD.cc](#)

7.87 Bds::Source Class Reference

This class defines a seismic data [Source](#).

```
#include <BdsD.h>
```

Inheritance diagram for Bds::Source:



Public Member Functions

- [Source](#) ([BUInt32](#) id=0, [BString](#) source= [BString](#)(), [BString](#) sourceMeta= [BString](#)(), [BString](#) alias= [BString](#)(), [BString](#) description= [BString](#)())
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- [BUInt32](#) [id](#)
Unique ID when stored in a database or for other uses.
- [BString](#) [source](#)
The sensor data's source name.
- [BString](#) [sourceMeta](#)
The associated metadata's source name.
- [BString](#) [alias](#)
The short alias for data files.
- [BString](#) [description](#)
The description.

7.87.1 Detailed Description

This class defines a seismic data [Source](#).

A Seismic data source allows different sources of data to be described and allows different Metadata sets to be used with the different data sources. It might be that there were two different digitisers in use on a [Channel](#) or one data set was received real-time though a particular data processing chain while the other was via CD medium with a different processing chain.

7.87.2 Constructor & Destructor Documentation

7.87.2.1 Source()

```
Bds::Source::Source (
    BUInt32 id = 0,
    BString source = BString(),
    BString sourceMeta = BString(),
    BString alias = BString(),
    BString description = BString() )
```

7.87.3 Member Function Documentation

7.87.3.1 getType()

```
BString Bds::Source::getType ( )
```

7.87.3.2 setMembers()

```
BError Bds::Source::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.87.3.3 setMember()

```
BError Bds::Source::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.87.3.4 getMembers()

```
BError Bds::Source::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.87.3.5 getMember()

```
BError Bds::Source::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.87.4 Member Data Documentation

7.87.4.1 id

```
BUInt32 Bds::Source::id
```

Unique ID when stored in a database or for other uses.

7.87.4.2 source

```
BString Bds::Source::source
```

The sensor data's source name.

7.87.4.3 sourceMeta

```
BString Bds::Source::sourceMeta
```

The associated metadata's source name.

7.87.4.4 alias

```
BString Bds::Source::alias
```

The short alias for data files.

7.87.4.5 description

```
BString Bds::Source::description
```

The description.

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

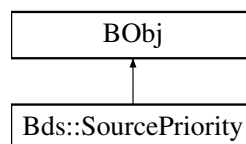
- [BdsD.h](#)
- [BdsD.cc](#)

7.88 Bds::SourcePriority Class Reference

This class defines a [Source](#) Priority entry.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::SourcePriority:



Public Member Functions

- [SourcePriority](#) ([BUInt32](#) id=0, [BTimeStamp](#) startTime= [BTimeStamp](#)(), [BTimeStamp](#) endTime= [BTime](#)↔
[Stamp](#)(), [BString](#) source= [BString](#)(), [BUInt32](#) priority=0)
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- **BUInt32** [id](#)
Unique ID when stored in a database or for other uses.
- **BTimeStamp** [startTime](#)
The Start Time.
- **BTimeStamp** [endTime](#)
The End Time the channel was available.
- **BString** [source](#)
The source name.
- **BUInt32** [priority](#)
The priority order, highest first.

7.88.1 Detailed Description

This class defines a [Source](#) Priority entry.

This allows the default source for data to be selected based on a priority level. It allows a particular source to be used if no other is available prioritised through all the different sources available.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 SourcePriority()

```
Bds::SourcePriority::SourcePriority (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString source = BString(),
    BUInt32 priority = 0 )
```

7.88.3 Member Function Documentation

7.88.3.1 getType()

```
BString Bds::SourcePriority::getType ( )
```

7.88.3.2 setMembers()

```
BError Bds::SourcePriority::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.88.3.3 setMember()

```
BError Bds::SourcePriority::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.88.3.4 getMembers()

```
BError Bds::SourcePriority::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.88.3.5 getMember()

```
BError Bds::SourcePriority::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.88.4 Member Data Documentation

7.88.4.1 id

```
BUInt32 Bds::SourcePriority::id
```

Unique ID when stored in a database or for other uses.

7.88.4.2 startTime

BTimeStamp Bds::SourcePriority::startTime

The Start Time.

7.88.4.3 endTime

BTimeStamp Bds::SourcePriority::endTime

The End Time the channel was available.

7.88.4.4 source

BString Bds::SourcePriority::source

The source name.

7.88.4.5 priority

BUInt32 Bds::SourcePriority::priority

The priority order, highest first.

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

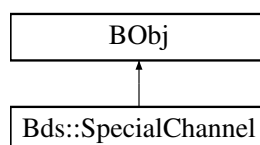
- [BdsD.h](#)
- [BdsD.cc](#)

7.89 Bds::SpecialChannel Class Reference

A Special channel identifier.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::SpecialChannel:



Public Member Functions

- **SpecialChannel** (**BUInt32** *id*=0, **BTimeStamp** *startTime*= **BTimeStamp**(), **BTimeStamp** *endTime*= **BTimeStamp**(), **BString** *network*= **BString**(), **BString** *station*= **BString**(), **BString** *channel*= **BString**(), **BString** *dataType*= **BString**(), **BString** *description*= **BString**())
- **BString** *getType* ()
- **BError** *setMembers* (**BDictString** &members)
- **BError** *setMember* (**BString** name, **BString** value)
- **BError** *getMembers* (**BDictString** &members)
- **BError** *getMember* (**BString** name, **BString** &value)

Public Attributes

- **BUInt32** *id*
Unique ID when stored in a database or for other uses.
- **BTimeStamp** *startTime*
The Start Time.
- **BTimeStamp** *endTime*
The End Time the channel was available.
- **BString** *network*
The Network Name, wildcards allowed.
- **BString** *station*
The Stations name, wildcards allowed.
- **BString** *channel*
The channels name, wildcards allowed (often as <channelType>_<channelAux>)
- **BString** *dataType*
The Type of channel (ignore)
- **BString** *description*
The channels description.

7.89.1 Detailed Description

A Special channel identifier.

This specifies a particular **Channel** as a special one. The Channels **Network**, **Station** and **Channel** are regular expresions to define one or more channels. If selected the *dataType* field is effectively applied to this channel. The *dataType* "ignore" allows Cahnnels to be ignored on export by default.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 SpecialChannel()

```
Bds::SpecialChannel::SpecialChannel (
    BUInt32 id = 0,
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp(),
    BString network = BString(),
    BString station = BString(),
    BString channel = BString(),
    BString dataType = BString(),
    BString description = BString() )
```

7.89.3 Member Function Documentation

7.89.3.1 getType()

```
BString Bds::SpecialChannel::getType ( )
```

7.89.3.2 setMembers()

```
BError Bds::SpecialChannel::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.89.3.3 setMember()

```
BError Bds::SpecialChannel::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.89.3.4 getMembers()

```
BError Bds::SpecialChannel::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.89.3.5 getMember()

```
BError Bds::SpecialChannel::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.89.4 Member Data Documentation

7.89.4.1 id

BUInt32 Bds::SpecialChannel::id

Unique ID when stored in a database or for other uses.

7.89.4.2 startTime

BTimeStamp Bds::SpecialChannel::startTime

The Start Time.

7.89.4.3 endTime

BTimeStamp Bds::SpecialChannel::endTime

The End Time the channel was available.

7.89.4.4 network

BString Bds::SpecialChannel::network

The [Network](#) Name, wildcards allowed.

7.89.4.5 station

BString Bds::SpecialChannel::station

The Stations name, wildcards allowed.

7.89.4.6 channel

BString Bds::SpecialChannel::channel

The channels name, wildcards allowed (often as <channelType>_<channelAux>)

7.89.4.7 dataType

BString Bds::SpecialChannel::dataType

The Type of channel (ignore)

7.89.4.8 description

BString Bds::SpecialChannel::description

The channels description.

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

- [BdsD.h](#)
- [BdsD.cc](#)

7.90 Bds::Station Class Reference

This class defines a seismic station.

```
#include <BdsD.h>
```

Public Member Functions

- [Station](#) ([BUInt32](#) id=0, [BString](#) network= [BString](#)(), [BString](#) name= [BString](#)(), [BString](#) alias= [BString](#)(), [BString](#) type= [BString](#)(), [BString](#) description= [BString](#)(), [BList](#)< [ArrayChannel](#) > channels= [BList](#)< [ArrayChannel](#) >())

Public Attributes

- [BUInt32](#) id
Unique ID when stored in a database or for other uses.
- [BString](#) network
The [Network](#) this station belongs to if for a partricular network.
- [BString](#) name
The name.
- [BString](#) alias
Alias name to be returned to the user.
- [BString](#) type
The [Station](#) type. Set to "array" or "station".
- [BString](#) description
Description.
- [BList](#)< [ArrayChannel](#) > channels
List of channels if an Array.

7.90.1 Detailed Description

This class defines a seismic station.

A [Station](#) could be an individual station or an array of stations. If it is an array of stations (type is set to "array") then the [Station](#) object contains the list of channels that make up the array. The list of channels contain [Location](#) offsets to the individual channels relative to the array's centre. This class defines a [Station](#)

7.90.2 Constructor & Destructor Documentation

7.90.2.1 Station()

```
Bds::Station::Station (
    BUInt32 id = 0,
    BString network = BString(),
    BString name = BString(),
    BString alias = BString(),
    BString type = BString(),
    BString description = BString(),
    BList< ArrayChannel > channels = BList<ArrayChannel >() )
```

7.90.3 Member Data Documentation

7.90.3.1 id

```
BUInt32 Bds::Station::id
```

Unique ID when stored in a database or for other uses.

7.90.3.2 network

```
BString Bds::Station::network
```

The [Network](#) this station belongs to if for a particular network.

7.90.3.3 name

```
BString Bds::Station::name
```

The name.

7.90.3.4 alias

```
BString Bds::Station::alias
```

Alias name to be returned to the user.

7.90.3.5 type

```
BString Bds::Station::type
```

The [Station](#) type. Set to "array" or "station".

7.90.3.6 description

```
BString Bds::Station::description
```

Description.

7.90.3.7 channels

```
BList<ArrayChannel > Bds::Station::channels
```

List of channels if an Array.

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

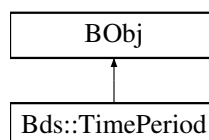
- [BdsD.h](#)
- [BdsD.cc](#)

7.91 Bds::TimePeriod Class Reference

This class defines a [TimePeriod](#).

```
#include <BdsD.h>
```

Inheritance diagram for Bds::TimePeriod:



Public Member Functions

- [TimePeriod](#) ([BTimeStamp](#) [startTime](#)= [BTimeStamp](#)(), [BTimeStamp](#) [endTime](#)= [BTimeStamp](#)())
- [BString](#) [getType](#) ()
- [BError](#) [setMembers](#) ([BDictString](#) &members)
- [BError](#) [setMember](#) ([BString](#) name, [BString](#) value)
- [BError](#) [getMembers](#) ([BDictString](#) &members)
- [BError](#) [getMember](#) ([BString](#) name, [BString](#) &value)

Public Attributes

- [BTimeStamp](#) [startTime](#)
The Start time to the nearest us.
- [BTimeStamp](#) [endTime](#)
The End time to the nearest us.

7.91.1 Detailed Description

This class defines a [TimePeriod](#).

It has [startTime](#) and [endTime](#) fields. [Note](#) the [endTime](#) is just after the actual period.

7.91.2 Constructor & Destructor Documentation

7.91.2.1 TimePeriod()

```
Bds::TimePeriod::TimePeriod (
    BTimeStamp startTime = BTimeStamp(),
    BTimeStamp endTime = BTimeStamp() )
```

7.91.3 Member Function Documentation

7.91.3.1 getType()

```
BString Bds::TimePeriod::getType ( )
```


7.91.3.2 setMembers()

```
BError Bds::TimePeriod::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.91.3.3 setMember()

```
BError Bds::TimePeriod::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.91.3.4 getMembers()

```
BError Bds::TimePeriod::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.91.3.5 getMember()

```
BError Bds::TimePeriod::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.91.4 Member Data Documentation

7.91.4.1 startTime

```
BTimeStamp Bds::TimePeriod::startTime
```

The Start time to the nearest us.

7.91.4.2 endTime

BTimeStamp Bds::TimePeriod::endTime

The End time to the nearest us.

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

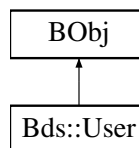
- [BdsD.h](#)
- [BdsD.cc](#)

7.92 Bds::User Class Reference

This holds information on a user.

```
#include <BdsD.h>
```

Inheritance diagram for Bds::User:



Public Member Functions

- **User** (BUInt32 id=0, BString user= BString(), BString password= BString(), BString name= BString(), BString email= BString(), BString telephone= BString(), BString address= BString(), BInt32 enabled=0, BList< BString > groups= BList< BString >())
- **BString** getType ()
- **BError** setMembers (BDictString &members)
- **BError** setMember (BString name, BString value)
- **BError** getMembers (BDictString &members)
- **BError** getMember (BString name, BString &value)

Public Attributes

- **BUInt32** id
The unique user ID.
- **BString** user
The User ID.
- **BString** password
The User's password.
- **BString** name
The User's full name.
- **BString** email
The User's email Address.
- **BString** telephone
The User's telephone number.
- **BString** address
The User's postal address.
- **BInt32** enabled
Whether the users account is enabled.
- **BList< BString >** groups
The security groups the user belongs to.

7.92.1 Detailed Description

This holds information on a user.

All information on a BDS user is stored along with the security groups they belong to.

7.92.2 Constructor & Destructor Documentation

7.92.2.1 User()

```
Bds::User::User (
    BUInt32 id = 0,
    BString user = BString(),
    BString password = BString(),
    BString name = BString(),
    BString email = BString(),
    BString telephone = BString(),
    BString address = BString(),
    BInt32 enabled = 0,
    BList< BString > groups = BList< BString >() )
```

7.92.3 Member Function Documentation

7.92.3.1 getType()

```
BString Bds::User::getType ( )
```

7.92.3.2 setMembers()

```
BError Bds::User::setMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.92.3.3 setMember()

```
BError Bds::User::setMember (
    BString name,
    BString value ) [virtual]
```

Reimplemented from **BObj**.

7.92.3.4 getMembers()

```
BError Bds::User::getMembers (
    BDictString & members ) [virtual]
```

Reimplemented from **BObj**.

7.92.3.5 getMember()

```
BError Bds::User::getMember (
    BString name,
    BString & value ) [virtual]
```

Reimplemented from **BObj**.

7.92.4 Member Data Documentation

7.92.4.1 id

```
BUInt32 Bds::User::id
```

The unique user ID.

7.92.4.2 user

```
BString Bds::User::user
```

The [User](#) ID.

7.92.4.3 password

```
BString Bds::User::password
```

The [User](#)'s password.

7.92.4.4 name

BString Bds::User::name

The [User](#)'s full name.

7.92.4.5 email

BString Bds::User::email

The [User](#)'s email Address.

7.92.4.6 telephone

BString Bds::User::telephone

The [User](#)'s telephone number.

7.92.4.7 address

BString Bds::User::address

The [User](#)'s postal address.

7.92.4.8 enabled

BInt32 Bds::User::enabled

Whether the users account is enabled.

7.92.4.9 groups

BList< **BString** > Bds::User::groups

The security groups the user belongs to.

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

- [BdsD.h](#)
- [BdsD.cc](#)

Chapter 8

File Documentation

8.1 /src/blacknest/bds/bds/bdsDataLib/BdsCompress.cpp File Reference

```
#include <BdsCompress.h>
#include <BEndian.h>
```

Namespaces

- namespace [Bds](#)

Functions

- **BError** [Bds::bdsUnCompressCm8](#) (**BUInt8** *buffer, **BUInt** n, **BArray**< **BInt32** > & data)
Uncompress CM8 formatted data.

8.2 /src/blacknest/bds/bds/bdsDataLib/BdsCompress.d File Reference

8.3 /src/blacknest/bds/bds/bdsDataLib/BdsCompress.h File Reference

```
#include <BError.h>
#include <BArray.h>
```

Classes

- class [Bds::CompressSteim1](#)
Steim1 un-compress class.

Namespaces

- namespace [Bds](#)

Functions

- **Error** [Bds::bdsUnCompressCm8](#) ([BUInt8](#) *buffer, [BUInt](#) n, [BArray](#)< [BInt32](#) > & data)
Uncompress CM8 formatted data.
- **Error** [Bds::bdsUnCompressSteim1](#) ([BUInt8](#) *buffer, [BUInt](#) n, [BArray](#)< [BInt32](#) > & data)
Uncompress STEIM1 formatted data.

8.4 BdsCompress.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsCompress.h Data Compression Functions
3  * T.Barnaby, BEAM Ltd, 2009-05-15
4  *****/
5 */
6 #ifndef BdsCompress_H
7 #define BdsCompress_H
8
9 #include <BError.h>
10 #include <BArray.h>
11
12 namespace Bds {
13
14 BError bdsUnCompressCm8(BUInt8* buffer, BUInt n, BArray<BInt32>& data);
15 BError bdsUnCompressSteim1(BUInt8* buffer, BUInt n, BArray<BInt32>& data);
16
17 class CompressSteim1 {
18 public:
19     CompressSteim1();
20     void setByteOrder(int swap);
21     void clear();
22
23     BError unCompress(void* buffer, BUInt numSamples, BArray<BInt32>& data);
24 private:
25     int oswapBytes;
26 };
27
28
29 }
30 #endif
```

8.5 /src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.cpp File Reference

```
#include <BdsDataCollate.h>
```

Namespaces

- namespace [Bds](#)

8.6 /src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.d File Reference

8.7 /src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.h File Reference

```
#include <BString.h>
#include <BFile.h>
#include <BTimeStamp.h>
#include <BdsD.h>
#include <BdsDataFile.h>
```

Classes

- class [Bds::DataCollate](#)

Not sure if this is used or what it does.

Namespaces

- namespace [Bds](#)

8.8 BdsDataCollate.h

[Go to the documentation of this file.](#)

```
1 /*****
2  *   BdsDataCollate.h   Data File Access
3  *   T.Barnaby, BEAM Ltd, 2008-07-29
4  *****/
5 */
6 #ifndef BdsDataCollate_H
7 #define BdsDataCollate_H
8
9 #include <BString.h>
10 #include <BFile.h>
11 #include <BTimeStamp.h>
12 #include <BdsD.h>
13 #include <BdsDataFile.h>
14
15 namespace Bds {
16
17     class DataCollate {
18     public:
19         DataCollate();
20         ~DataCollate();
21
22         BError addSource(DataFile& dataFile, BUInt channel);
23         BError readData(BUInt32 blockNumber, DataBlock& data);
24
25     protected:
26     };
27
28 };
29
30 }
31 #endif
```

8.9 /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.cpp File Reference

```
#include <BdsDataFile.h>
```

Namespaces

- namespace [Bds](#)

8.10 /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.d File Reference

8.11 /src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h File Reference

```
#include <BString.h>
#include <BFile.h>
#include <BdsLib.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataBlockPos](#)
This defines the position of a data block in a file. It is used by the BDS data converters to order blocks by time.
- class [Bds::DataFileOptions](#)
This defines a list of BDS data converter options.
- class [Bds::DataFile](#)
This class defines the interface for generic data file access that all of the BDS data converters share.

Namespaces

- namespace [Bds](#)

8.12 BdsDataFile.h

[Go to the documentation of this file.](#)

```
1  /*****
2  *   BdsDataFile.h   Data File Access
3  *   T.Barnaby, BEAM Ltd, 2008-06-24
4  *****/
5  */
6  #ifndef BdsDataFile_H
7  #define BdsDataFile_H
8
9  #include <BString.h>
10 #include <BFile.h>
11 #include <BdsLib.h>
12 #include <BdsD.h>
13
14 namespace Bds {
15
16     class DataBlockPos {
17     public:
18         DataBlockPos(BTimeStamp startTime = 0, BTimeStamp endTime = 0, BUInt64 position = 0,
19                     BUInt order = 0, int ref = 0, BUInt numSamples = 0) : startTime(startTime), endTime(endTime),
20                     position(position), order(order), ref(ref), numSamples(numSamples){}
21         int operator<(const DataBlockPos& b) const { return startTime < b.startTime; }
22         BTimeStamp startTime;
23         BTimeStamp endTime;
24         BUInt64 position;
25         BUInt order;
26         int ref;
27         BUInt numSamples;
```

```

27 };
28
29 class DataFileOptions {
30 public:
31     DataFileOptions(int options = 0){    ooptionList = options; }
32     operator int (){    return ooptionList; }
33     DataFileOptions& operator|=(int o){    ooptionList |= o; return *this; }
34
35     int    ooptionList;
36     BArray<BUInt>  oignoreBlockList;
37 };
38
39 class DataFile {
40 public:
41     enum DataOrder    { DataOrderUnknown, DataOrderAll, DataOrderSample, DataOrderChannel };
42     enum Features     { FeatureNone = 0x00, FeatureCanWrite = 0x01, FeatureCanRead = 0x02 };
43     enum WriteOptionsList { WriteOptionNone = 0x00, WriteOptionSensorData = 0x01, WriteOptionNoMetadata
44     = 0x02 };
45     enum ReadOptionsList { ReadOptionNone = 0x00, ReadOptionValidate = 0x01, ReadOptionFileNameProcess =
46     0x02, ReadOptionFixCorruptions = 0x04, ReadOptionReorder = 0x08, ReadOptionDeleteDuplicates = 0x10,
47     ReadOptionInfoExtra = 0x20, ReadOptionIgnoreSamplerate = 0x40, ReadOptionPrintBlocks = 0x80,
48     ReadOptionFixSampleRate = 0x100 };
49
50     DataFile();
51     virtual ~DataFile();
52
53     virtual void    init();
54     virtual BError    open(BString fileName, BString mode);
55     virtual BError    close();
56
57     virtual BError    setFormat(BString format);
58     virtual BString    getFileName();
59     virtual DataOrder    getDataOrder();
60     virtual int    getFeatures();
61     virtual BString    getFixesInfo();
62
63     // Write routines
64     virtual BError    setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
65     WriteOptionsList options = WriteOptionNone);
66     virtual BError    start(BUInt channel, BUInt segment);
67     virtual BError    writeData(const DataBlock& data);
68     virtual BError    end();
69     virtual BError    flush();
70
71     // Read routines
72     virtual BError    fileNameProcess();
73     virtual BError    getFormat(BString& format);
74     virtual BError    getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
75
76     virtual BError    seekBlock(BUInt32 channel, BUInt segment, BTimeStamp time, BUInt32&
77     blockNumber, BUInt64& sampleNumber, DataBlock& data);
78     virtual BError    readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock&
79     dataBlock);
80
81     virtual BError    getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>&
82     errors);
83
84     // Misc functions
85     void    dataErrorFixup(const DataInfo& dataInfo, BList<DataError>& errors);
86     BInt64    timeCompare(BTimeStamp t1, BTimeStamp t2, BUInt diff);
87     int    duplicateCheck(const DataBlock& data1, const DataBlock& data2, BUInt channel = 0);
88     BUInt64    getFilePosition();
89     static DataFormats    getFormats();
90
91 protected:
92     BString    offileName;
93     BString    omode;
94     BTimeStamp    offileNameTime;
95     BFile    offile;
96     BString    offormat;
97 };
98
99 #endif

```

8.13 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.cpp File Reference

```

#include <BdsDataFileAd22.h>
#include <TimeControlPll.h>

```

```
#include <BDebug.h>
#include <errno.h>
```

Namespaces

- namespace [Bds](#)

Macros

- #define [DEBUG_VELATRACK](#) 1

8.13.1 Macro Definition Documentation

8.13.1.1 [DEBUG_VELATRACK](#)

```
#define DEBUG_VELATRACK 1
```

8.14 [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.d](#) File Reference

8.15 [/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.h](#) File Reference

```
#include <BdsDataFile.h>
```

Classes

- class [Bds::DataFileAd22](#)
Data file convertor for AD22 format files.

Namespaces

- namespace [Bds](#)

8.16 BdsDataFileAd22.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileAd22.h   BDRS Data File Access
3  *   T.Barnaby, BEAM Ltd, 2014-10-08
4  *****/
5  */
6  #ifndef BdsDataFileAd22_H
7  #define BdsDataFileAd22_H
8
9  #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     class DataFileAd22 : public DataFile{
14     public:
15         DataFileAd22();
16
17         int         getFeatures();
18         DataOrder    getDataOrder();
19         BString      getFixesInfo();
20
21         BError        getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
22         BError        readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
23
24         static DataFormats    getFormats();
25     private:
26         BError        readBlock(BUInt32 channel, BUInt64 pos, DataBlock& data);
27         BError        getTimeCode(BTimeStamp startTime, BTimeStampMs& timeCode);
28
29         int           omagic;
30         int           oyear;
31         int           oblockYear;
32         BUInt32        oblockSize;
33         double         osampleRate;
34         DataInfo        odataInfo;
35         BArray<DataBlockPos> oblockPositions;
36     };
37 };
38
39 }
40 #endif

```

8.17 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.cpp File Reference

```

#include <BdsDataFileAscii.h>
#include <BTimeStamp.h>

```

Namespaces

- namespace [Bds](#)

Functions

- static **BString** [Bds::nullString](#) (BString s)

8.18 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.d File Reference

8.19 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.h File Reference

```
#include <BdsDataFile.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataFileAscii](#)
Data file convertor for ASCII format files.

Namespaces

- namespace [Bds](#)

8.20 BdsDataFileAscii.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileAscii.h Data File Access
3  * T.Barnaby, BEAM Ltd, 2008-06-24
4  *****/
5 */
6 #ifndef BdsDataFileAscii_H
7 #define BdsDataFileAscii_H
8
9 #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14 class DataFileAscii : public DataFile {
15 public:
16     DataFileAscii();
17
18     BError open(BString fileName, BString mode);
19
20     DataOrder getDataOrder();
21     int getFeatures();
22
23     BError setFormat(BString format);
24     BError setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
25     WriteOptionsList options = WriteOptionSensorData);
26     BError start(BUInt channel, BUInt segment);
27     BError writeData(const DataBlock& data);
28     BError end();
29
30     static DataFormats getFormats();
31
32 private:
33     BError writeMetadata(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
34     WriteOptionsList options = WriteOptionSensorData);
35
36     DataInfo odataInfo;
37     ChannelInfos ochannelInfos;
38     BString oformat;
39 };
40 }
41 #endif
```

8.21 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.cpp File Reference

```
#include <BdsDataFileBdrs.h>
#include <BDebug.h>
#include <errno.h>
```

Namespaces

- namespace [Bds](#)

8.22 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.d File Reference

8.23 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.h File Reference

```
#include <BdsDataFile.h>
```

Classes

- class [Bds::DataFileBdrs](#)
Data file convertor for BDRS format files.

Namespaces

- namespace [Bds](#)

8.24 BdsDataFileBdrs.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileBdrs.h   BDRS Data File Access
3  *      T.Barnaby,  BEAM Ltd,   2008-06-24
4  *****/
5 */
6 #ifndef BdsDataFileBdrs_H
7 #define BdsDataFileBdrs_H
8
9 #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     class DataFileBdrs : public DataFile{
14     public:
15         DataFileBdrs();
16
17         int    getFeatures();
18         DataOrder    getDataOrder();
19     }
```

```

20     BString          getFixesInfo();
21
22     BError           getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
23     BError           readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
24
25     static DataFormats getFormats();
26 private:
27     BError           readBlock(BUInt32 channel, BUInt64 pos, DataBlock& data);
28
29     int              omagic;
30     int              oyear;
31     int              oblockYear;
32     BUInt32          oblockSize;
33     double           osampleRate;
34     DataInfo         odataInfo;
35     BArray<DataBlockPos> oblockPositions;
36 };
37
38 }
39 #endif

```

8.25 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.cpp File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <byteswap.h>
#include <BdsLib.h>
#include <BdsDataLib.h>
#include <BdsDataFileBds.h>
#include <BBuffer.h>
#include <BDebug.h>
#include <zlib.h>
#include <canada_compress.h>

```

Namespaces

- namespace [Bds](#)

Macros

- #define [LDEBUG](#) 0
- #define [LDEBUG2](#) 0
- #define [LDEBUG3](#) 0
- #define [dlprintf](#)(fmt, a...)
- #define [dl2printf](#)(fmt, a...)
- #define [dl3printf](#)(fmt, a...)
- #define [ALLOW_TIMESTAMP_JITTER](#) 1
- #define [TIMESTAMP_JITTER](#) 100

Functions

- [BUInt32 Bds::crc](#) ([BUInt32](#) crc, void * **data**, int numBytes)

Variables

- const BString Bds::BdsDataFileVersion = "1.2.0"

8.25.1 Macro Definition Documentation

8.25.1.1 LDEBUG

```
#define LDEBUG 0
```

8.25.1.2 LDEBUG2

```
#define LDEBUG2 0
```

8.25.1.3 LDEBUG3

```
#define LDEBUG3 0
```

8.25.1.4 dlprintf

```
#define dlprintf(  
    fmt,  
    a... )
```

8.25.1.5 dl2printf

```
#define dl2printf(  
    fmt,  
    a... )
```

8.25.1.6 dl3printf

```
#define dl3printf(  
    fmt,  
    a... )
```

8.25.1.7 ALLOW_TIMESTAMP_JITTER

```
#define ALLOW_TIMESTAMP_JITTER 1
```

8.25.1.8 TIMESTAMP_JITTER

```
#define TIMESTAMP_JITTER 100
```

8.26 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.d File Reference

8.27 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h File Reference

```
#include <BdsDataFile.h>
#include <BBuffer.h>
```

Classes

- struct [Bds::BdsDataBlockHeader](#)
BdsDataFileBds: internal fixed size BDS Data Block header.
- struct [Bds::BdsDataBlock](#)
BdsDataFileBds: internal fixed size BDS Data Block.
- struct [Bds::BdsDataPacketHeader](#)
BdsDataFileBds internal file storage packet header.
- class [Bds::BdsDataPacket](#)
BdsDataFileBds: internal file storage packet.
- class [Bds::BdsDataBlockPos](#)
BdsDataFileBds: internal file storage data block position.
- class [Bds::BdsDataSegment](#)
BdsDataFileBds: internal file storage data segment.
- class [Bds::BdsDataStreamlet](#)
BdsDataFileBds: internal file storage data streamlet.
- class [Bds::DataFileBds](#)
This class implements the BDS Data File/Stream access system.

Namespaces

- namespace [Bds](#)

Enumerations

- enum `Bds::BdsDataType` { `Bds::BdsDataTypeBlock` = 0x42534442 , `Bds::BdsDataTypeInfo` = 0x30534442 , `Bds::BdsDataTypeData` = 0x31534442 , `Bds::BdsDataTypeInfoExtra` = 0x32534442 }

BdsDataFileBds: internal file block type field.

8.28 BdsDataFileBds.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileBds.h    Bds internal data file format
3  *
4  * T.Barnaby, BEAM Ltd, 2008-05-06
5  *
6  * Supported sub-formats:
7  *   BDS, BDS-SM BDS Sample Multiplexed
8  *   BDS-CM BDS Channel Multiplexed
9  *   BDS-SM-CC BDS Sample Multiplexed, Canadian Compression
10 *
11 * A BDS data file contains multiple streams. Each stream, numbered from 0 upwards, contains information
12 *   blocks or
13 *   data blocks for a specic data channel or set of channels if the channel data is sample multiplexed
14 *   into a block.
15 * Data streams are split into segments at time discontinuities when using the getInfo() function..
16 * Each segment consists of a set of data blocks and each data block can be for one channel or a number
17 *   of sample multiplexed channels.
18 * The siesmic channel to stream mapping is stored in the Information blocks. An information block is
19 *   located at the start of the
20 *   file and at each append point. You can only append data channels that the files was originally
21 *   created for (ie. the first setInfo() API call).
22 *
23 * If the ReadOptionReorder option is used the data segments and blocks are sorted into time order.
24 *
25 * The file can be appended to. When opened for appending
26 */
27 #ifndef BdsDataFileBds_H
28 #define BdsDataFileBds_H    1
29
30 #include <BdsDataFile.h>
31 #include <BBuffer.h>
32
33 namespace Bds {
34
35 enum BdsDataType { BdsDataTypeBlock = 0x42534442, BdsDataTypeInfo = 0x30534442, BdsDataTypeData =
36   0x31534442, BdsDataTypeInfoExtra = 0x32534442 };
37
38 struct BdsDataBlockHeader {
39   BUInt32    type;
40   BUInt32    length;
41   BUInt32    packetOffset;
42 };
43
44 struct BdsDataBlock {
45   BdsDataBlockHeader header;
46   char               data[4];
47 };
48
49 struct BdsDataPacketHeader {
50   BUInt32    type;
51   BUInt32    length;
52   BUInt32    streamlet;
53   BUInt32    sequence;
54   BUInt32    checksum;
55   BTimeStamp startTime;
56   BTimeStamp endTime;
57 };
58
59 class BdsDataPacket : public BBufferStore {
60 public:
61   BdsDataPacket();
62   ~BdsDataPacket();
63
64   void clear();
65   void reset();
66
67   void setChecksumAndLength();
68   BError validateChecksum();
69
70   BError setHeader(const BdsDataPacketHeader& header);
71
72 }
```

```

70     BError          getHeader(BdsDataPacketHeader& header);
71
72     void            dump();
73 };
74
75 class BdsDataBlockPos {
76 public:
77     BdsDataBlockPos(BTimeStamp startTime = 0, BTimeStamp endTime = 0, BUInt32 channel = 0,
78         BUInt32 numChannels = 0, BUInt32 segment = 0, BUInt64 position = 0, BUInt64 numSamples = 0) :
79         startTime(startTime), endTime(endTime), channel(channel), numChannels(numChannels),
80         segment(segment), position(position), numSamples(numSamples){}
81     int              operator<(const BdsDataBlockPos& b) const { return startTime < b.startTime; }
82     BTimeStamp       startTime;
83     BTimeStamp       endTime;
84     BUInt32          channel;
85     BUInt32          numChannels;
86     BUInt32          segment;
87     BUInt64          position;
88     BUInt64          numSamples;
89 };
90
91 class BdsDataSegment {
92 public:
93     BdsDataSegment() : numBlocks(0), numSamples(0), sampleRate(0.0){};
94     int              operator<(const BdsDataSegment& b) const { return startTime < b.startTime; }
95     BTimeStamp       startTime;
96     BTimeStamp       endTime;
97     BUInt32          numBlocks;
98     BUInt32          numSamples;
99     double           sampleRate;
100     BArray<BdsDataBlockPos> blocks;          // The list of blocks in the segment
101 };
102
103 class BdsDataStreamlet {
104 public:
105     BdsDataStreamlet() : packetNumber(0), position(0), channel(0), numChannels(0){}
106     BUInt32          packetNumber;
107     BUInt64          position;
108     BUInt32          channel;          // The base channel number of data in this streamlet
109     BUInt32          numChannels;      // The number of channels of data in this streamlet
110     BArray<BdsDataBlockPos> blocks;    // All blocks in this streamlet
111     BArray<BdsDataSegment> segments;  // Segments of data
112 };
113
114 class DataFileBds : public DataFile {
115 public:
116     enum { StreamsMax = 256 };
117     enum { DefaultBlockSize = 65536 };
118     enum PackFormat { PackFormat_Unknown = 0, PackFormat_SM = 1, PackFormat_CM = 2,
119         PackFormat_SM_CC = 3 };
120
121     // Generic access
122     DataFileBds();
123     ~DataFileBds();
124
125     BError          open(BString fileName, BString mode);
126     BError          flush();
127     BError          close();
128
129     // Write routines
130     BError          setFormat(BString format);
131     BError          setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
132         WriteOptionsList options = WriteOptionSensorData);
133     BError          writeData(const DataBlock& data);
134
135     // Read routines
136     DataOrder       getDataOrder();
137     BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
138
139     BError          seekBlock(BUInt32 channel, BUInt segment, BTimeStamp time, BUInt32& blockNumber,
140         BUInt64& sampleNumber, DataBlock& dataBlock);
141     BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock&
142         dataBlock);
143
144     static DataFormats getFormats();
145
146     BError          setDiskBlockSize(BUInt32 blockSize);
147     BUInt32         getDiskBlockSize();
148
149     // Low Level Packet I/O
150     BError          streamletToChannel(BUInt streamlet, BUInt& channel);
151     BError          setWritePositionForAppend();

```

```

154     BError          setReadPositionToStart();
155     BError          packetRead(BdsDataPacket& packet);
156     BError          packetWrite(BdsDataPacket& packet);
157
158 private:
159     void            clear();
160
161     // DataBlock exists checks
162     BError          dataBlockAdd(BUInt32 streamlet, BTimeStamp startTime, BTimeStamp endTime);
163     Bool            dataBlockCheckIfExists(BUInt32 streamlet, BTimeStamp startTime, BTimeStamp endTime);
164
165     // Low level file access
166     BError          setInfoRepeat(BUInt32 repeat);
167     BError          infoSet(BUInt32 streamlet, BTimeStamp startTime, BTimeStamp endTime, BDictString&
info);
168     BError          infoGet(BdsDataPacket& packet, BDictString& info);
169     BError          readInfoPacket();
170     BError          addInfoPacket();
171     BError          addInfoExtraPacket();
172     BError          streamletFromChannel(BUInt channel, BUInt& streamlet);
173
174     BError          dataAppend(const DataBlock& data);
175     BError          dataGet(BUInt channel, BdsDataPacket& packet, DataBlock& data);
176
177     // Low level Disk Block access functions
178     BError          diskBlockWrite(void* data, BUInt32 numBytes, int header = 0);
179     BError          diskBlockWriteFlush();
180     BError          diskBlockRead(void* data, BUInt32 numBytes, int header = 0);
181     BError          diskBlockSeek(BUInt64 position);
182
183     BString         omode;
184     BString         oformat;
185     PackFormat      opackFormat;
186     DataInfo        odataInfo;
187     ChannelInfos     ochannelInfos;
188
189     BUInt32         oinfoRepeat;
190     BDictString      oinfo;
191     BdsDataPacketHeader oinfoHeader;
192
193     // Disk Block access data
194     BUInt32         odiskBlockSize;
195     BdsDataBlock*   odiskBlockRead;
196     BUInt64         odiskPositionRead;
197     BdsDataBlock*   odiskBlockWrite;
198     BUInt64         odiskPositionWrite;
199
200     BArray<BdsDataStreamlet> ostreamlets;
201     BdsDataPacket    opacket;
202 };
203 }
204 #endif

```

8.29 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.cpp File Reference

```

#include <BdsDataFileBknas.h>
#include <BTimeStamp.h>
#include <limits.h>

```

Namespaces

- namespace [Bds](#)

Functions

- `template<typename T>`
`T clip (T in, T low, T high)`

8.29.1 Function Documentation

8.29.1.1 clip()

```
template<typename T >
T clip (
    T in,
    T low,
    T high ) [inline]
```

8.30 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.d File Reference

8.31 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.h File Reference

```
#include <BdsDataFile.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataFileBknas](#)
Data file convertor for BKNAS format files.

Namespaces

- namespace [Bds](#)

8.32 BdsDataFileBknas.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileBknas.h Data File Access
3  * T.Barnaby, BEAM Ltd, 2008-06-24
4  *****/
5 */
6 #ifndef BdsDataFileBknas_H
7 #define BdsDataFileBknas_H
8
9 #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14     class DataFileBknas : public DataFile {
15     public:
16         DataFileBknas();
17
18     }
```

```

19     BError          open(BString fileName, BString mode);
20
21     BError          setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
22     WriteOptionsList options = WriteOptionSensorData);
23     BError          writeData(const DataBlock& data);
24
25     static DataFormats getFormats();
26 private:
27     DataInfo          odataInfo;
28     ChannelInfos       ochannelInfos;
29 };
30
31 }
32 #endif

```

8.33 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.cpp File Reference

```

#include <BdsDataFileCd.h>
#include <arpa/inet.h>
#include <errno.h>
#include <canada_compress.h>
#include <BEndian.h>
#include <BDebug.h>

```

Namespaces

- namespace [Bds](#)

Macros

- #define [LDEBUG](#) 0
- #define [dprintf](#)(fmt, a...)
- #define [INCLUDE_CHANNEL_AUTH](#) 1
- #define [ALLOW_TIMESTAMP_JITTER](#) 1
- #define [TIMESTAMP_JITTER](#) 100
- #define [MULTIPLE_SEGMENT](#) 0
- #define [SEGMENT_GAP](#) 3600000
- #define [ntohl](#)(x) __bswap_64(x)
- #define [htonl](#)(x) [ntohl](#)(x)

Functions

- static void [Bds::crclnit](#) ()
- static uint64_t [Bds::crc64](#) (const void *buffer, const uint32_t len)
- **BString** [Bds::getHexString](#) (char * data, int len)
- int [Bds::duplicateDump](#) (DataBlock &data1, DataBlock &data2, int channel)

Variables

- const int [ErrorFormatNoDataFormat](#) = 100
- static uint64_t [Bds::crcVec](#) [256]
- static int [Bds::crclnitDone](#)

8.33.1 Macro Definition Documentation

8.33.1.1 LDEBUG

```
#define LDEBUG 0
```

8.33.1.2 dprintf

```
#define dprintf(  
    fmt,  
    a... )
```

8.33.1.3 INCLUDE_CHANNEL_AUTH

```
#define INCLUDE_CHANNEL_AUTH 1
```

8.33.1.4 ALLOW_TIMESTAMP_JITTER

```
#define ALLOW_TIMESTAMP_JITTER 1
```

8.33.1.5 TIMESTAMP_JITTER

```
#define TIMESTAMP_JITTER 100
```

8.33.1.6 MULTIPLE_SEGMENT

```
#define MULTIPLE_SEGMENT 0
```

8.33.1.7 SEGMENT_GAP

```
#define SEGMENT_GAP 3600000
```


8.33.1.8 ntohll

```
#define ntohll(
    x ) __bswap_64(x)
```

8.33.1.9 htonll

```
#define htonll(
    x ) ntohll(x)
```

8.33.2 Variable Documentation

8.33.2.1 ErrorFormatNoDataFormat

```
const int ErrorFormatNoDataFormat = 100
```

8.34 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.d File Reference

8.35 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h File Reference

```
#include <BdsDataFile.h>
```

Classes

- struct [Bds::CdChannel_1v0](#)
BdsDataFile: Internal CD1.0 channel information.
- struct [Bds::CdDataFormatFrame_1v0](#)
BdsDataFile: Internal CD1.0 frame information.
- class [Bds::CdDataChannel](#)
BdsDataFile: Internal CD channel information.
- class [Bds::CdPacketData](#)
BdsDataFile: Internal CD data packet.
- class [Bds::CdFlag](#)
BdsDataFile: Internal CD flag.
- class [Bds::DataFileCd](#)
Data file convertor for CD1.0 and CD1.1 file formats.

Namespaces

- namespace [Bds](#)

8.36 BdsDataFileCd.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileCd.h BDRS Data File Access
3  *   T.Barnaby, BEAM Ltd, 2011-06-01
4  *****/
5  */
6  #ifndef BdsDataFileCd_H
7  #define BdsDataFileCd_H
8
9  #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     struct CdChannel_lv0 {
14         BUInt8      auth;
15         BUInt8      compress;
16         BUInt8      spare0;
17         BUInt8      spare1;
18         BFloat32    calibrationFactor;
19         BFloat32    calibrationPeriod;
20         char        name[16];
21         char        stationName[16];
22         char        channelName[16];
23         BUInt32     channel;
24     };
25
26     struct CdDataFormatFrame_lv0 {
27         BUInt32     frameType;
28         BUInt32     frameLength;
29         BUInt32     maxFrameLength;
30         BUInt32     numChannels;
31         BUInt32     period;
32         CdChannel_lv0 channels[100];
33     };
34
35     class CdDataChannel {
36     public:
37         BString      station;
38         BString      channel;
39         char         mode[24];
40         char         status[32];
41         BTimeStamp    startTime;
42         BUInt32      period;
43         BUInt32      numSamples;
44         BUInt32      dataSize;
45         BUInt8*      data;
46     };
47
48     class CdPacketData {
49     public:
50         BUInt32      frameType;
51         BUInt32      trailerOffset;
52         char         creator[8];
53         char         destination[8];
54         BUInt64      sequenceNum;
55         BUInt32      series;
56
57         BUInt32      numChannels;
58         BUInt32      period;
59         BTimeStamp    startTime;
60         BArray<CdDataChannel> channels;
61
62         BUInt32      authKey;
63         BUInt32      authSize;
64         char*        auth;
65         BUInt64      crc;
66     };
67
68     class CdFlag {
69     public:
70         CdFlag() { dead = zeroed = 0; };
71         int      dead;
72         int      zeroed;
73     };
74
75     class DataFileCd : public DataFile{
76     public:
77         DataFileCd();
78
79         int      getFeatures();
80         DataOrder    getDataOrder();
81         BString     getFixesInfo();
82     };
83

```

```

89     BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
90     BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
91
92     static DataFormats getFormats();
93 private:
94     BError          readPacket_lv0(int validateNext);
95     BError          readPacket_lv0_Scan(BUInt64& pos);
96     BError          readBlock_lv0(BUInt32 channel, BUInt64 pos, DataBlock& data, int addInfo, int
    validateNext);
97     BError          readPacket_lv1(int validateNext);
98     BError          readPacket_lv1_Scan(BUInt64& pos);
99     BError          readBlock_lv1(BUInt32 channel, BUInt64 pos, DataBlock& data, int addInfo, int
    validateNext);
100    DataError        getBlockReorderInfo();
101
102    DataFileOptions  ooptions;
103    bool             ohasYear2000Blocks;
104    DataInfo         odataInfo;
105    BBufferStore     opacket;
106    BBufferStore     opacketNext;
107    BArray<BArray<DataBlockPos> > oblockPositions;
108
109    int             odataFormat;
110    BArray<CdDataFormatFrame_lv0> odataFormats;
111    BArray<CdFlag>   ochannelFlags;
112 };
113
114 }
115 #endif

```

8.37 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.cpp File Reference

```

#include <BdsDataFileCss.h>
#include <BEndian.h>
#include <BDebug.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

8.38 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.d File Reference

8.39 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.h File Reference

```

#include <BdsDataFile.h>
#include <BdsSeedType.h>

```

Classes

- class [Bds::DataFileCssData](#)
DataFileCss internal CSS data type.
- class [Bds::DataFileCss](#)
Data file convertor for CSS format files.

Namespaces

- namespace [Bds](#)

8.40 BdsDataFileCss.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileCss.h   CSS Data File Access
3  *       T.Barnaby,  BEAM Ltd,  2015-03-02
4  *****/
5  */
6  #ifndef BdsDataFileCss_H
7  #define BdsDataFileCss_H
8
9  #include <BdsDataFile.h>
10 #include <BdsSeedType.h>
11
12 namespace Bds {
13
14     class DataFileCssData : public BdsSeedType {
15     public:
16         DataFileCssData();
17         ~DataFileCssData();
18         BError          set(BString line);
19
20         BString         sta;
21         BString         chan;
22         double          startTime;
23         int             wfid;
24         int             chanid;
25         int             jdate;
26         double          endTime;
27         int             nsamp;
28         double          sampleRate;
29         double          calibrationFactor;
30         double          calibrationFreq;
31         BString         instType;
32         BString         segtype;
33         BString         datatype;
34         BString         clip;
35         BString         dirName;
36         BString         fileName;
37         BUInt32         fileOffset;
38         int             commId;
39         BString         loadDate;
40
41         BFile*          file;
42         BUInt32         sampleFormat;
43         BUInt32         sampleSize;
44         int             sampleBigEndian;
45     };
46
47     class DataFileCss : public DataFile {
48     public:
49         DataFileCss();
50
51         int             getFeatures();
52         DataOrder        getDataOrder();
53
54         BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
55         BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
56
57         static DataFormats    getFormats();
58     private:
59         BError          getCssFormat(BString format, BUInt32& sampleFormat, BUInt32& sampleSize, int&
60             sampleEndian);
61
62         BUInt           oblockSamples;
63         DataInfo         odataInfo;
64         BArray<BArray<DataFileCssData> > odataCss;
65     };
66 };
67
68 #endif

```

8.41 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.cpp File Reference

```
#include <BdsDataFileGcf.h>
#include <arpa/inet.h>
#include <errno.h>
#include <gcf2.h>
#include <BDebug.h>
```

Namespaces

- namespace [Bds](#)

Macros

- #define [DEBUG](#) 0
- #define [TEST_REORDER](#) 0

8.41.1 Macro Definition Documentation

8.41.1.1 DEBUG

```
#define DEBUG 0
```

8.41.1.2 TEST_REORDER

```
#define TEST_REORDER 0
```

8.42 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.d File Reference

8.43 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.h File Reference

```
#include <BdsDataFile.h>
```

Classes

- struct [Bds::GcfChannel](#)
DataFileGcf internal GCF channel information.
- class [Bds::DataFileGcf](#)
Data file convertor for GCF format files.

Namespaces

- namespace [Bds](#)

8.44 BdsDataFileGcf.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileGcf.h   BDRS Data File Access
3  *   T.Barnaby, BEAM Ltd, 2008-06-24
4  *****/
5  */
6  #ifndef BdsDataFileGcf_H
7  #define BdsDataFileGcf_H
8
9  #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     struct GcfChannel {
14         BString      systemId;
15         BString      streamId;
16         BUInt        type;
17         BUInt        sampleRate;
18         BUInt        format;
19         BUInt32      channel;
20     };
21
22     class DataFileGcf : public DataFile{
23     public:
24         DataFileGcf();
25
26         int          getFeatures();
27         DataOrder    getDataOrder();
28         BString      getFixesInfo();
29
30         BError       getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
31         BError       readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
32
33         static DataFormats getFormats();
34     private:
35         BError       readBlock(BUInt32 channel, BUInt64 pos, DataBlock& data, int addInfo, BUInt&
36         dataChan);
37         DataError    getBlockReorderInfo();
38
39         BUInt32      oblockSize;
40         DataFileOptions options;
41         DataInfo      odataInfo;
42         BArray<GcfChannel> ochannels;
43         BArray<BArray<DataBlockPos> > oblockPositions;
44     };
45
46 }
47
48 #endif

```

8.45 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.cpp File Reference

```

#include <BdsDataFileIdc.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

Macros

- `#define` [LDEBUG](#) 0
- `#define` [dprintf](#)(fmt, a...)

8.45.1 Macro Definition Documentation

8.45.1.1 LDEBUG

```
#define LDEBUG 0
```

8.45.1.2 dprintf

```
#define dprintf(  
    fmt,  
    a... )
```

8.46 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.d File Reference

8.47 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.h File Reference

```
#include <BdsDataFile.h>  
#include <BdsD.h>
```

Classes

- class [Bds::DataFileIdc](#)
This class defines the interface for IDC response data file access.

Namespaces

- namespace [Bds](#)

8.48 BdsDataFileIdc.h

[Go to the documentation of this file.](#)

```

1 /*****
2  * BdsDataFileIdc.h    IDC Response Data File Access
3  * T.Barnaby, BEAM Ltd, 2021-05-27
4  *****/
5 */
6 #ifndef BdsDataFileIdc_H
7 #define BdsDataFileIdc_H
8
9 #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14 class DataFileIdc : public DataFile {
15 public:
16     DataFileIdc();
17
18     int         getFeatures();
19
20     // Read routines
21     BError      getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>& errors);
22
23     // Write routines
24     BError      setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
25         WriteOptionsList options);
26
27     static DataFormats    getFormats();
28 private:
29     BError      readIdcResponse();
30     BError      writeIdcResponse(ChannelInfo& ci, Response& r);
31
32     DataInfo    odataInfo;
33     ChannelInfo ochannelInfo;
34     ChannelInfos ochannelInfos;
35 };
36
37 }
38 #endif

```

8.49 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIms.cpp File Reference

```

#include <BdsDataFileIms.h>
#include <BTimeStamp.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

Functions

- static **BError** [Bds::fixedString](#) (double v, int fieldWidth, int numDecimal, **BString** &str)
- void [Bds::dataCalculateDifference](#) (**Blnt32** &prevValue, **BArray**< **Blnt32** > & data)
- void [Bds::dataCalculateUnDifference](#) (**Blnt32** &prevValue, **BArray**< **Blnt32** > & data)
- **Blnt32** [Bds::dataChecksum](#) (**Blnt32** checksum, **BArray**< **Blnt32** > & data)
- **BError** [Bds::dataCompressCm6](#) (int &prevValue1, int &prevValue2, **BArray**< **Blnt32** > & data, **BString** &d)
- **BError** [Bds::dataDeCompressCm6](#) (int &prevValue1, int &prevValue2, **BString** &d, **BArray**< **Blnt32** > & data)
- static void [Bds::dataConvert](#) (const **BArray**< **BFloat64** > &dataIn, **BArray**< **Blnt32** > &dataOut)
- static **BString** [Bds::unitsCode](#) (Response &r)

Variables

- static char [Bds::cm6Table](#) [64]
- static **BUInt8** [Bds::cm6TableRev](#) [128]

8.50 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIms.d File Reference

8.51 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileIms.h File Reference

```
#include <BdsDataFile.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataFileIms](#)
Data file convertor for IMS format files.

Namespaces

- namespace [Bds](#)

8.52 BdsDataFileIms.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileIms.h    Data File Access
3  *      T.Barnaby, BEAM Ltd, 2008-08-08
4  *****/
5 */
6 #ifndef BdsDataFileIms_H
7 #define BdsDataFileIms_H
8
9 #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14 class DataFileIms : public DataFile {
15 public:
16     DataFileIms();
17
18     BError      open(BString fileName, BString mode);
19     BError      close();
20
21     DataOrder   getDataOrder();
22     int         getFeatures();
23
24     BError      setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
25         WriteOptionsList options = WriteOptionNone);
26     BError      start(BUInt channel, BUInt segment);
27     BError      writeData(const DataBlock& data);
28     BError      end();
29
30     BError      getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>& errors);
31
32     static DataFormats   getFormats();
33 private:
34     BError      writeResponses();
35     BError      writeResponse(ChannelInfo& ci, Response& r);
36 }
```

```

37     DataInfo      odataInfo;
38     ChannelInfos  ochannelInfos;
39     BUInt         owriteChannel;
40     BInt32        owriteChecksum;
41     BUInt         owriteColumn;
42
43     BInt32        oprevValue1;
44     BInt32        oprevValue2;
45 };
46
47 }
48 #endif

```

8.53 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.cpp File Reference

```

#include <BdsDataFileLac.h>
#include <BEndian.h>
#include <BDebug.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

8.54 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.d File Reference

8.55 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.h File Reference

```

#include <BdsDataFile.h>

```

Classes

- class [Bds::DataFileLac](#)
Data file convertor for LAC format files.

Namespaces

- namespace [Bds](#)

8.56 BdsDataFileLac.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileLac.h       LAC/BDRS Data File Access
3  *   T.Barnaby, BEAM Ltd, 2014-11-04
4  *****/
5  */
6  #ifndef BdsDataFileLac_H
7  #define BdsDataFileLac_H
8
9  #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     class DataFileLac : public DataFile{
14     public:
15         DataFileLac();
16
17         int         getFeatures();
18         DataOrder    getDataOrder();
19         BString      getFixesInfo();
20
21         BError       getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
22         BError       readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
23
24         static DataFormats    getFormats();
25     private:
26         BError       readBlock(BUInt32 channel, BUInt64 pos, DataBlock& data, BUInt64& posNext);
27
28         int          omagic;
29         int          oyear;
30         int          oblockYear;
31         BUInt32      oblockSize;
32         double        osampleRate;
33         DataInfo      odataInfo;
34         BArray<DataBlockPos> oblockPositions;
35         BUInt32      obadBlocks;
36     };
37 };
38
39 }
40 #endif

```

8.57 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.cpp File Reference

```

#include <BdsDataFileLog.h>
#include <BTimeStamp.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

Functions

- static **BString** [Bds::stringFormat](#) (BTimeStamp t)
- static **BString** [Bds::removeCR](#) (BString str)

8.58 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.d File Reference

8.59 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.h File Reference

```
#include <BdsDataFile.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataFileLog](#)
Data file convertor for LOG format files.

Namespaces

- namespace [Bds](#)

8.60 BdsDataFileLog.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileLog.h    Data File Access
3  *      T.Barnaby,  BEAM Ltd,   2008-06-24
4  *****/
5 */
6 #ifndef BdsDataFileLog_H
7 #define BdsDataFileLog_H
8
9 #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14 class DataFileLog : public DataFile {
15 public:
16     DataFileLog();
17
18     BError          open(BString fileName, BString mode);
19
20     DataOrder       getDataOrder();
21     int            getFeatures();
22
23     BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
24     BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
25
26     BError          setFormat(BString format);
27     BError          setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
28                             WriteOptionsList options = WriteOptionSensorData);
29     BError          start(BUInt channel, BUInt segment);
30     BError          writeData(const DataBlock& data);
31     BError          end();
32
33     static DataFormats getFormats();
34
35 private:
36     BError          readBlock(BUInt32 channel, BUInt64 pos, DataBlock& data, BUInt64& posNext);
37
38     DataInfo         odataInfo;
39     ChannelInfos      ochannelInfos;
40     BString           oformat;
41     BArray<DataBlockPos> oblockPositions;
42 };
43
44 }
45 #endif
```

8.61 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.cpp File Reference

```
#include <BdsDataFileResponse.h>
#include <errno.h>
```

Namespaces

- namespace [Bds](#)

Macros

- #define [LDEBUG](#) 0
- #define [dprintf](#)(fmt, a...)

8.61.1 Macro Definition Documentation

8.61.1.1 LDEBUG

```
#define LDEBUG 0
```

8.61.1.2 dprintf

```
#define dprintf(  
    fmt,  
    a... )
```

8.62 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.d File Reference

8.63 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.h File Reference

```
#include <BdsDataFile.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataFileResponse](#)

This class defines the interface for generic response data file access.

Namespaces

- namespace [Bds](#)

8.64 BdsDataFileResponse.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileResponse.h   Response Data File Access
3  *   T.Barnaby, BEAM Ltd, 2012-04-02
4  *****/
5  */
6  #ifndef BdsDataFileResponse_H
7  #define BdsDataFileResponse_H
8
9  #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14 class DataFileResponse : public DataFile {
15 public:
16     DataFileResponse();
17
18     int getFeatures();
19
20     // Read routines
21     BError getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>& errors);
22
23     // Write routines
24     BError setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
25                   WriteOptionsList options);
26
27     static DataFormats getFormats();
28 private:
29     BError readSacPoleZero();
30     BError readRawFap();
31     BError readIdcResponse();
32     BError writeIdcResponse(Response& r);
33
34     DataInfo odataInfo;
35     ChannelInfo ochannelInfo;
36     ChannelInfos ochannelInfos;
37 };
38
39 }
40 #endif

```

8.65 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.cpp File Reference

```

#include <BdsDataFileSac.h>
#include <BDebug.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

Macros

- `#define BDEBUGL1 0`

8.65.1 Macro Definition Documentation

8.65.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.66 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.d File Reference

8.67 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.h File Reference

```
#include <BdsDataFile.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataFileSac](#)
Data file convertor for SAC format files.

Namespaces

- namespace [Bds](#)

8.68 BdsDataFileSac.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileSac.h   Data File Access
3  *       T.Barnaby,  BEAM Ltd,   2010-09-15
4  *****/
5  */
6  #ifndef BdsDataFileSac_H
7  #define BdsDataFileSac_H
8
9  #include <BdsDataFile.h>
10 #include <BdsD.h>
11
12 namespace Bds {
13
14     class DataFileSac : public DataFile {
15     public:
16         DataFileSac();
17
18         int         getFeatures();
19
20         // Read routines
21         BError      getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>& errors);
22
23         // Write routines
24         BError      setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
25                             WriteOptionsList options = WriteOptionNone);
26
27         static DataFormats    getFormats();
28     private:
29         BError      readSacPoleZero();
30         BError      writeResponses();
31
32         DataInfo    odataInfo;
33         ChannelInfo  ochannelInfo;
34         ChannelInfos ochannelInfos;
35     };
36
37 }
38 #endif

```

8.69 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.cpp File Reference

```

#include <BdsDataFileStationXml.h>
#include <BDebug.h>
#include <cmath>

```

Namespaces

- namespace [Bds](#)

Macros

- #define [BDEBUGL1](#) 0
- #define [BDEBUGL2](#) 0

Variables

- const char * [Bds::node_types](#) []

8.69.1 Macro Definition Documentation

8.69.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.69.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.70 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.d File Reference

8.71 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.h File Reference

```
#include <BdsDataFile.h>  
#include <BdsD.h>  
#include <pugixml.hpp>
```

Classes

- class [Bds::DataFileStationXml](#)

This class defines the interface for generic response data file access.

Namespaces

- namespace [Bds](#)

8.72 BdsDataFileStationXml.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileStationXml.h Metadata Data Convertor
3  *   T.Barnaby, BEAM Ltd, 2021-01-14
4  *****/
5  */
6  #ifndef BdsDataFileStationXml_H
7  #define BdsDataFileStationXml_H
8
9  #include <BdsDataFile.h>
10 #include <BdsD.h>
11 #include <pugixml.hpp>
12
13 namespace Bds {
14
15 class DataFileStationXml : public DataFile {
16 public:
17     DataFileStationXml();
18
19     int getFeatures();
20
21     // Write routines
22     BError setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
23     WriteOptionsList options);
24
25     // Read routines
26     BError getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>& errors);
27
28     static DataFormats getFormats();
29
30 private:
31     DataInfo odataInfo;
32     ChannelInfos ochannelInfos;
33 };
34
35 }
36 #endif

```

8.73 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.cpp

File Reference

```

#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <BdsDataFileTapeDigitiser.h>
#include <BTimeStampMs.h>

```

Namespaces

- namespace [Bds](#)

Enumerations

- enum [Bds::FileHeaderType](#) { [Bds::FileHeaderType_Standard](#) = 1 , [Bds::FileHeaderType_TapeDigitiser](#) = 10 }
- enum [Bds::FileSampleType](#) { [Bds::FileSampleType_Unknown](#) , [Bds::FileSampleType_Float32](#) , [Bds::FileSampleType_Float64](#) , [Bds::FileSampleType_Int16](#) , [Bds::FileSampleType_Int32](#) }

Variables

- const double `Bds::Scale` = 16777216.0

8.74 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.d File Reference

8.75 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.h File Reference

```
#include <BTypes.h>
#include <BError.h>
#include <BFile.h>
#include <BEntry.h>
#include <BBuffer.h>
#include <BDict.h>
#include <BdsDataFile.h>
```

Classes

- class `Bds::DataFileTapeDigitiser`

This class implements the TapeDigitiser's file output conversion and storing system.

Namespaces

- namespace `Bds`

8.76 BdsDataFileTapeDigitiser.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileTapeDigitiser.h DataFileTapeDigitiser
3  * T.Barnaby, BEAM Ltd, 2006-03-17
4  *****/
5 */
6 #ifndef BdsDataFileTapeDigitiser_H
7 #define BdsDataFileTapeDigitiser_H 1
8
9 #include <BTypes.h>
10 #include <BError.h>
11 #include <BFile.h>
12 #include <BEntry.h>
13 #include <BBuffer.h>
14 #include <BDict.h>
15 #include <BdsDataFile.h>
16
17 namespace Bds {
18
19 class DataFileTapeDigitiser : public DataFile {
20 public:
21     DataFileTapeDigitiser();
22
23     BError open(BString fileName, BString mode);
24
25     // Read routines
26 }
```

```

27     BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
28     BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
29
30     static DataFormats getFormats();
31 private:
32     uint32_t         blockSize();
33     uint32_t         blockNumSamples();
34     BError           readHeader(BDictString& header);
35     uint32_t         computeChecksum(void* data, int nBytes);
36
37     BDictString       oheader;
38     off64_t          oheaderSize;
39     uint32_t          oblockSize;
40     uint32_t          oblockNumSamples;
41     double            osampleRate;
42     BBuffer           obuffer;
43     BString           overision;
44     DataInfo          odataInfo;
45 };
46
47 }
48 #endif

```

8.77 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.cpp File Reference

```

#include <BdsDataFileWra.h>
#include <BDebug.h>
#include <errno.h>

```

Namespaces

- namespace [Bds](#)

8.78 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.d File Reference

8.79 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.h File Reference

```

#include <BdsDataFile.h>

```

Classes

- class [Bds::DataFileWra](#)
Data file convertor for WRA format files.

Namespaces

- namespace [Bds](#)

8.80 BdsDataFileWra.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataFileWra.h   BDRS Data File Access
3  *   T.Barnaby, BEAM Ltd, 2009-05-11
4  *****/
5  */
6  #ifndef BdsDataFileWra_H
7  #define BdsDataFileWra_H
8
9  #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     class DataFileWra : public DataFile{
14     public:
15         DataFileWra();
16
17         BError          setFormat(BString format);
18         int             getFeatures();
19         DataOrder        getDataOrder();
20         BString          getFixesInfo();
21
22         BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
23         BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
24
25         static DataFormats getFormats();
26     private:
27         BError          readBlock(BUInt32 channel, BUInt64 pos, DataBlock& data);
28
29         int             omagic;
30         int             oyear;
31         int             oblockYear;
32         BUInt32         oblockSize;
33         BUInt32         onumChannels;
34         double           osampleRate;
35         DataInfo         odataInfo;
36         BArray<BUInt64> oblockPositions;
37     };
38 };
39
40 }
41 #endif

```

8.81 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.cpp File Reference

```

#include <BdsDataFileWraAgso.h>
#include <BdsCompress.h>
#include <BDebug.h>
#include <errno.h>
#include <math.h>

```

Namespaces

- namespace [Bds](#)

Functions

- static **BList< BString >** [parseStringFixedFields](#) (**BString** s, int *fieldWidths)

8.81.1 Function Documentation

8.81.1.1 parseStringFixedFields()

```
static BList< BString > parseStringFixedFields (
    BString s,
    int * fieldWidths ) [static]
```

8.82 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.d File Reference

8.83 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.h File Reference

```
#include <BdsDataFile.h>
```

Classes

- class [Bds::DataFileWraAgso](#)
Data file convertor for WRA AGSO format files.

Namespaces

- namespace [Bds](#)

8.84 BdsDataFileWraAgso.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileWraAgso.h   BDRS Data File Access
3  *       T.Barnaby,  BEAM Ltd,   2009-05-11
4  *****/
5 */
6 #ifndef BdsDataFileWraAgso_H
7 #define BdsDataFileWraAgso_H
8
9 #include <BdsDataFile.h>
10
11 namespace Bds {
12
13     class DataFileWraAgso : public DataFile{
14     public:
15         DataFileWraAgso();
16
17         int         getFeatures();
18         DataOrder   getDataOrder();
19
20         BError       getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
21         BError       readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
22
23         static DataFormats   getFormats();
24     private:
25         BError       readBlock(BUInt32 blockNumber, BString& channelheader, DataBlock& data);
26         int          oyear;
27         int          oblockYear;
28         BUInt32      oblockSize;
29         double       osampleRate;
30         DataInfo     odataInfo;
31         BArray<BUInt> ochannelStartBlocks;
32         BUInt        oreadChannel;
33         BUInt        oreadBlock;
34         BUInt        ocurrentBlock;
35         static int    ofieldWidths[];
36     };
37 };
38
39 }
40 #endif
```

8.85 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.cpp File Reference

```
#include <BdsDataLib.h>
#include <BdsDataFileAscii.h>
#include <BdsDataFileBknas.h>
#include <BdsDataFileBdrs.h>
#include <BdsDataFileBds.h>
#include <BdsDataFileGcf.h>
#include <BdsDataFileIms.h>
#include <BdsDataFileTapeDigitiser.h>
#include <BdsDataFileWra.h>
#include <BdsDataFileWraAgso.h>
#include <BdsDataFileSeed.h>
#include <BdsDataFileSac.h>
#include <BdsDataFileCd.h>
#include <BdsDataFileResponse.h>
#include <BdsDataFileLog.h>
#include <BdsDataFileAd22.h>
#include <BdsDataFileLac.h>
#include <BdsDataFileCss.h>
#include <BdsDataFileStationXml.h>
#include <BdsDataFileIdc.h>
```

Namespaces

- namespace [Bds](#)

Functions

- **BString** [Bds::fixedWidthValue](#) (double v, int width)
This returns a double as a fixed width string truncating the data.

Variables

- DataFormatAll [Bds::dataFormatAll](#)

8.86 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.d File Reference

8.87 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.h File Reference

```
#include <BdsDataFile.h>
```

Classes

- class [Bds::DataFormatAll](#)
This class defines the interface for generic data file access.

Namespaces

- namespace [Bds](#)

Functions

- **BString** [Bds::fixedWidthValue](#) (double v, int width)
This returns a double as a fixed width string truncating the data.

8.88 BdsDataLib.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsDataLib.h   Data File Access
3  *   T.Barnaby,  BEAM Ltd,  2008-11-20
4  *****/
5  */
6  #ifndef BdsDataLib_H
7  #define BdsDataLib_H
8
9  #include <BdsDataFile.h>
10
11 namespace Bds {
12
13 BString fixedWidthValue(double v, int width);
14
15 class DataFormatAll {
16 public:
17     DataFormatAll();
18     ~DataFormatAll();
19
20     BError formatList(DataFormats& formats);
21     BError formatGet(BString format, DataFile*& dataFile, DataFormatSet formatSet =
22         DataFormatSetNone);
23     BString formatGetExtension(BString format);
24
25 protected:
26     int findFormat(DataFormats dataFormats, BString string, DataFormatSet formatSet);
27 };
28
29 extern DataFormatAll dataFormatAll;
30
31 }
32 #endif

```

8.89 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.cpp

File Reference

```

#include <BdsDataFileSeed.h>
#include <BEndian.h>
#include <errno.h>
#include <BDebug.h>
#include <libmseed.h>

```

Namespaces

- namespace [Bds](#)

Macros

- `#define BDEBUGL1 0`
- `#define BDEBUGL2 0`
- `#define BDEBUGL3 0`
- `#define DEBUG 0`
- `#define DEBUG_BLOCKETTE 0`
- `#define DEBUG_BLOCKS 0`
- `#define FILL_BLOCKS 1`
- `#define ROUND_TIMESTAMPS_US 10`

Functions

- static double `Bds::roundDigits` (double v, int nDigits)
- static void `Bds::bdsDataFileSeedLogWarning` (char *str)
- static void `Bds::bdsDataFileSeedLogError` (char *str)
- static `hptime_t` `Bds::seedTime` (`BTimeStamp` t)
- static `BString` `Bds::seedTimeString` (`BTimeStamp` t)
- static `BTimeStamp` `Bds::fromSeedTimeString` (`BString` str)
- static void `Bds::dataConvert` (const `BArray`< `BFloat64` > &dataIn, `BArray`< `BInt32` > &dataOut)
- static void `Bds::dataConvert` (const `BArray`< `BFloat64` > &dataIn, `BArray`< `BFloat32` > &dataOut)
- static void `Bds::record_handler` (char *record, int reclen, void *info)

8.89.1 Macro Definition Documentation

8.89.1.1 BDEBUGL1

```
#define BDEBUGL1 0
```

8.89.1.2 BDEBUGL2

```
#define BDEBUGL2 0
```

8.89.1.3 BDEBUGL3

```
#define BDEBUGL3 0
```

8.89.1.4 DEBUG

```
#define DEBUG 0
```

8.89.1.5 DEBUG_BLOCKETTE

```
#define DEBUG_BLOCKETTE 0
```

8.89.1.6 DEBUG_BLOCKS

```
#define DEBUG_BLOCKS 0
```

8.89.1.7 FILL_BLOCKS

```
#define FILL_BLOCKS 1
```

8.89.1.8 ROUND_TIMESTAMPS_US

```
#define ROUND_TIMESTAMPS_US 10
```

8.90 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.d File Reference

8.91 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.h File Reference

```
#include <BdsDataFile.h>  
#include <BdsSeedTypes.h>  
#include <BMutex.h>
```

Classes

- class [Bds::DataFileSeed](#)
Data file convertor for SEED file formats.

Namespaces

- namespace [Bds](#)

Typedefs

- typedef struct MSRecord_s [MSRecord](#)

8.91.1 Typedef Documentation

8.91.1.1 MSRecord

```
typedef struct MSRecord_s MSRecord
```

8.92 BdsDataFileSeed.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsDataFileSeed.h  SEED Data File Access
3  * T.Barnaby, BEAM Ltd, 2009-12-18
4  *****/
5 */
6 #ifndef BdsDataFileSeed_H
7 #define BdsDataFileSeed_H
8
9 #include <BdsDataFile.h>
10 #include <BdsSeedTypes.h>
11 #include <BMutex.h>
12
13 typedef struct MSRecord_s MSRecord; // Forward-declaration for libmseed type.
14
15 namespace Bds {
16
17     class DataFileSeed : public DataFile{
18     public:
19         DataFileSeed();
20         ~DataFileSeed();
21
22         BError          close();
23
24         DataOrder       getDataOrder();
25         int             getFeatures();
26         BString         getFixesInfo();
27         BError          setFormat(BString format);
28
29         // Read routines
30         BError          getInfo(DataInfo& dataInfo, DataFileOptions options, BList<DataError>& errors);
31         BError          readData(BUInt32 channel, BUInt segment, BUInt32 blockNumber, DataBlock& data);
32         BError          getMetaData(ChannelInfos& channelInfos, BUInt32 options, BList<DataError>& errors);
33
34         // Write routines
35         BError          setInfo(const DataInfo& dataInfo, const ChannelInfos& channelInfos,
36                               WriteOptionsList options = WriteOptionSensorData);
37         BError          start(BUInt channel, BUInt segment);
38         BError          writeData(const DataBlock& data);
39         BError          end();
40
41         void            msrFileWrite(void* data, int len);
42
43         static DataFormats getFormats();
44         static BError      omsrErr;
45         static int         onoLock;
46
47     private:
```

```

48     class NameDesc {
49     public:
50         NameDesc(BString name = "", BString desc = "") : name(name), desc(desc){}
51         BString      name;
52         BString      desc;
53     };
54
55     BError      processControl();
56     BError      processBlockette();
57     BError      processData(DataBlock& data, int addInfo);
58     BError      setBlockSize(BUInt blockSize);
59     BError      readBlockData(char* buf, BUInt numBytes);
60     BError      readBlock(BUInt64 blockPosition, char& type, char& cont, BUInt& seq);
61     BError      writeVolumeHeader(int rl);
62     BError      writeTimeSpans();
63     BError      writeBlockData(char type, const char* buf, BUInt numBytes);
64     BError      writeFlush();
65     int         getChannel(BString name);
66     void        sortChannels();
67     DataError    getBlockReorderInfo();
68
69     BUInt        addCode(BString str);
70     BUInt        addUnitCode(BString name, BString desc);
71     Response*    getResponse(BUInt stage);
72
73     DataFileOptions    oreadOptions;
74     DataInfo           odataInfo;
75     BString            onetwork;
76     ChannelInfos       ochannelInfos;
77     BArray<ChannelInfo> ochannelInfoSegments;
78     ChannelInfo        ochannelInfo;
79     double             osampleRate;
80     BArray<BArray<DataBlockPos> > oblockPositions;
81
82     BUInt32           oblockSize;
83     char*             oblock;
84     BUInt64           oblockPosition;
85     BUInt             opos;
86     BUInt             oblockNumberRead;
87     BUInt             oblockNumberWrite;
88     BUInt             oblocketteNumber;
89     BUInt             ologNumber;
90
91     // SEED information
92     BDictString       ocodes;
93     BDict<NameDesc>   ounitCodes;
94
95     // SEED import information
96     BdsSeedType10     otype10;
97     BdsSeedType11     otype11;
98     BdsSeedType30     otype30;
99     BdsSeedType52     otype52;
100    BArray<BdsSeedType30> oseedDataFormats;
101    BArray<BdsSeedStation> oseedStations;
102    BArray<BdsSeedType52> oseedChannels;
103
104    BArray<BString>       odataChannels;
105    BArray<int>           odataChannelFormats;
106
107    // SEED export information
108    BArray<BInt32>        odataInt32;
109    BArray<BFloat32>      odataFloat32;
110
111    BUInt                odataChannel;
112    BUInt                odataSegment;
113    BUInt                odataTimeSpanBlock;
114    BDict<int>           ostationBlockNumbers;
115    BArray<BArray<BUInt32> > odataStartBlocks;
116    BArray<BArray<BUInt32> > odataEndBlocks;
117
118    static BMutex        olock;
119    MSRecord*            omsr;
120
121    BUInt                oabbrevCode;
122    BUInt                oabbrevUnitsCode;
123 };
124
125 }
126 #endif

```

8.93 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.cpp File Reference

```
#include <BdsSeedType.h>
```

Namespaces

- namespace [Bds](#)

8.94 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.d File Reference

8.95 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.h File Reference

```
#include <BError.h>
```

Classes

- class [Bds::BdsSeedType](#)
BdsDataFileSeed internal parent for all SEED types.

Namespaces

- namespace [Bds](#)

8.96 BdsSeedType.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsSeedType.cpp    BDS Seed data file access library
3  *                    T.Barnaby, BEAM Ltd, 2009-12-14
4  *****/
5 */
6 #ifndef BdsSeedType_H
7 #define BdsSeedType_H
8
9 #include <BError.h>
10
11 namespace Bds {
12
13     class BdsSeedType {
14     public:
15         BdsSeedType();
16
17         BError    getInt(char** data, int size, int& v);
18         BError    getUInt(char** data, int size, unsigned int& v);
19         BError    getDouble(char** data, int size, double& v);
20         BError    getString(char** data, int size, BString& v);
21         BError    getStringVariable(char** data, int size, BString& v);
22
23         BError    appendInt(BString& s, int v, int size);
24         BError    appendDouble(BString& s, double v, int size, int precision);
25         BError    appendExp(BString& s, double v, int size, int precision, int sign);
26         BError    appendString(BString& s, BString v, int size);
27         BError    appendStringVariable(BString& s, BString v, int size);
28     };
29
30 }
31 #endif
```

8.97 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.cpp File Reference

```
#include <BdsSeedTypes.h>
```

Namespaces

- namespace [Bds](#)

8.98 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.d File Reference

8.99 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedTypes.idl File Reference

8.100 /src/blacknest/bds/bds/bdsDataLib/canada_compress.d File Reference

8.101 /src/blacknest/bds/bds/bdsDataLib/canada_compress.h File Reference

```
#include <arpa/inet.h>
```

Macros

- #define [CANCOMP_ERR](#) -1 /* unrecoverable **error** (malloc fails) */
- #define [CANCOMP_SUCCESS](#) 0 /* success */
- #define [CANCOMP_NOT_20](#) 1 /* **number** of samples not divisible by 20 */
- #define [CANCOMP_CORRUPT](#) 2 /* corrupted call */
- #define [CANCOMP_EXCEED](#)

Functions

- int [canada_uncompress](#) (unsigned char *b, uint32_t *y, int *n, int m, uint32_t *v0)
De-compressses Canada format seismic data.
- int [canada_compress](#) (unsigned char *b, unsigned long *y, int *n, int m, unsigned long *v0)
Compressses Canada format seismic data.

8.101.1 Macro Definition Documentation

8.101.1.1 CANCOMP_ERR

```
#define CANCOMP_ERR -1 /* unrecoverable error (malloc fails) */
```

8.101.1.2 CANCOMP_SUCCESS

```
#define CANCOMP_SUCCESS 0 /* success */
```

8.101.1.3 CANCOMP_NOT_20

```
#define CANCOMP_NOT_20 1 /* number of samples not divisible by 20 */
```

8.101.1.4 CANCOMP_CORRUPT

```
#define CANCOMP_CORRUPT 2 /* corrupted call */
```

8.101.1.5 CANCOMP_EXCEED

```
#define CANCOMP_EXCEED
```

Value:

```
3 /* number of bytes available in compressed  
data exceeded during decompression */
```

8.101.2 Function Documentation

8.101.2.1 canada_uncompress()

```
int canada_uncompress (  
    unsigned char * b,  
    uint32_t * y,  
    int * n,  
    int m,  
    uint32_t * v0 )
```

De-compresses Canada format seismic data.

8.101.2.2 canada_compress()

```
int canada_compress (
    unsigned char * b,
    unsigned long * y,
    int * n,
    int m,
    unsigned long * v0 )
```

Compresses Canada format seismic data.

8.102 canada_compress.h

[Go to the documentation of this file.](#)

```
1 /* Copyright 1994 Science Applications International Corporation
2  *
3  * This software may not be used, copied, modified, or distributed without
4  * the express written permission of Science Applications International
5  * Corporation (SAIC). SAIC makes no warranty of any kind with regard
6  * to this software, including, but not limited to, the implied warranties
7  * of fitness for a particular purpose.
8  */
9
10 /* #pragma ident "(#)canada_compress.h 1.1 05/18/00    SAIC" */
11
12 #include <arpa/inet.h>
13
14 #ifndef CANADACOMP_H
15 #define CANADACOMP_H
16
17 #define CANCOMP_ERR -1 /* unrecoverable error (malloc fails) */
18 #define CANCOMP_SUCCESS 0 /* success */
19 #define CANCOMP_NOT_20 1 /* number of samples not divisible by 20 */
20 #define CANCOMP_CORRUPT 2 /* corrupted call */
21 #define CANCOMP_EXCEED 3 /* number of bytes available in compressed
22    data exceeded during decompression */
23
24
26 int canada_uncompress(unsigned char *b, uint32_t *y, int *n, int m,
27    uint32_t *v0);
28
30 int canada_compress(unsigned char *b, unsigned long *y, int *n, int m,
31    unsigned long *v0);
32
33 #endif /* CANADACOMP_H */
```

8.103 BdsC.cc File Reference

```
#include <BdsC.h>
```

Namespaces

- namespace [Bds](#)

8.104 BdsC.d File Reference

8.105 BdsC.h File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <Boap.h>
#include <BString.h>
#include <BList.h>
#include <BArray.h>
#include <BdsD.h>
```

Classes

- class [Bds::DataAccess](#)
This is the Data Access API interface to the BDS system.
- class [Bds::DataAddAccess](#)
This is the DataAdd Access API interface.
- class [Bds::AdminAccess](#)
This is the [AdminAccess](#) Access API interface.

Namespaces

- namespace [Bds](#)

Variables

- const **BUInt32** [Bds::apiVersion](#) = 0

8.106 BdsC.h

[Go to the documentation of this file.](#)

```
1 /*****
2  * BdsC.h Produced by Bidl
3  *****/
4 */
5
6 #ifndef BDSC_H
7 #define BDSC_H 1
8
9 #include <stdlib.h>
10 #include <stdint.h>
11 #include <Boap.h>
12 #include <BString.h>
13 #include <BList.h>
14 #include <BArray.h>
15 #include <BdsD.h>
16
17 namespace Bds {
18     const BUInt32 apiVersion = 0;
19
20     class DataAccess : public BoapClientObject {
21     public:
22         DataAccess(BString name = "");
```

```

29     BError connect(BString user, BString password);
30     BError validateUser(BString user, BString email);
31     BError setUser(BString user, BString email);
32     BError setUserReal();
33     BError getVersion(BString& version, BString& name);
34     // User functions
35     BError userGetFromId(BUInt32 id, User& user);
36     BError userGet(User& user);
37     BError userSet(User user);
38     BError userGetGroups(BList<BString> >& groups);
39     BError userGetOptions(BDict<BString> >& items);
40     BError userSetOptions(BDict<BString> >& items);
41     BError groupGetList(BList<Group> >& groups);
42     // Information functions
43     BError networkGetList(BList<Network> >& networks);
44     BError stationGetList(Selection sel, BList<Station> >& stations);
45     BError channelGetList(Selection sel, BList<Channel> >& channels);
46     BError sourceGetList(BList<Source> >& sources);
47     BError sourcePriorityGetList(BList<SourcePriority> >& sourcePriorities);
48     BError dataFileGetList(Selection sel, BList<DataFileInfo> >& dataFile);
49     BError dataChannelGetList(Selection sel, BList<DataChannel> >& dataChannel);
50     BError channelInstrumentGetList(Selection sel, BList<ChannelInstrument> >& channelInstruments);
51     BError digitiserGetList(Selection sel, BList<Digitiser> >& digitisers);
52     BError digitiserGet(BUInt32 id, Digitiser& digitiser);
53     BError sensorGetList(Selection sel, BList<Sensor> >& sensors);
54     BError sensorGet(BUInt32 id, Sensor& sensor);
55     BError calibrationGetList(Selection sel, BList<Calibration> >& calibrations);
56     BError responseGetList(Selection sel, BList<Response> >& responses);
57     BError locationGetList(Selection sel, BList<Location> >& locations);
58     BError eventGetList(Selection sel, BList<Event> >& events);
59     BError specialChannelGetList(Selection sel, BList<SpecialChannel> >& specialChannels);
60     BError metadataGetChannelInfo(Selection sel, ChannelInfos& channelInfos);
61     BError metadataGetFormatted(Selection sel, BString format, BArray<BUInt8> >& data);
62     // Selections
63     BError getSelectionInfo(SelectionGroup group, Selection selectionIn, SelectionInfo&
selectionInfo);
64     BError getSelections(SelectionGroup group, Selection selectionIn, Selection& selectionOut);
65     // Data access functions
66     BError dataAvailability(Selection selection, BUInt32 num, BArray<DataAvailChan> >&
dataAvailChans);
67     BError dataSearch(Selection selection, DataInfo& dataInfo);
68     BError dataGetChannelInfo(DataInfo dataInfo, ChannelInfos& channelInfos);
69     BError dataOpen(DataInfo dataInfo, BString mode, BString format, BUInt32 flags, DataHandle&
dataHandle);
70     BError dataGetInfo(DataHandle dataHandle, BUInt32 infoExtra, DataInfo& dataInfo);
71     BError dataGetNotes(DataHandle dataHandle, BList<Note> >& notes);
72     BError dataGetWarnings(DataHandle dataHandle, BList<BString> >& warnings);
73     BError dataSeekBlock(DataHandle dataHandle, BUInt32 channel, BUInt32 segment, BTimeStamp time,
BUInt32& blockNumber);
74     BError dataGetBlock(DataHandle dataHandle, BUInt32 channel, BUInt32 segment, BUInt32 blockNumber,
DataBlock& data);
75     BError dataClose(DataHandle dataHandle, BError error, BInt32 del);
76     BError dataFormattedRead(DataHandle dataHandle, BUInt32 number, BArray<BUInt8> >& data);
77     BError dataFormattedGetLength(DataHandle dataHandle, BUInt64& length);
78     // Real-time data functions
79     BError dataRealtimeConfig(BInt32 enable, Selection sel);
80     BError dataRealtimeGet(BUInt32 numBlocks, BUInt32& numBlocksAvailable, BList<DataBlockChannel> >&
dataBlocks);
81     BError noteGetList(Selection sel, BList<Note> >& notes);
82     BError noteUpdate(BInt32 append, Note note, BUInt32& id);
83     BError noteWriteDocument(BUInt32 id, BString format, BArray<BUInt8> > data);
84     BError noteReadDocument(BUInt32 id, BString& format, BArray<BUInt8> >& data);
85     BError logUpdate(BInt32 append, Log log, BUInt32& id);
86     BError logAppend(BString type, BUInt32 priority, BString subSystem, BString title, BString
description);
87     // Management functions
88     BError modeSet(Mode mode, Mode& previousMode);
89     BError modeSnapshotPause(BInt32 on);
90     BError clean(CleanOptions cleanOptions);
91     BError databaseBackup(BString& ref);
92     // Information functions
93     BError statisticsGet(BDict<BString> >& info);
94     BError serverConfigurationGet(BDict<BString> >& items);
95     BError dataFormatGetList(BList<DataFormat> >& formats);
96 private:
97 };
103 class DataAddAccess : public BoapClientObject {
104 public:
105     DataAddAccess(BString name = "");
106     BError connect(BString user, BString password);
107     BError validateUser(BString user, BString email);
108     BError setUser(BString user, BString email);
109     BError setUserReal();
110     BError getVersion(BString& version, BString& name);
111     // User functions
112     BError userGetFromId(BUInt32 id, User& user);
113     BError userGet(User& user);

```

```

114     BError userSet(User user);
115     BError userGetGroups(BList<BString >& groups);
116     BError userGetOptions(BDict<BString >& items);
117     BError userSetOptions(BDict<BString >& items);
118     BError groupGetList(BList<Group >& groups);
119     // Information functions
120     BError networkGetList(BList<Network >& networks);
121     BError stationGetList(Selection sel, BList<Station >& stations);
122     BError channelGetList(Selection sel, BList<Channel >& channels);
123     BError sourceGetList(BList<Source >& sources);
124     BError sourcePriorityGetList(BList<SourcePriority >& sourcePriorities);
125     BError dataFileGetList(Selection sel, BList<DataFileInfo >& dataFile);
126     BError dataChannelGetList(Selection sel, BList<DataChannel >& dataChannel);
127     BError channelInstrumentGetList(Selection sel, BList<ChannelInstrument >& channelInstruments);
128     BError digitiserGetList(Selection sel, BList<Digitiser >& digitisers);
129     BError digitiserGet(BUInt32 id, Digitiser& digitiser);
130     BError sensorGetList(Selection sel, BList<Sensor >& sensors);
131     BError sensorGet(BUInt32 id, Sensor& sensor);
132     BError calibrationGetList(Selection sel, BList<Calibration >& calibrations);
133     BError responseGetList(Selection sel, BList<Response >& responses);
134     BError locationGetList(Selection sel, BList<Location >& locations);
135     BError eventGetList(Selection sel, BList<Event >& events);
136     BError eventUpdate(BInt32 append, Event event, BUInt32& id);
137     BError eventDelete(BUInt32 id);
138     BError specialChannelGetList(Selection sel, BList<SpecialChannel >& specialChannels);
139     BError metadataGetChannelInfo(Selection sel, ChannelInfos& channelInfos);
140     BError metadataGetFormatted(Selection sel, BString format, BArray<BUInt8 >& data);
141     // Selections
142     BError getSelectionInfo(SelectionGroup group, Selection selectionIn, SelectionInfo&
selectionInfo);
143     BError getSelections(SelectionGroup group, Selection selectionIn, Selection& selectionOut);
144     // Data access functions
145     BError dataAvailability(Selection selection, BUInt32 num, BArray<DataAvailChan >&
dataAvailChans);
146     BError dataSearch(Selection selection, DataInfo& dataInfo);
147     BError dataGetChannelInfo(DataInfo dataInfo, ChannelInfos& channelInfos);
148     BError dataOpen(DataInfo dataInfo, BString mode, BString format, BUInt32 flags, DataHandle&
dataHandle);
149     BError dataGetInfo(DataHandle dataHandle, BUInt32 infoExtra, DataInfo& dataInfo);
150     BError dataGetNotes(DataHandle dataHandle, BList<Note >& notes);
151     BError dataGetWarnings(DataHandle dataHandle, BList<BString >& warnings);
152     BError dataSeekBlock(DataHandle dataHandle, BUInt32 channel, BUInt32 segment, BTimeStamp time,
BUInt32& blockNumber);
153     BError dataGetBlock(DataHandle dataHandle, BUInt32 channel, BUInt32 segment, BUInt32
blockNumber, DataBlock& data);
154     BError dataSetInfo(DataHandle dataHandle, DataInfo dataInfo);
155     BError dataPutBlock(DataHandle dataHandle, DataBlock data);
156     BError dataClose(DataHandle dataHandle, BError error, BInt32 del);
157     BError dataFormattedRead(DataHandle dataHandle, BUInt32 number, BArray<BUInt8 >& data);
158     BError dataFormattedGetLength(DataHandle dataHandle, BUInt64& length);
159     // Real-time data functions
160     BError dataRealtimeConfig(BInt32 enable, Selection sel);
161     BError dataRealtimeGet(BUInt32 numBlocks, BUInt32& numBlocksAvailable, BList<DataBlockChannel >&
dataBlocks);
162     BError noteGetList(Selection sel, BList<Note >& notes);
163     BError noteUpdate(BInt32 append, Note note, BUInt32& id);
164     BError noteWriteDocument(BUInt32 id, BString format, BArray<BUInt8 > data);
165     BError noteReadDocument(BUInt32 id, BString& format, BArray<BUInt8 >& data);
166     BError logUpdate(BInt32 append, Log log, BUInt32& id);
167     BError logAppend(BString type, BUInt32 priority, BString subSystem, BString title, BString
description);
168     // Management functions
169     BError modeSet(Mode mode, Mode& previousMode);
170     BError modeSnapshotPause(BInt32 on);
171     BError clean(CleanOptions cleanOptions);
172     BError databaseBackup(BString& ref);
173     // Information functions
174     BError statisticsGet(BDict<BString >& info);
175     BError serverConfigurationGet(BDict<BString >& items);
176     BError dataFormatGetList(BList<DataFormat >& formats);
177 private:
178 };
179
180 class AdminAccess : public BoapClientObject {
181 public:
182     AdminAccess(BString name = "");
183     BError connect(BString user, BString password);
184     BError validateUser(BString user, BString email);
185     BError setUser(BString user, BString email);
186     BError setUserReal();
187     BError getVersion(BString& version, BString& name);
188     // User functions
189     BError userGetList(BList<User >& users);
190     BError userUpdate(BInt32 append, User user, BUInt32& id);
191     BError userDelete(BUInt32 id);
192     BError userGetFromId(BUInt32 id, User& user);
193     BError userGet(User& user);
194     BError userSet(User user);

```

```

198     BError userGetGroups(BList<BString> & groups);
199     BError userGetOptions(BDict<BString> & items);
200     BError userSetOptions(BDict<BString> & items);
201     BError groupGetList(BList<Group> & groups);
202     BError groupUpdate(BInt32 append, Group group, BUInt32& id);
203     BError groupDelete(BUInt32 id);
204     BError accessGroupGetList(BList<AccessGroup> & accessGroups);
205     BError accessGroupUpdate(BInt32 append, AccessGroup group, BUInt32& id);
206     BError accessGroupDelete(BUInt32 id);
207     // Selections
208     BError getSelectionInfo(SelectionGroup group, Selection selectionIn, SelectionInfo&
selectionInfo);
209     BError getSelections(SelectionGroup group, Selection selectionIn, Selection& selectionOut);
210     BError networkGetList(BList<Network> & networks);
211     BError networkUpdate(BInt32 append, Network network, BUInt32& id);
212     BError networkDelete(BUInt32 id);
213     BError stationGetList(Selection sel, BList<Station> & stations);
214     BError stationUpdate(BInt32 append, Station station, BUInt32& id);
215     BError stationDelete(BUInt32 id);
216     BError locationGetList(Selection sel, BList<Location> & locations);
217     BError locationUpdate(BInt32 append, Location location, BUInt32& id);
218     BError locationDelete(BUInt32 id);
219     BError channelGetList(Selection sel, BList<Channel> & channels);
220     BError channelGet(BUInt32 id, Channel& channel);
221     BError channelUpdate(BInt32 append, Channel channel, BUInt32& id);
222     BError channelDelete(BUInt32 id);
223     BError sourceGetList(BList<Source> & sources);
224     BError sourceUpdate(BInt32 append, Source source, BUInt32& id);
225     BError sourceDelete(BUInt32 id);
226     BError sourcePriorityGetList(BList<SourcePriority> & sourcePriorities);
227     BError sourcePriorityUpdate(BInt32 append, SourcePriority sourcePriority, BUInt32& id);
228     BError sourcePriorityDelete(BUInt32 id);
229     BError channelInstrumentGetList(Selection sel, BList<ChannelInstrument> & channelInstruments);
230     BError channelInstrumentUpdate(BInt32 append, ChannelInstrument channelInstrument, BUInt32& id);

231     BError channelInstrumentDelete(BUInt32 id);
232     BError digitiserGetList(Selection sel, BList<Digitiser> & digitisers);
233     BError digitiserGet(BUInt32 id, Digitiser& digitiser);
234     BError digitiserUpdate(BInt32 append, Digitiser digitiser, BUInt32& id);
235     BError digitiserDelete(BUInt32 id);
236     BError sensorGetList(Selection sel, BList<Sensor> & sensors);
237     BError sensorGet(BUInt32 id, Sensor& sensor);
238     BError sensorUpdate(BInt32 append, Sensor sensor, BUInt32& id);
239     BError sensorDelete(BUInt32 id);
240     BError calibrationGetList(Selection sel, BList<Calibration> & calibrations);
241     BError calibrationUpdate(BInt32 append, Calibration calibration, BUInt32& id);
242     BError calibrationDelete(BUInt32 id);
243     BError responseGetList(Selection sel, BList<Response> & responses);
244     BError responseUpdate(BInt32 append, Response response, BUInt32& id);
245     BError responseDelete(BUInt32 id);
246     BError eventGetList(Selection sel, BList<Event> & events);
247     BError eventUpdate(BInt32 append, Event event, BUInt32& id);
248     BError eventDelete(BUInt32 id);
249     BError specialChannelGetList(Selection sel, BList<SpecialChannel> & specialChannels);
250     BError specialChannelUpdate(BInt32 append, SpecialChannel specialChannel, BUInt32& id);
251     BError specialChannelDelete(BUInt32 id);
252     BError metadataGetChannelInfo(Selection sel, ChannelInfos& channelInfos);
253     BError metadataGetFormatted(Selection sel, BString format, BArray<BUInt8> & data);
254     // Data access functions
255     BError dataFileGetList(Selection sel, BList<DataFileInfo> & dataFile);
256     BError dataFileUpdate(BInt32 append, DataFileInfo dataFile, BUInt32& id);
257     BError dataFileDelete(BUInt32 id);
258     BError dataChannelGetList(Selection sel, BList<DataChannel> & dataChannel);
259     BError dataChannelUpdate(BInt32 append, DataChannel dataChannel, BUInt32& id);
260     BError dataChannelDelete(BUInt32 id);
261     BError dataAvailability(Selection selection, BUInt32 num, BArray<DataAvailChan> &
dataAvailChans);
262     BError dataSearch(Selection selection, DataInfo& dataInfo);
263     BError dataGetChannelInfo(DataInfo dataInfo, ChannelInfos& channelInfos);
264     BError dataOpen(DataInfo dataInfo, BString mode, BString format, BUInt32 flags, DataHandle&
dataHandle);
265     BError dataGetInfo(DataHandle dataHandle, BUInt32 infoExtra, DataInfo& dataInfo);
266     BError dataGetNotes(DataHandle dataHandle, BList<Note> & notes);
267     BError dataGetWarnings(DataHandle dataHandle, BList<BString> & warnings);
268     BError dataGetBlock(DataHandle dataHandle, BUInt32 channel, BUInt32 segment, BUInt32
blockNumber, DataBlock& data);
269     BError dataSeekBlock(DataHandle dataHandle, BUInt32 channel, BUInt32 segment, BTimeStamp time,
BUInt32& blockNumber);
270     BError dataSetInfo(DataHandle dataHandle, DataInfo dataInfo);
271     BError dataPutBlock(DataHandle dataHandle, DataBlock data);
272     BError dataClose(DataHandle dataHandle, BError error, BInt32 del);
273     BError dataFormattedRead(DataHandle dataHandle, BUInt32 number, BArray<BUInt8> & data);
274     BError dataFormattedGetLength(DataHandle dataHandle, BUInt64& length);
275     // Real-time data functions
276     BError dataRealtimeConfig(BInt32 enable, Selection sel);
277     BError dataRealtimeGet(BUInt32 numBlocks, BUInt32& numBlocksAvailable, BList<DataBlockChannel> &
dataBlocks);

```

```

278      // Change/Logging functions
279      BError changeGroupStart(ChangeGroup changeGroup);
280      BError changeGroupEnd();
281      BError changeGroupGetList(ListRange range, BList<ChangeGroup> & changeGroups);
282      BError changeGroupDelete(BTimeStamp beforeDate, BString type, BInt32 empty);
283      BError changeGetListNumber(BUInt32 id, BUInt32& number);
284      BError changeGetList(BUInt32 id, ListRange range, BList<Change> & changes);
285      BError changeDelete(BTimeStamp beforeDate, BString type);
286      BError noteGetList(Selection sel, BList<Note> & notes);
287      BError noteUpdate(BInt32 append, Note note, BUInt32& id);
288      BError noteDelete(BUInt32 id);
289      BError noteWriteDocument(BUInt32 id, BString format, BArray<BUInt8> data);
290      BError noteReadDocument(BUInt32 id, BString& format, BArray<BUInt8> & data);
291      BError logGetList(LogSelect sel, BList<Log> & logs);
292      BError logUpdate(BInt32 append, Log log, BUInt32& id);
293      BError logDelete(BUInt32 id);
294      BError logAppend(BString type, BUInt32 priority, BString subSystem, BString title, BString
description);
295      // Information functions
296      BError statisticsGet(BDict<BString> & info);
297      BError serverConfigurationGet(BDict<BString> & items);
298      BError dataFormatGetList(BList<DataFormat> & formats);
299      // Management functions
300      BError transactionStart();
301      BError transactionEnd(BInt32 abort);
302      BError modeSet(Mode mode, Mode& previousMode);
303      BError modeSnapshotPause(BInt32 on);
304      BError clean(CleanOptions cleanOptions);
305      BError databaseBackup(BString& ref);
306      BError databaseRestore(BString ref, BString type);
307      // Low level functions
308      BError sqlQuery(BString query, BList<BDict<BString>> & result);
309      BError extraCall(BUInt32 function, BString args, BString& result);
310  private:
311  };
312 }
313 #endif

```

8.107 BdsD.cc File Reference

```
#include <BdsD.h>
```

Namespaces

- namespace [Bds](#)

8.108 BdsD.d File Reference

8.109 BdsD.h File Reference

BOAP data class definitions for: [Bds](#).

```

#include <Boap.h>
#include <BObj.h>
#include <BDate.h>
#include <BTimeStamp.h>
#include <BComplex.h>
#include <BList.h>
#include <BArray.h>

```

Classes

- class [Bds::Point](#)
This class defines an X,Y location.
- class [Bds::TimePeriod](#)
This class defines a [TimePeriod](#).
- class [Bds::ListRange](#)
This class defines an integer based range.
- class [Bds::Network](#)
This class defines a seismic [Network](#) organisation.
- class [Bds::Source](#)
This class defines a seismic data [Source](#).
- class [Bds::SourcePriority](#)
This class defines a [Source](#) Priority entry.
- class [Bds::ChannelName](#)
This class defines a full channel name.
- class [Bds::ArrayChannel](#)
This class defines an array's channel.
- class [Bds::Station](#)
This class defines a seismic station.
- class [Bds::Location](#)
This class defines the physical location of a [Station](#).
- class [Bds::PoleZero](#)
This class defines a Pole/Zero [Response](#).
- class [Bds::Fap](#)
This class defines an entry in an Amplitude/Phase [Response](#) table.
- class [Bds::FirEntry](#)
This class defines an entry in a FIR coefficient table.
- class [Bds::Fir](#)
This class defines an FIR response table.
- class [Bds::PolynomialEntry](#)
This class defines an entry in a [Polynomial](#) coefficient table.
- class [Bds::Polynomial](#)
This class defines an [Polynomial](#) response table.
- class [Bds::Response](#)
This class defines a seismic [Response](#) characteristic.
- class [Bds::Calibration](#)
This class defines a calibration setting.
- class [Bds::Digitiser](#)
This class defines a seismic [Digitiser](#).
- class [Bds::Sensor](#)
This class defines a seismic [Sensor](#).
- class [Bds::ChannelInstrument](#)
This class defines a [Channel](#)'s instrument.
- class [Bds::Channel](#)
This class defines a seismic data [Channel](#).
- class [Bds::SelectionInfo](#)
This class defines the set of Metadata or Siesmic sensor data to be selected when `getSelectionInfo()` is use.
- class [Bds::SelectionChannel](#)
This class defines an individual channel for selection.
- class [Bds::Selection](#)

- This class defines a generic Metadata or [Sensor](#) data selection.*

 - class [Bds::ChannelInfo](#)

This class provides full Metadata information on a channel.
 - class [Bds::ChannelInfos](#)

This class provides Metadata information on a set of channels.
 - class [Bds::DataFileInfo](#)

This class defines information on a [Sensor](#) data file.
 - class [Bds::DataChannel](#)

This class defines information on a single channel's set of data stored in a file.
 - class [Bds::DataInfo](#)

This class defines information on a set of data.
 - class [Bds::DataAvail](#)

This class provides availability information on a particular period of data.
 - class [Bds::DataAvailChan](#)

This class defines availability information on a set of data.
 - class [Bds::DataHandle](#)

This defines a handle to a sensor data stream/file when opened for read or write.
 - class [Bds::DataBlock](#)

This class provides the actual [Sensor](#) data values contained within a single data block.
 - class [Bds::DataBlockChannel](#)

This class provides the actual seismic data values contained within a single data block along with the network↔:station:channel:source information.
 - class [Bds::User](#)

This holds information on a user.
 - class [Bds::Group](#)

This holds information on a [User](#) security group.
 - class [Bds::AccessGroup](#)

This holds information on data access groups.
 - class [Bds::Change](#)

This holds information on a Metadata or [Sensor](#) data database change.
 - class [Bds::ChangeGroup](#)

This holds information on a set of Changes.
 - class [Bds::Note](#)

This holds information on a [Note](#) for general information.
 - class [Bds::Log](#)

This holds information on a [Log](#) entry.
 - class [Bds::LogSelect](#)

This defines the selection criteria when requesting a set of log entries.
 - class [Bds::CleanOptions](#)

This defines the set of clean options used in the `clean()` function.
 - class [Bds::DataFormat](#)

This holds information on a [Sensor](#) data format.
 - class [Bds::SpecialChannel](#)

A Special channel identifier.
 - class [Bds::Event](#)

This class defines a seismic event.

Namespaces

- namespace [Bds](#)

Typedefs

- typedef **BList**< DataFormat > **Bds::DataFormats**

A list of the available [Sensor](#) data formats.

Enumerations

- enum **Bds::Errors** {
Bds::ErrorNoMetaData = 64 , **Bds::ErrorDataQuality** = 65 , **Bds::ErrorSlaveMode** = 66 , **Bds::ErrorTimeStamp** = 67 ,
Bds::ErrorValidate = 80 , **Bds::ErrorValidateMissingBlocks** = 81 , **Bds::ErrorValidateTimeBackwards** = 82 ,
Bds::ErrorValidateFilenameTime = 83 ,
Bds::ErrorValidateMetaData = 84 , **Bds::ErrorValidateFix** = 85 , **Bds::ErrorValidateDuplicate** = 86 ,
Bds::ErrorValidateReorder = 87 ,
Bds::ErrorValidateBdsFudge = 88 }
The System Error number list in addition to standard system error numbers.
- enum **Bds::Priority** { **Bds::PriorityLow** , **Bds::PriorityNormal** , **Bds::PriorityHigh** }
Priority levels.
- enum **Bds::Mode** { **Bds::ModeMaster** , **Bds::ModeSlave** }
BdsServer mode.
- enum **Bds::DataFlags** {
Bds::DataFlagNone = 0x00 , **Bds::DataFlagClipDataToTime** = 0x01 , **Bds::DataFlagClipDataToChannels** = 0x02 , **Bds::DataFlagMergeSegments** = 0x04 ,
Bds::DataFlagNoMetadata = 0x08 }
Flags when opening data files.
- enum **Bds::SelectionGroup** { **Bds::SelectionGroupData** , **Bds::SelectionGroupMetaData** , **Bds::SelectionGroupDataWithCount** }
The Selection group when making selections.
- enum **Bds::SampleFormat** {
Bds::SampleFormatUnknown , **Bds::SampleFormatInt16** , **Bds::SampleFormatInt32** , **Bds::SampleFormatFloat32** ,
Bds::SampleFormatFloat64 , **Bds::SampleFormatInt24** }
The actual format of a data sample.
- enum **Bds::AvailType** { **Bds::AvailNone** , **Bds::AvailPartial** , **Bds::AvailFull** }
A flag defining the data availability state.
- enum **Bds::DataFormatSet** {
Bds::DataFormatSetNone = 0x00 , **Bds::DataFormatSetMetadataRead** = 0x01 , **Bds::DataFormatSetMetadataWrite** = 0x02 , **Bds::DataFormatSetSensordataRead** = 0x04 ,
Bds::DataFormatSetSensordataWrite = 0x08 }
Data format abilities bitset.
- enum **Bds::LocationSelect** { **Bds::LocationSelectAll** , **Bds::LocationSelectStation** , **Bds::LocationSelectChannel** }
Which Locations to select.

8.109.1 Detailed Description

BOAP data class definitions for: [Bds](#).

Date

2023-07-26T15:47:56

The classes in here have been defined by a BOAP *.bidl file and define classes able to be communicated across a BOAP link

8.110 BdsD.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsD.h   Produced by Bidl
3  *****/
4  */
10
11 #ifndef BDSD_H
12 #define BDSD_H 1
13
14 #include <Boap.h>
15 #include <BObj.h>
16 #include <BDate.h>
17 #include <BTimeStamp.h>
18 #include <BComplex.h>
19 #include <BList.h>
20 #include <BArray.h>
21
22 //*****
23 //   BDS API
24 //       T.Barnaby,  BEAM Ltd,   2021-05-19
25 //*****
26 //
27 namespace Bds {
28     enum Errors { ErrorNoMetaData = 64, ErrorDataQuality = 65, ErrorSlaveMode = 66, ErrorTimeStamp =
29         67, ErrorValidate = 80, ErrorValidateMissingBlocks = 81, ErrorValidateTimeBackwards = 82,
30         ErrorValidateFilenameTime = 83, ErrorValidateMetaData = 84, ErrorValidateFix = 85,
31         ErrorValidateDuplicate = 86, ErrorValidateReorder = 87, ErrorValidateBdsFudge = 88};
32
33     enum Priority { PriorityLow, PriorityNormal, PriorityHigh};
34
35     enum Mode { ModeMaster, ModeSlave};
36
37     enum DataFlags { DataFlagNone = 0x00, DataFlagClipDataToTime = 0x01, DataFlagClipDataToChannels =
38         0x02, DataFlagMergeSegments = 0x04, DataFlagNoMetadata = 0x08};
39
40     enum SelectionGroup { SelectionGroupData, SelectionGroupMetaData, SelectionGroupDataWithCount};
41
42     enum SampleFormat { SampleFormatUnknown, SampleFormatInt16, SampleFormatInt32, SampleFormatFloat32,
43         SampleFormatFloat64, SampleFormatInt24};
44
45     enum AvailType { AvailNone, AvailPartial, AvailFull};
46
47     enum DataFormatSet { DataFormatSetNone = 0x00, DataFormatSetMetadataRead = 0x01,
48         DataFormatSetMetadataWrite = 0x02, DataFormatSetSensordataRead = 0x04, DataFormatSetSensordataWrite =
49         0x08};
50
51     enum LocationSelect { LocationSelectAll, LocationSelectStation, LocationSelectChannel};
52
53     class Point {
54     public:
55         Point(BFloat64 x = 0, BFloat64 y = 0);
56     public:
57         BFloat64 x;
58         BFloat64 y;
59     };
60
61     class TimePeriod : public BObj {
62     public:
63         TimePeriod(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp());
64         BString getType();
65         BError setMembers(BDictString& members);
66         BError setMember(BString name, BString value);
67         BError getMembers(BDictString& members);
68         BError getMember(BString name, BString& value);
69     public:
70         BTimeStamp startTime;
71         BTimeStamp endTime;
72     };
73
74     class ListRange : public BObj {
75     public:
76         ListRange(BUInt32 start = 0, BUInt32 number = 0, BInt32 reverse = 0);
77         BString getType();
78         BError setMembers(BDictString& members);
79         BError setMember(BString name, BString value);
80         BError getMembers(BDictString& members);
81         BError getMember(BString name, BString& value);
82     public:
83         BUInt32 start;
84         BUInt32 number;
85         BInt32 reverse;
86     };
87
88     };
89
90     };
91
92     };
93
94     };
95

```

```

98     class Network : public BObj {
99     public:
100         Network(BUInt32 id = 0, BString network = BString(), BString description = BString(),
101         BList<BString> stations = BList<BString>());
102         BString getType();
103         BError setMembers(BDictString& members);
104         BError setMember(BString name, BString value);
105         BError getMembers(BDictString& members);
106         BError getMember(BString name, BString& value);
107     public:
108         BUInt32 id;
109         BString network;
110         BString description;
111         BList<BString> stations;
112     };
113
114     class Source : public BObj {
115     public:
116         Source(BUInt32 id = 0, BString source = BString(), BString sourceMeta = BString(), BString alias
117         = BString(), BString description = BString());
118         BString getType();
119         BError setMembers(BDictString& members);
120         BError setMember(BString name, BString value);
121         BError getMembers(BDictString& members);
122         BError getMember(BString name, BString& value);
123     public:
124         BUInt32 id;
125         BString source;
126         BString sourceMeta;
127         BString alias;
128         BString description;
129     };
130
131     class SourcePriority : public BObj {
132     public:
133         SourcePriority(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
134         BTimeStamp(), BString source = BString(), BUInt32 priority = 0);
135         BString getType();
136         BError setMembers(BDictString& members);
137         BError setMember(BString name, BString value);
138         BError getMembers(BDictString& members);
139         BError getMember(BString name, BString& value);
140     public:
141         BUInt32 id;
142         BTimeStamp startTime;
143         BTimeStamp endTime;
144         BString source;
145         BUInt32 priority;
146     };
147
148     class ChannelName {
149     public:
150         ChannelName(BString network = BString(), BString station = BString(), BString channel =
151         BString(), BString source = BString());
152     public:
153         BString network;
154         BString station;
155         BString channel;
156         BString source;
157     };
158
159     class ArrayChannel {
160     public:
161         ArrayChannel(BString network = BString(), BString station = BString(), BString channel =
162         BString(), BFloat64 arrayOffsetEast = 0, BFloat64 arrayOffsetNorth = 0);
163     public:
164         BString network;
165         BString station;
166         BString channel;
167         BFloat64 arrayOffsetEast;
168         BFloat64 arrayOffsetNorth;
169     };
170
171     class Station {
172     public:
173         Station(BUInt32 id = 0, BString network = BString(), BString name = BString(), BString alias =
174         BString(), BString type = BString(), BString description = BString(), BList<ArrayChannel> channels =
175         BList<ArrayChannel>());
176     public:
177         BUInt32 id;
178         BString network;
179         BString name;
180         BString alias;
181         BString type;
182         BString description;
183         BList<ArrayChannel> channels;
184     };

```

```

199
205     class Location : public BObj {
206     public:
207         Location(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(),
BString network = BString(), BString station = BString(), BString channel = BString(), BString datum
= BString(), BFloat64 longitude = 0, BFloat64 latitude = 0, BFloat64 elevation = 0, BFloat64
arrayOffsetEast = 0, BFloat64 arrayOffsetNorth = 0);
208         BString getType();
209         BError setMembers(BDictString& members);
210         BError setMember(BString name, BString value);
211         BError getMembers(BDictString& members);
212         BError getMember(BString name, BString& value);
213     public:
214         BUInt32 id;
215         BTimeStamp startTime;
216         BTimeStamp endTime;
217         BString network;
218         BString station;
219         BString channel;
220         BString datum;
221         BFloat64 longitude;
222         BFloat64 latitude;
223         BFloat64 elevation;
224         BFloat64 arrayOffsetEast;
225         BFloat64 arrayOffsetNorth;
226     };
227
228     //
231     class PoleZero {
232     public:
233         PoleZero(BArray<BComplex > poles = BArray<BComplex >(), BArray<BComplex > zeros =
BArray<BComplex >());
234     public:
235         BArray<BComplex > poles;
236         BArray<BComplex > zeros;
237     };
238
239     //
241     class Fap {
242     public:
243         Fap(BFloat64 frequency = 0, BFloat64 amplitude = 0, BFloat64 phase = 0);
244     public:
245         BFloat64 frequency;
246         BFloat64 amplitude;
247         BFloat64 phase;
248     };
249
250     //
252     class FirEntry {
253     public:
254         FirEntry(BFloat64 coefficient = 0, BFloat64 error = 0);
255     public:
256         BFloat64 coefficient;
257         BFloat64 error;
258     };
259
260     //
263     class Fir {
264     public:
265         Fir(BArray<FirEntry > b = BArray<FirEntry >(), BArray<FirEntry > a = BArray<FirEntry >());
266     public:
267         BArray<FirEntry > b;
268         BArray<FirEntry > a;
269     };
270
271     //
273     class PolynomialEntry {
274     public:
275         PolynomialEntry(BFloat64 coefficient = 0, BFloat64 plusError = 0, BFloat64 minusError = 0,
BString measurementMethod = BString());
276     public:
277         BFloat64 coefficient;
278         BFloat64 plusError;
279         BFloat64 minusError;
280         BString measurementMethod;
281     };
282
283     //
286     class Polynomial {
287     public:
288         Polynomial(BString transferType = BString(), BString approximationType = BString(), BString
validFrequencyUnits = BString(), BFloat64 frequencyLowerBound = 0, BFloat64 frequencyUpperBound = 0,
BFloat64 approximationLowerBound = 0, BFloat64 approximationUpperBound = 0, BFloat64 maximumError =
0, BArray<PolynomialEntry > coefficients = BArray<PolynomialEntry >());
289     public:
290         BString transferType;
291         BString approximationType;

```

```

292     BString    validFrequencyUnits;
293     BFloat64   frequencyLowerBound;
294     BFloat64   frequencyUpperBound;
295     BFloat64   approximationLowerBound;
296     BFloat64   approximationUpperBound;
297     BFloat64   maximumError;
298     BArray<PolynomialEntry >    coefficients;
299 };
300
301 //
302 class Response {
303 public:
304     Response(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(),
305 BString network = BString(), BString station = BString(), BString channel = BString(), BString source
306 = BString(), BUInt32 stage = 0, BString name = BString(), BString type = BString(), PoleZero
307 poleZeros = PoleZero(), BArray<Fap > faps = BArray<Fap >(), Fir fir = Fir(), Polynomial polynomial =
308 Polynomial(), BFloat64 gain = 0, BFloat64 gainFrequency = 0, BString stageType = BString(), BFloat64
309 decimation = 0, BFloat64 decimationOffset = 0, BFloat64 decimationDelay = 0, BFloat64 decimationCorr
310 = 0, BString symmetry = BString(), BString description = BString(), BInt32 measured = 0, BFloat64
311 sampleRate = 0, BString inputUnits = BString(), BString inputUnitsDesc = BString(), BString
312 outputUnits = BString(), BString outputUnitsDesc = BString());
313
314 public:
315     BUInt32    id;
316     BTimeStamp  startTime;
317     BTimeStamp  endTime;
318     BString     network;
319     BString     station;
320     BString     channel;
321     BString     source;
322     BUInt32     stage;
323     BString     name;
324     BString     type;
325     PoleZero    poleZeros;
326     BArray<Fap >    faps;
327     Fir         fir;
328     Polynomial   polynomial;
329     BFloat64     gain;
330     BFloat64     gainFrequency;
331     BString      stageType;
332     BFloat64     decimation;
333     BFloat64     decimationOffset;
334     BFloat64     decimationDelay;
335     BFloat64     decimationCorr;
336     BString      symmetry;
337     BString      description;
338     BInt32       measured;
339     BFloat64     sampleRate;
340     BString      inputUnits;
341     BString      inputUnitsDesc;
342     BString      outputUnits;
343     BString      outputUnitsDesc;
344 };
345
346 class Calibration : public BObj {
347 public:
348     Calibration(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
349 BTimeStamp(), BString network = BString(), BString station = BString(), BString channel = BString(),
350 BString source = BString(), BString name = BString(), BFloat64 samplingFrequency = 0, BFloat64
351 calibrationFrequency = 0, BFloat64 calibrationFactor = 0, BString calibrationUnits = BString(),
352 BString calibrationUnitsDesc = BString(), BFloat64 rawCalibrationFrequency = 0, BFloat64
353 rawCalibrationFactor = 0, BString rawCalibrationUnits = BString(), BFloat64 depth = 0, BFloat64
354 waterLevel = 0, BFloat64 horizontalAngle = 0, BFloat64 verticalAngle = 0);
355
356     BString getType();
357     BError setMembers(BDictString& members);
358     BError setMember(BString name, BString value);
359     BError getMembers(BDictString& members);
360     BError getMember(BString name, BString& value);
361
362 public:
363     BUInt32    id;
364     BTimeStamp  startTime;
365     BTimeStamp  endTime;
366     BString     network;
367     BString     station;
368     BString     channel;
369     BString     source;
370     BString     name;
371     BFloat64     samplingFrequency;
372     BFloat64     calibrationFrequency;
373     BFloat64     calibrationFactor;
374     BString      calibrationUnits;
375     BString      calibrationUnitsDesc;
376     BFloat64     rawCalibrationFrequency;
377     BFloat64     rawCalibrationFactor;
378     BString      rawCalibrationUnits;
379     BFloat64     depth;
380     BFloat64     waterLevel;
381     BFloat64     horizontalAngle;

```

```

380         BFloat64    verticalAngle;
381     };
382
383     class Digitiser : public BObj {
384     public:
385         Digitiser(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
BTimeStamp(), BString name = BString(), BString type = BString(), BString serialNumber = BString(),
BUInt32 numberChannels = 0, BFloat64 baseSamplingFrequency = 0, BFloat64 initialSamplingFrequency =
0, BFloat64 gain = 0, BInt32 shared = 0);
388         BString getType();
389         BError setMembers(BDictString& members);
390         BError setMember(BString name, BString value);
391         BError getMembers(BDictString& members);
392         BError getMember(BString name, BString& value);
393     public:
394         BUInt32 id;
395         BTimeStamp startTime;
396         BTimeStamp endTime;
397         BString name;
398         BString type;
399         BString serialNumber;
400         BUInt32 numberChannels;
401         BFloat64 baseSamplingFrequency;
402         BFloat64 initialSamplingFrequency;
403         BFloat64 gain;
404         BInt32 shared;
405     };
406
407     class Sensor : public BObj {
408     public:
409         Sensor(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(),
BString name = BString(), BString type = BString(), BString serialNumber = BString(), BUInt32
numberChannels = 0, BString gainUnits = BString(), BFloat64 gain = 0, BUInt32 oldId = 0, BInt32
shared = 0);
412         BString getType();
413         BError setMembers(BDictString& members);
414         BError setMember(BString name, BString value);
415         BError getMembers(BDictString& members);
416         BError getMember(BString name, BString& value);
417     public:
418         BUInt32 id;
419         BTimeStamp startTime;
420         BTimeStamp endTime;
421         BString name;
422         BString type;
423         BString serialNumber;
424         BUInt32 numberChannels;
425         BString gainUnits;
426         BFloat64 gain;
427         BUInt32 oldId;
428         BInt32 shared;
429     };
430
431     class ChannelInstrument : public BObj {
432     public:
433         ChannelInstrument(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
BTimeStamp(), BUInt32 channelId = 0, BString source = BString(), BUInt32 digitiserId = 0, BUInt32
sensorId = 0);
438         BString getType();
439         BError setMembers(BDictString& members);
440         BError setMember(BString name, BString value);
441         BError getMembers(BDictString& members);
442         BError getMember(BString name, BString& value);
443     public:
444         BUInt32 id;
445         BTimeStamp startTime;
446         BTimeStamp endTime;
447         BUInt32 channelId;
448         BString source;
449         BUInt32 digitiserId;
450         BUInt32 sensorId;
451     };
452
453     class Channel : public BObj {
454     public:
455         Channel(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(),
BString network = BString(), BString station = BString(), BString channel = BString(), BString
channelType = BString(), BString channelAux = BString(), BString dataType = BString(), BString
description = BString());
461         BString getType();
462         BError setMembers(BDictString& members);
463         BError setMember(BString name, BString value);
464         BError getMembers(BDictString& members);
465         BError getMember(BString name, BString& value);
466     public:
467         BUInt32 id;
468         BTimeStamp startTime;

```

```

469         BTimeStamp    endTime;
470         BString    network;
471         BString    station;
472         BString    channel;
473         BString    channelType;
474         BString    channelAux;
475         BString    dataType;
476         BString    description;
477     };
478
481     class SelectionInfo {
482     public:
483         SelectionInfo(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(),
BList<BString > networks = BList<BString >(), BList<BString > arrays = BList<BString >(),
BList<BString > stations = BList<BString >(), BList<BString > arraysAndStations = BList<BString >(),
BList<BString > channels = BList<BString >(), BList<BString > sources = BList<BString >(), BUInt32
numDataChannels = 0);
484     public:
485         BTimeStamp    startTime;
486         BTimeStamp    endTime;
487         BList<BString > networks;
488         BList<BString > arrays;
489         BList<BString > stations;
490         BList<BString > arraysAndStations;
491         BList<BString > channels;
492         BList<BString > sources;
493         BUInt32    numDataChannels;
494     };
495
498     class SelectionChannel {
499     public:
500         SelectionChannel(BString network = BString(), BString station = BString(), BString channel =
BString(), BString source = BString());
501     public:
502         BString    network;
503         BString    station;
504         BString    channel;
505         BString    source;
506     };
507
515     class Selection {
516     public:
517         Selection(BUInt32 id = 0, ListRange range = ListRange(), BTimeStamp startTime = BTimeStamp(),
BTimeStamp endTime = BTimeStamp(), BList<SelectionChannel > channels = BList<SelectionChannel >(),
BUInt32 channelId = 0, BUInt32 digitiserId = 0, BUInt32 sensorId = 0, BUInt32 sensorOldId = 0, BInt32
completeSegments = 0, BString calibrationName = BString(), BString array = BString(), BUInt32 eventId
= 0, BString name = BString(), LocationSelect locationSelect = LocationSelect(), BString dataTypes =
BString(), BString excludeChannels = BString());
518     public:
519         BUInt32    id;
520         ListRange    range;
521         BTimeStamp    startTime;
522         BTimeStamp    endTime;
523         BList<SelectionChannel > channels;
524         BUInt32    channelId;
525         BUInt32    digitiserId;
526         BUInt32    sensorId;
527         BUInt32    sensorOldId;
528         BInt32    completeSegments;
529         BString    calibrationName;
530         BString    array;
531         BUInt32    eventId;
532         BString    name;
533         LocationSelect    locationSelect;
534         BString    dataTypes;
535         BString    excludeChannels;
536     };
537
542     class ChannelInfo {
543     public:
544         ChannelInfo(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(), Station
station = Station(), Location stationLocation = Location(), Channel channel = Channel(), Location
channelLocation = Location(), BString source = BString(), Digitiser digitiser = Digitiser(), Sensor
sensor = Sensor(), Calibration calibration = Calibration(), BList<Response > responses =
BList<Response >());
545     public:
546         BTimeStamp    startTime;
547         BTimeStamp    endTime;
548         Station    station;
549         Location    stationLocation;
550         Channel    channel;
551         Location    channelLocation;
552         BString    source;
553         Digitiser    digitiser;
554         Sensor    sensor;
555         Calibration    calibration;
556         BList<Response > responses;

```

```

557     };
558
559     class ChannelInfos {
560     public:
561         ChannelInfos(Station array = Station(), BArray<BArray<ChannelInfo > > channels =
BArray<BArray<ChannelInfo > >());
562     public:
563         Station array;
564         BArray<BArray<ChannelInfo > > channels;
565     };
566
567     class DataFileInfo : public BObj {
568     public:
569         DataFileInfo(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
BTimeStamp(), BString location = BString(), BString format = BString(), BString url = BString(),
BString stream = BString(), BString comment = BString(), BUInt32 importUserId = 0, BTimeStamp
importTime = BTimeStamp(), BString state = BString());
570     public:
571         BString getType();
572         BError setMembers(BDictString& members);
573         BError setMember(BString name, BString value);
574         BError getMembers(BDictString& members);
575         BError getMember(BString name, BString& value);
576     public:
577         BUInt32 id;
578         BTimeStamp startTime;
579         BTimeStamp endTime;
580         BString location;
581         BString format;
582         BString url;
583         BString stream;
584         BString comment;
585         BUInt32 importUserId;
586         BTimeStamp importTime;
587         BString state;
588     };
589
590     class DataChannel : public BObj {
591     public:
592         DataChannel(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
BTimeStamp(), BString network = BString(), BString station = BString(), BString channel = BString(),
BString source = BString(), BUInt32 numBlocks = 0, BUInt64 numSamples = 0, BFloat64 sampleRate = 0,
BUInt32 sampleFormat = 0, BUInt32 dataFileId = 0, BUInt32 dataFileChannel = 0, BString importFormat =
BString(), BString importFilename = BString(), BTimeStamp importStartTime = BTimeStamp(),
BDict<BString > info = BDict<BString >());
593     public:
594         BString getType();
595         BError setMembers(BDictString& members);
596         BError setMember(BString name, BString value);
597         BError getMembers(BDictString& members);
598         BError getMember(BString name, BString& value);
599     public:
600         BUInt32 id;
601         BTimeStamp startTime;
602         BTimeStamp endTime;
603         BString network;
604         BString station;
605         BString channel;
606         BString source;
607         BUInt32 numBlocks;
608         BUInt64 numSamples;
609         BFloat64 sampleRate;
610         BUInt32 sampleFormat;
611         BUInt32 dataFileId;
612         BUInt32 dataFileChannel;
613         BString importFormat;
614         BString importFilename;
615         BTimeStamp importStartTime;
616         BDict<BString > info;
617     };
618
619     class DataInfo {
620     public:
621         DataInfo(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(), BString array =
BString(), BString description = BString(), BUInt32 synchronous = 0, BArray<BArray<DataChannel > >
channels = BArray<BArray<DataChannel > >(), BDict<BString > info = BDict<BString >(), BDict<BString >
infoExtra = BDict<BString >(), BList<BString > warnings = BList<BString >());
622     public:
623         BTimeStamp startTime;
624         BTimeStamp endTime;
625         BString array;
626         BString description;
627         BUInt32 synchronous;
628         BArray<BArray<DataChannel > > channels;
629         BDict<BString > info;
630         BDict<BString > infoExtra;
631         BList<BString > warnings;
632     };
633
634     };
635
636     };
637
638     };
639
640     };
641
642     };
643
644     };
645
646     };
647
648     };
649
650     };
651
652     };
653
654     };

```

```

656     class DataAvail {
657     public:
658         DataAvail(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(), AvailType
availType = AvailType());
659     public:
660         BTimeStamp    startTime;
661         BTimeStamp    endTime;
662         AvailType    availType;
663     };
664
665     class DataAvailChan {
666     public:
667         DataAvailChan(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(), BString
network = BString(), BString station = BString(), BString channel = BString(), BString source =
BString(), BArray<DataAvail > segments = BArray<DataAvail >());
668     public:
669         BTimeStamp    startTime;
670         BTimeStamp    endTime;
671         BString    network;
672         BString    station;
673         BString    channel;
674         BString    source;
675         BArray<DataAvail >    segments;
676     };
677
678     class DataHandle {
679     public:
680         DataHandle(BUInt32 handle = 0, BUInt32 dataFileId = 0);
681     public:
682         BUInt32    handle;
683         BUInt32    dataFileId;
684     };
685
686     class DataBlock {
687     public:
688         DataBlock(BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(), BUInt32
channelNumber = 0, BUInt32 segmentNumber = 0, BArray<BArray<BFloat64 > > channelData =
BArray<BArray<BFloat64 > >(), BDict<BString > info = BDict<BString >());
689     public:
690         BTimeStamp    startTime;
691         BTimeStamp    endTime;
692         BUInt32    channelNumber;
693         BUInt32    segmentNumber;
694         BArray<BArray<BFloat64 > >    channelData;
695         BDict<BString >    info;
696     };
697
698     class DataBlockChannel {
699     public:
700         DataBlockChannel(BString network = BString(), BString station = BString(), BString channel =
BString(), BString source = BString());
701     public:
702         BString    network;
703         BString    station;
704         BString    channel;
705         BString    source;
706     };
707
708     class User : public BObj {
709     public:
710         User(BUInt32 id = 0, BString user = BString(), BString password = BString(), BString name =
BString(), BString email = BString(), BString telephone = BString(), BString address = BString(),
BInt32 enabled = 0, BList<BString > groups = BList<BString >());
711         BString    getType();
712         BError    setMembers(BDictString& members);
713         BError    setMember(BString name, BString value);
714         BError    getMembers(BDictString& members);
715         BError    getMember(BString name, BString& value);
716     public:
717         BUInt32    id;
718         BString    user;
719         BString    password;
720         BString    name;
721         BString    email;
722         BString    telephone;
723         BString    address;
724         BInt32    enabled;
725         BList<BString >    groups;
726     };
727
728     class Group : public BObj {
729     public:
730         Group(BUInt32 id = 0, BString group = BString(), BString description = BString());
731         BString    getType();
732         BError    setMembers(BDictString& members);
733         BError    setMember(BString name, BString value);
734         BError    getMembers(BDictString& members);

```



```

762     BError getMember(BString name, BString& value);
763 public:
764     UInt32 id;
765     BString group;
766     BString description;
767 };
768
769 class AccessGroup : public BObj {
770 public:
771     AccessGroup(UInt32 id = 0, BString group = BString(), BTimeStamp startTime = BTimeStamp(),
772 BTimeStamp endTime = BTimeStamp(), BString network = BString(), BString station = BString());
773     BString getType();
774     BError setMembers(BDictString& members);
775     BError setMember(BString name, BString value);
776     BError getMembers(BDictString& members);
777     BError getMember(BString name, BString& value);
778 public:
779     UInt32 id;
780     BString group;
781     BTimeStamp startTime;
782     BTimeStamp endTime;
783     BString network;
784     BString station;
785 };
786
787 class Change : public BObj {
788 public:
789     Change(UInt32 id = 0, UInt32 changeGroupId = 0, BTimeStamp time = BTimeStamp(), BString type =
790 BString(), BString table = BString(), UInt32 rowId = 0);
791     BString getType();
792     BError setMembers(BDictString& members);
793     BError setMember(BString name, BString value);
794     BError getMembers(BDictString& members);
795     BError getMember(BString name, BString& value);
796 public:
797     UInt32 id;
798     UInt32 changeGroupId;
799     BTimeStamp time;
800     BString type;
801     BString table;
802     UInt32 rowId;
803 };
804
805 class ChangeGroup : public BObj {
806 public:
807     ChangeGroup(UInt32 id = 0, BTimeStamp time = BTimeStamp(), BString type = BString(), BString
808 user = BString(), BString title = BString(), BString description = BString());
809     BString getType();
810     BError setMembers(BDictString& members);
811     BError setMember(BString name, BString value);
812     BError getMembers(BDictString& members);
813     BError getMember(BString name, BString& value);
814 public:
815     UInt32 id;
816     BTimeStamp time;
817     BString type;
818     BString user;
819     BString title;
820     BString description;
821 };
822
823 class Note : public BObj {
824 public:
825     Note(UInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime = BTimeStamp(),
826 BString network = BString(), BString station = BString(), BString channel = BString(), BString source
827 = BString(), BString type = BString(), BString user = BString(), BTimeStamp timeAdded = BTimeStamp(),
828 UInt32 errorNumber = 0, BString title = BString(), BString description = BString(), BString docFormat
829 = BString(), BString docUrl = BString(), UInt32 dataFileId = 0, BString importFilename = BString(),
830 UInt32 eventId = 0);
831     BString getType();
832     BError setMembers(BDictString& members);
833     BError setMember(BString name, BString value);
834     BError getMembers(BDictString& members);
835     BError getMember(BString name, BString& value);
836 public:
837     UInt32 id;
838     BTimeStamp startTime;
839     BTimeStamp endTime;
840     BString network;
841     BString station;
842     BString channel;
843     BString source;
844     BString type;
845     BString user;
846     BTimeStamp timeAdded;
847     UInt32 errorNumber;
848     BString title;

```

```

856     BString    description;
857     BString    docFormat;
858     BString    docUrl;
859     BUInt32    dataFileId;
860     BString    importFilename;
861     BUInt32    eventId;
862 };
863
864 class Log : public BObj {
865 public:
866     Log(BUInt32 id = 0, BTimeStamp time = BTimeStamp(), BString type = BString(), BUInt32 priority =
0, BString subSystem = BString(), BString title = BString(), BString description = BString());
870     BString    getType();
871     BError    setMembers(BDictString& members);
872     BError    setMember(BString name, BString value);
873     BError    getMembers(BDictString& members);
874     BError    getMember(BString name, BString& value);
875 public:
876     BUInt32    id;
877     BTimeStamp    time;
878     BString    type;
879     BUInt32    priority;
880     BString    subSystem;
881     BString    title;
882     BString    description;
883 };
884
885 class LogSelect {
886 public:
887     LogSelect(BTimeStamp startTime = BTimeStamp(), BString type = BString(), BUInt32 priority = 0,
BString subSystem = BString());
889 public:
890     BTimeStamp    startTime;
891     BString    type;
892     BUInt32    priority;
893     BString    subSystem;
894 };
895
896 class CleanOptions {
897 public:
898     CleanOptions(BInt32 logs = 0, BInt32 changes = 0, BInt32 deletedFiles = 0);
899 public:
900     BInt32    logs;
901     BInt32    changes;
902     BInt32    deletedFiles;
903 };
904
905 class DataFormat {
906 public:
907     DataFormat(BList<BString> names = BList<BString>(), BInt32 dataRead = 0, BInt32 dataWrite = 0,
BInt32 metadataRead = 0, BInt32 metadataWrite = 0, BString extension = BString(), BString description
= BString());
911 public:
912     BList<BString>    names;
913     BInt32    dataRead;
914     BInt32    dataWrite;
915     BInt32    metadataRead;
916     BInt32    metadataWrite;
917     BString    extension;
918     BString    description;
919 };
920
921 typedef BList<DataFormat> DataFormats;
922
923 class SpecialChannel : public BObj {
924 public:
925     SpecialChannel(BUInt32 id = 0, BTimeStamp startTime = BTimeStamp(), BTimeStamp endTime =
BTimeStamp(), BString network = BString(), BString station = BString(), BString channel = BString(),
BString dataType = BString(), BString description = BString());
930     BString    getType();
931     BError    setMembers(BDictString& members);
932     BError    setMember(BString name, BString value);
933     BError    getMembers(BDictString& members);
934     BError    getMember(BString name, BString& value);
935 public:
936     BUInt32    id;
937     BTimeStamp    startTime;
938     BTimeStamp    endTime;
939     BString    network;
940     BString    station;
941     BString    channel;
942     BString    dataType;
943     BString    description;
944 };
945
946 class Event {
947 public:

```

```

951         Event(BUInt32 id = 0, BUInt32 userId = 0, BString type = BString(), BString title = BString(),
BString network = BString(), BString source = BString(), BTimeStamp startTime = BTimeStamp(),
BTimeStamp endTime = BTimeStamp(), BTimeStamp eventTime = BTimeStamp(), BFloat64 longitude = 0,
BFloat64 latitude = 0, BFloat64 elevation = 0, BFloat64 waterDepth = 0, BFloat64 magnitude = 0,
BString magnitudeUnits = BString(), BString description = BString(), BString notes = BString(),
BDict<BString > extra = BDict<BString >(), BList<SelectionChannel > dataChannels =
BList<SelectionChannel >());
952     public:
953         BUInt32 id;
954         BUInt32 userId;
955         BString type;
956         BString title;
957         BString network;
958         BString source;
959         BTimeStamp startTime;
960         BTimeStamp endTime;
961         BTimeStamp eventTime;
962         BFloat64 longitude;
963         BFloat64 latitude;
964         BFloat64 elevation;
965         BFloat64 waterDepth;
966         BFloat64 magnitude;
967         BString magnitudeUnits;
968         BString description;
969         BString notes;
970         BDict<BString > extra;
971         BList<SelectionChannel > dataChannels;
972     };
973 }
987 }
988
989 #endif

```

8.111 BdsLib.cpp File Reference

```

#include <BdsLib.h>
#include <math.h>
#include <complex>

```

Namespaces

- namespace [Bds](#)

Functions

- **Error** [Bds::bdsLibInit](#) (DataAccess &bds)
Initialise the bdsLib with settings from the BdsServer.
- **Error** [Bds::bdsLibInit](#) (DataAddAccess &bds)
Initialise the bdsLib with settings from the BdsServer.
- **Error** [Bds::bdsLibInit](#) (AdminAccess &bds)
Initialise the bdsLib with settings from the BdsServer.
- void [Bds::bdsDumpPoleZeros](#) (PoleZero poleZeros)
Debug print out a PoleZeros object.
- void [Bds::bdsChannelGetTypeAux](#) (**BString** name, **BString** &type, **BString** &aux)
Get the channel type and aux fields from a generic channel name.
- **BString** [Bds::bdsChannelGetName](#) (**BString** type, **BString** aux)
Create a full channel name from a channels type and aux fields.
- **Error** [Bds::bdsDataInfoSetTimeRange](#) (DataInfo &dataInfo)
Restricts the time tange of all of the [DataInfo](#)'s channels to match the [DataInfo](#)'s startTime/endTime fields.
- **Error** [Bds::bdsDataInfoFromInfo](#) (**BDictString** info, DataInfo &dataInfo, **Bool** append)

- Convert info to [DataInfo](#).
- **Error** [Bds::bdsInfoFromDataInfo](#) (const [DataInfo](#) &dataInfo, **BDictString** &info)

Converts a [DataInfo](#) object into a [BDictString](#) list of named strings.
- **Error** [Bds::bdsDataInfoFlatten](#) ([DataInfo](#) &dataInfo)

Flattens a [DataInfo](#) to 1 segment per channel for use in `dataOpen()` calls.
- **Error** [Bds::bdsDataInfoMergeFlatten](#) ([DataInfo](#) &dataInfo, const [DataInfo](#) &dataInfoAdd)

Merges a [DataInfo](#) into another flattening the segments to 1 for use in `dataOpen()` calls.
- **String** [Bds::bdsUnitsConvert](#) (**String** units)

Tidy up units name to standard SI units format.
- **Error** [Bds::bdsDataInfoFromChannelInfos](#) (const [ChannelInfos](#) &channelInfos, [DataInfo](#) &dataInfo)

Sets up a [DataInfo](#) object from [ChannelInfos](#).
- static int [Bds::responseSort](#) ([Response](#) &a, [Response](#) &b)
- **Error** [Bds::bdsMetadataImportFix](#) ([ChannelInfos](#) &channelInfos, **Bool** stageRenummer)

Fix up [ChannelInfos](#) from import. Mainly making sure response stages and their units are correct.
- **Error** [Bds::bdsMetadataExportFix](#) ([ChannelInfos](#) &channelInfos, **Bool** singleResponse, **Bool** stage↔Renummer, **Bool** changeUnits, **Bool** stageGains, **Bool** decimation, **Bool** toDisplacement, **Bool** toNm)

Fix up [ChannelInfos](#) for export. Mainly making sure response stages and their units are correct.
- **String** [Bds::bdsStationAlias](#) ([Station](#) station)

Returns the station alias if set else its name.
- void [Bds::bdsDumpSelection](#) ([Selection](#) sel)

Debug print out a [Selection](#) object.
- void [Bds::bdsDumpSelectionInfo](#) ([SelectionInfo](#) sel)

Debug print out a [SelectionInfo](#) object.
- void [Bds::bdsDumpDataInfo](#) ([DataInfo](#) dataInfo, int includeInfo=0)

Debug print out a [DataInfo](#) object.
- void [Bds::bdsDumpChannelInfos](#) (const [ChannelInfos](#) &channelInfos)

Debug print out a [ChannelInfos](#) object.
- void [Bds::bdsDumpDataBlock](#) (const [DataBlock](#) &dataBlock, int nSamples=0)

Debug print out a [DataBlock](#) object.
- void [Bds::bdsDumpLocation](#) ([Location](#) location)

Debug printout location.
- **String** [Bds::bdsDataChannelInfo](#) (const [DataChannel](#) &dataChannel)

Returns a string representation of a [DataChannel](#) object.
- **String** [Bds::bdsDataChannelRef](#) (const [DataChannel](#) &dataChannel)

Returns the string reference name of a [DataChannel](#) object.
- **String** [Bds::bdsDataChannelRef](#) (const [ChannelInfo](#) &channelInfo)

Returns the string reference name of a [ChannelInfo](#) object.
- **Error** [Bds::bdsDataChannelOverallResponse](#) (const [ChannelInfo](#) &channelInfo, [Response](#) &response)

Returns the overal response from the list of responses in a [ChannelInfo](#).
- **String** [Bds::bdsSelectionChannelInfo](#) (const [Selection](#) &selection, **UInt** channel)

Returns a string describing the name and time period of a selection channel.
- double [Bds::bdsPoleZeroGain](#) (const [PoleZero](#) &poleZero, double frequency)

Calculates the overall gain of the given [PoleZero](#) transfer function.
- void [Bds::bdsPoleZeroGainPhase](#) (const [PoleZero](#) &poleZero, double frequency, double &gain, double &phase)

Calculates the overall gain and phase of the given [PoleZero](#) transfer function.
- void [Bds::bdsPoleZeroToFap](#) (const [PoleZero](#) &poleZero, **UInt** nPoints, double calibrationFrequency, double sampleFrequency, **Array**< [Fap](#) > &fap)

Convert [PoleZero](#) to FAP.
- static **String** [Bds::fileNameTime](#) (**TimeStamp** t)
- **String** [Bds::bdsFileNameExpand](#) (**String** fileName, [ChannelInfo](#) &channelInfo)

- Default filename from a [ChannelInfo](#).*

 - **BString** [Bds::bdsFileNameExpand](#) (**BString** fileName, **ChannelInfos** &channelInfos)
- Default filename from a list of [ChannelInfo](#)'s.*

 - **BString** [Bds::bdsFileNameExpand](#) (**BString** fileName, **Selection** &sel)
- Default filename from a [Selection](#).*

 - void [Bds::bdsSpecialChannelsSet](#) (const **BList**< **SpecialChannel** > specialChannels)
- Set the special channels list.*

 - **BList**< **SpecialChannel** > [Bds::bdsSpecialChannels](#) ()
- Return list of special channels.*

 - **Bool** [Bds::bdsSpecialChannelIgnore](#) (**BString** network, **BString** station, **BString** channel)
- Check if channel should be ignored.*

 - char [Bds::seedChannelInstrumentCode](#) (**BString** dataType)
- Returns SEED instrument code from dataType.*

 - **BString** [Bds::seedChannelDataType](#) (**BString** channel)
- Returns dataType from channel name based on SEED channel name convention.*

 - **BStringList** [Bds::bdsDataTypes](#) ()
- Returns all known data types.*

 - **BStringList** [Bds::bdsUnits](#) ()
- Returns all known SI units.*

 - **BString** [Bds::bdsUnitCase](#) (**BString** unit)
- Converts character case of units.*

Variables

- static **BList**< **SpecialChannel** > [Bds::bdsSpecialChannelsList](#)
- SeedICodeToDataType [Bds::seedICodeToDataTypes](#) []

8.112 BdsLib.d File Reference

8.113 BdsLib.dox File Reference

Namespaces

- namespace [Bds](#)

Functions

- void [Bds::bdsChannelGetTypeAux](#) (**BString** name, **BString** &type, **BString** &aux)
- Get the channel type and aux fields from a generic channel name.*

8.114 BdsLib.h File Reference

General BdsLib API functions.

```
#include <BdsD.h>
#include <BdsC.h>
#include <BString.h>
```

Classes

- class [Bds::ResponseObj](#)
Response object adding string conversion.
- class [Bds::DataError](#)
This stores a data error. It includes an error number and a string as well as information on what seismic channel it is for.

Namespaces

- namespace [Bds](#)

Functions

- **Error** [Bds::bdsLibInit](#) (DataAccess &bds)
Initialise the bdsLib with settings from the BdsServer.
- **Error** [Bds::bdsLibInit](#) (DataAddAccess &bds)
Initialise the bdsLib with settings from the BdsServer.
- **Error** [Bds::bdsLibInit](#) (AdminAccess &bds)
Initialise the bdsLib with settings from the BdsServer.
- void [Bds::bdsChannelGetTypeAux](#) (**BString** name, **BString** &type, **BString** &aux)
Get the channel type and aux fields from a generic channel name.
- **BString** [Bds::bdsChannelGetName](#) (**BString** type, **BString** aux)
Create a full channel name from a channels type and aux fields.
- **Error** [Bds::bdsDataInfoSetTimeRange](#) (DataInfo &dataInfo)
Restricts the time range of all of the [DataInfo](#)'s channels to match the [DataInfo](#)'s startTime/endTime fields.
- **Error** [Bds::bdsDataInfoFromInfo](#) (**BDictString** info, DataInfo &dataInfo, **Bool** append)
Convert info to [DataInfo](#).
- **Error** [Bds::bdsInfoFromDataInfo](#) (const DataInfo &dataInfo, **BDictString** &info)
Converts a [DataInfo](#) object into a [BDictString](#) list of named strings.
- **Error** [Bds::bdsDataInfoFlatten](#) (DataInfo &dataInfo)
Flattens a [DataInfo](#) to 1 segment per channel for use in [dataOpen\(\)](#) calls.
- **Error** [Bds::bdsDataInfoMergeFlatten](#) (DataInfo &dataInfo, const DataInfo &dataInfoAdd)
Merges a [DataInfo](#) into another flattening the segments to 1 for use in [dataOpen\(\)](#) calls.
- **BString** [Bds::bdsUnitsConvert](#) (**BString** units)
Tidy up units name to standard SI units format.
- **Error** [Bds::bdsDataInfoFromChannelInfos](#) (const ChannelInfos &channelInfos, DataInfo &dataInfo)
Sets up a [DataInfo](#) object from [ChannelInfos](#).
- **Error** [Bds::bdsMetadataImportFix](#) (ChannelInfos &channelInfos, **Bool** stageRenum)
Fix up [ChannelInfos](#) from import. Mainly making sure response stages and their units are correct.
- **Error** [Bds::bdsMetadataExportFix](#) (ChannelInfos &channelInfos, **Bool** singleResponse, **Bool** stage↔Renum, **Bool** changeUnits, **Bool** stageGains, **Bool** decimation, **Bool** toDisplacement, **Bool** toNm)
Fix up [ChannelInfos](#) for export. Mainly making sure response stages and their units are correct.
- **BString** [Bds::bdsStationAlias](#) (Station station)
Returns the station alias if set else its name.
- **BString** [Bds::bdsDataChannelInfo](#) (const DataChannel &dataChannel)
Returns a string representation of a [DataChannel](#) object.
- **BString** [Bds::bdsDataChannelRef](#) (const DataChannel &dataChannel)
Returns the string reference name of a [DataChannel](#) object.
- **BString** [Bds::bdsDataChannelRef](#) (const ChannelInfo &channelInfo)

- Returns the string reference name of a [ChannelInfo](#) object.*

 - **BError** [Bds::bdsDataChannelOverallResponse](#) (const [ChannelInfo](#) &channelInfo, Response &response)

Returns the overal response from the list of responses in a [ChannelInfo](#).
- **BString** [Bds::bdsSelectionChannelInfo](#) (const Selection &selection, **BUInt** channel)

Returns a string describing the name and time period of a selection channel.
- **BString** [Bds::bdsFileNameExpand](#) (**BString** fileName, [ChannelInfo](#) &channelInfo)

Default filename from a [ChannelInfo](#).
- **BString** [Bds::bdsFileNameExpand](#) (**BString** fileName, [ChannelInfos](#) &channelInfos)

Default filename from a list of [ChannelInfo](#)'s.
- **BString** [Bds::bdsFileNameExpand](#) (**BString** fileName, Selection &sel)

Default filename from a [Selection](#).
- void [Bds::bdsSpecialChannelsSet](#) (const **BList**< [SpecialChannel](#) > specialChannels)

Set the special channels list.
- **BList**< [SpecialChannel](#) > [Bds::bdsSpecialChannels](#) ()

Return list of special channels.
- **Bool** [Bds::bdsSpecialChannelIgnore](#) (**BString** network, **BString** station, **BString** channel)

Check if channel should be ignored.
- char [Bds::seedChannelInstrumentCode](#) (**BString** dataType)

Returns SEED instrument code from dataType.
- **BString** [Bds::seedChannelDataType](#) (**BString** channel)

Returns dataType from channel name based on SEED channel name convention.
- **BStringList** [Bds::bdsDataTypes](#) ()

Returns all known data types.
- **BStringList** [Bds::bdsUnits](#) ()

Returns all known SI units.
- **BString** [Bds::bdsUnitCase](#) (**BString** unit)

Converts character case of units.
- double [Bds::bdsPoleZeroGain](#) (const [PoleZero](#) &poleZero, double frequency)

Calculates the overall gain of the given [PoleZero](#) transfer function.
- void [Bds::bdsPoleZeroGainPhase](#) (const [PoleZero](#) &poleZero, double frequency, double &gain, double &phase)

Calculates the overall gain and phase of the given [PoleZero](#) transfer function.
- void [Bds::bdsPoleZeroToFap](#) (const [PoleZero](#) &poleZero, **BUInt** nPoints, double calibrationFrequency, double sampleFrequency, **BArray**< [Fap](#) > &fap)

Convert [PoleZero](#) to FAP.
- void [Bds::bdsDumpSelection](#) (Selection sel)

Debug print out a [Selection](#) object.
- void [Bds::bdsDumpSelectionInfo](#) (SelectionInfo sel)

Debug print out a [SelectionInfo](#) object.
- void [Bds::bdsDumpDataInfo](#) (DataInfo dataInfo, int includeInfo=0)

Debug print out a [DataInfo](#) object.
- void [Bds::bdsDumpChannelInfos](#) (const [ChannelInfos](#) &channelInfos)

Debug print out a [ChannelInfos](#) object.
- void [Bds::bdsDumpData](#) (const [DataBlock](#) &dataBlock, int nSamples=0)

Debug print out a [DataBlock](#) object.
- void [Bds::bdsDumpPoleZeros](#) ([PoleZero](#) poleZeros)

Debug print out a [PoleZeros](#) object.
- void [Bds::bdsDumpLocation](#) ([Location](#) location)

Debug printout location.

Variables

- const int `Bds::NetworkNameLen` = 3
Maximum [Network](#) name length.
- const int `Bds::StationNameLen` = 5
Maximum [Station](#) name length.
- const int `Bds::ChannelTypeLen` = 3
Maximum [Channel](#) type name length.
- const int `Bds::ChannelAuxLen` = 2
Maximum [Channel](#) Aux length.
- const int `Bds::SourceLen` = 16
Maximum [Source](#) length.

8.114.1 Detailed Description

General BdsLib API functions.

8.115 BdsLib.h

[Go to the documentation of this file.](#)

```

1  /*****
2  *   BdsLib.h       Data File Access
3  *   T.Barnaby, BEAM Ltd, 2008-12-03
4  *****/
5  */
9  #ifndef BdsLib_H
10 #define BdsLib_H
11
12 #include <BdsD.h>
13 #include <BdsC.h>
14 #include <BString.h>
15
16 namespace Bds {
17
18     const int    NetworkNameLen = 3;
19     const int    StationNameLen = 5;
20     const int    ChannelTypeLen = 3;
21     const int    ChannelAuxLen = 2;
22     const int    SourceLen = 16;
23
24     class ResponseObj : public Response {
25     public:
26         ResponseObj(const Response& response);
27         ~ResponseObj();
28
29         BString    getString();
30         void       setString(BString str);
31     };
32
33     class DataError {
34     public:
35         DataError();
36         DataError(int errorNumber, BString title, BString filename, BTimeStamp startTime, BTimeStamp
37             endTime, DataInfo& dataInfo, BUInt channel, BString description, BString user = "");
38         DataError& set(int errorNumber, BString title, BString importFilename, BTimeStamp startTime,
39             BTimeStamp endTime, DataInfo& dataInfo, BUInt channel, BString description, BString user = "");
40         void       mergeDataInfo(const DataInfo& dataInfo, BUInt channel);
41
42         int        getErrorNumber() const;
43         BString     getTitle() const;
44
45         BError      setString(BString str);
46         BError      setStringUser(BString str, BString user);
47         BString     getString() const;
48
49         int         num() const;
50         const char* str() const;
51
52         operator int() const;

```



```

53
54 public:
55     BInt32          oerrorNumber;
56     BString         otitle;
57     BString         odescription;
58     BString         ofilename;
59     BTimeStamp      ostartTime;
60     BTimeStamp      oendTime;
61     BString         onetwork;
62     BString         ostation;
63     BString         ochannel;
64     BString         osource;
65     BString         ouser;
66 };
67
68
69 BError bdsLibInit(DataAccess& bds);
70 BError bdsLibInit(DataAddAccess& bds);
71 BError bdsLibInit(AdminAccess& bds);
72
73 // Bds Channel name manipulations
74 void bdsChannelGetTypeAux(BString name, BString& type, BString& aux);
75 BString bdsChannelGetName(BString type, BString aux);
76
77 // Bds Data Meta data manipulations
78 BError bdsDataInfoSetTimeRange(DataInfo& dataInfo);
79 BError bdsDataInfoFromInfo(BDictString info, DataInfo& dataInfo, Bool append = 0);
80 BError bdsInfoFromDataInfo(const DataInfo& dataInfo, BDictString& info);
81 BError bdsDataInfoFlatten(DataInfo& dataInfo);
82 BError bdsDataInfoMergeFlatten(DataInfo& dataInfo, const DataInfo& dataInfoAdd);
83 BString bdsUnitsConvert(BString units);
84 BError bdsDataInfoFromChannelInfos(const ChannelInfos& channelInfos, DataInfo& dataInfo);
85
86 BError bdsMetadataImportFix(ChannelInfos& channelInfos, Bool stageRenumber);
87 BError bdsMetadataExportFix(ChannelInfos& channelInfos, Bool singleResponse, Bool stageRenumber, Bool
    changeUnits, Bool stageGains, Bool decimation, Bool toDisplacement, Bool toNm);
88
89 BString bdsStationAlias(Station station);
90 BString bdsDataChannelInfo(const DataChannel& dataChannel);
91 BString bdsDataChannelRef(const DataChannel& dataChannel);
92 BString bdsDataChannelRef(const ChannelInfo& channelInfo);
93 BError bdsDataChannelOverallResponse(const ChannelInfo& channelInfo, Response& response);
94 BString bdsSelectionChannelInfo(const Selection& selection, BUInt channel);
95 BString bdsFileNameExpand(BString fileName, ChannelInfo& channelInfo);
96 BString bdsFileNameExpand(BString fileName, ChannelInfos& channelInfos);
97 BString bdsFileNameExpand(BString fileName, Selection& sel);
98
99 // BDS Metadata higher level functionality
100 void bdsSpecialChannelsSet(const BList<SpecialChannel> specialChannels);
101 BList<SpecialChannel> bdsSpecialChannels();
102 Bool bdsSpecialChannelIgnore(BString network, BString station, BString channel);
103 char seedChannelInstrumentCode(BString dataType);
104 BString seedChannelDataType(BString channel);
105 BStringList bdsDataTypes();
106 BStringList bdsUnits();
107 BString bdsUnitCase(BString unit);
108
109 // Calculations
110 double bdsPoleZeroGain(const PoleZero& poleZero, double frequency);
111 void bdsPoleZeroGainPhase(const PoleZero& poleZero, double frequency, double& gain, double& phase);
112 void bdsPoleZeroToFap(const PoleZero& poleZero, BUInt nPoints, double calibrationFrequency, double
    sampleFrequency, BArray<Fap>& fap);
113
114 // Bds Debug functions
115 void bdsDumpSelection(Selection sel);
116 void bdsDumpSelectionInfo(SelectionInfo sel);
117 void bdsDumpDataInfo(DataInfo dataInfo, int includeInfo = 0);
118 void bdsDumpChannelInfos(const ChannelInfos& channelInfos);
119 void bdsDumpData(const DataBlock& dataBlock, int nSamples = 0);
120 void bdsDumpPoleZeros(PoleZero poleZeros);
121 void bdsDumpLocation(Location location);
122
123 }
124 #endif

```

8.116 BdsS.cc File Reference

```

#include <BdsC.h>
#include <BdsS.h>

```

Namespaces

- namespace [Bds](#)

8.117 BdsS.d File Reference

8.118 BdsT.cc File Reference

```
#include <stdlib.h>
#include <stdint.h>
#include <BdsT.h>
#include <Control.h>
```

8.119 /src/blacknest/bds/bds/doc/bdsApi-extra.dox File Reference

8.120 /src/blacknest/bds/bds/doc/bdsApiOverview.dox File Reference

8.121 /src/blacknest/bds/bds/doc/BdsPython.dox File Reference

Index

/src/blacknest/bds/bds/bdsDataLib/BdsCompress.cpp, 401
/src/blacknest/bds/bds/bdsDataLib/BdsCompress.d, 401
/src/blacknest/bds/bds/bdsDataLib/BdsCompress.h, 401, 402
/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.cpp, 402
/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.d, 403
/src/blacknest/bds/bds/bdsDataLib/BdsDataCollate.h, 403
/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.cpp, 403
/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.d, 404
/src/blacknest/bds/bds/bdsDataLib/BdsDataFile.h, 404
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.cpp, 405
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.d, 406
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAd22.h, 406, 407
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.cpp, 407
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.d, 408
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileAscii.h, 408
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.cpp, 409
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.d, 409
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBdrs.h, 409
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.cpp, 410
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.d, 412
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBds.h, 412, 413
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.cpp, 415
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.d, 416
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileBknas.h, 416
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.cpp, 417
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.d, 419
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCd.h, 419, 420
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.cpp, 421
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.d, 421
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileCss.h, 421, 422
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.cpp, 423
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.d, 423
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileGcf.h, 423, 424
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.cpp, 424
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.d, 425
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileIdc.h, 425, 426
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileImms.cpp, 426
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileImms.d, 427
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileImms.h, 427
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.cpp, 428
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.d, 428
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLac.h, 428, 429
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.cpp, 429
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.d, 430
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileLog.h, 430
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.cpp, 431
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.d, 431
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileResponse.h, 431, 432
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.cpp, 432
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.d, 433
/src/blacknest/bds/bds/bdsDataLib/BdsDataFileSac.h,

[433, 434](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.cpp, Bds::DataFile, [223](#)
[434](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.d, Bds::DataFileBds, [242](#)
[435](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileStationXml.h, Bds::DataFileCssData, [255](#)
[435, 436](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.cpp, Bds::DataFileSeed, [288](#)
[436](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.d, Bds::DataFormatAll, [306](#)
[437](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileTapeDigitiser.h, Bds::ResponseObj, [366](#)
[437](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.cpp, a
[438](#) Bds::Fir, [324](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.d, AccessGroup
[438](#) Bds::AccessGroup, [52](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWra.h, accessGroupDelete
[438, 439](#) Bds::AdminAccess, [64](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.cpp, accessGroupGetList
[439](#) Bds::AdminAccess, [64](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.d, accessGroupUpdate
[440](#) Bds::AdminAccess, [64](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataFileWraAgso.h, address
[440](#) Bds::User, [399](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.cpp, [441](#) addSource
 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.d, [441](#) Bds::DataCollate, [213](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsDataLib.h, [441](#), AdminAccess
[442](#) Bds::AdminAccess, [60](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.cpp, alias
[442](#) Bds::Source, [383](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.d, Bds::Station, [392](#)
[444](#) ALLOW_TIMESTAMP_JITTER
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsDataFileSeed.h, BdsDataFileBds.cpp, [411](#)
[444, 445](#) BdsDataFileCd.cpp, [418](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.cpp, amplitude
[447](#) Bds::Fap, [322](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.d, apiVersion
[447](#) Bds, [47](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.h, appendDouble
[447](#) Bds::BdsSeedType, [109](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.cpp, appendExp
[448](#) Bds::BdsSeedType, [109](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.d, appendInt
[448](#) Bds::BdsSeedType, [108](#)
 /src/blacknest/bds/bds/bdsDataLib/BdsSeed/BdsSeedType.h, appendString
[448](#) Bds::BdsSeedType, [109](#)
 /src/blacknest/bds/bds/bdsDataLib/canada_compress.d, appendStringVariable
[448](#) Bds::BdsSeedType, [109](#)
 /src/blacknest/bds/bds/bdsDataLib/canada_compress.h, approximationLowerBound
[448, 450](#) Bds::Polynomial, [355](#)
 /src/blacknest/bds/bds/doc/BdsPython.dox, [476](#) approximationType
 /src/blacknest/bds/bds/doc/bdsApi-extra.dox, [476](#) Bds::Polynomial, [355](#)
 /src/blacknest/bds/bds/doc/bdsApiOverview.dox, [476](#) approximationUpperBound
 ~BdsDataPacket Bds::Polynomial, [356](#)
 Bds::BdsDataPacket, [100](#) array
 ~DataCollate Bds::ChannelInfos, [140](#)
 Bds::DataCollate, [212](#) Bds::DataInfo, [310](#)
 Bds::Selection, [371](#)

- ArrayChannel
 - Bds::ArrayChannel, 94
- arrayOffsetEast
 - Bds::ArrayChannel, 95
 - Bds::Location, 336
- arrayOffsetNorth
 - Bds::ArrayChannel, 95
 - Bds::Location, 337
- arrays
 - Bds::SelectionInfo, 375
- arraysAndStations
 - Bds::SelectionInfo, 375
- auth
 - Bds::CdChannel_1v0, 117
 - Bds::CdPacketData, 124
- authKey
 - Bds::CdPacketData, 124
- authSize
 - Bds::CdPacketData, 124
- AvailFull
 - Bds, 32
- AvailNone
 - Bds, 32
- AvailPartial
 - Bds, 32
- AvailType
 - Bds, 32
- availType
 - Bds::DataAvail, 197
- b
 - Bds::Fir, 324
- baseSamplingFrequency
 - Bds::Digitiser, 315
- BDEBUGL1
 - BdsDataFileSac.cpp, 433
 - BdsDataFileSeed.cpp, 443
 - BdsDataFileStationXml.cpp, 435
- BDEBUGL2
 - BdsDataFileSeed.cpp, 443
 - BdsDataFileStationXml.cpp, 435
- BDEBUGL3
 - BdsDataFileSeed.cpp, 443
- Bds, 21
 - apiVersion, 47
 - AvailFull, 32
 - AvailNone, 32
 - AvailPartial, 32
 - AvailType, 32
 - bdsChannelGetName, 35
 - bdsChannelGetTypeAux, 34
 - bdsDataChannelInfo, 38
 - bdsDataChannelOverallResponse, 39
 - bdsDataChannelRef, 38, 39
 - bdsDataFileSeedLogError, 46
 - bdsDataFileSeedLogWarning, 46
 - BdsDataFileVersion, 48
 - bdsDataInfoFlatten, 35
 - bdsDataInfoFromChannelInfos, 36
 - bdsDataInfoFromInfo, 35
 - bdsDataInfoMergeFlatten, 36
 - bdsDataInfoSetTimeRange, 35
 - BdsDataType, 33
 - BdsDataTypeBlock, 33
 - BdsDataTypeData, 33
 - BdsDataTypeInfo, 33
 - BdsDataTypeInfoExtra, 33
 - bdsDataTypes, 42
 - bdsDumpChannelInfos, 38
 - bdsDumpData, 38
 - bdsDumpDataInfo, 37
 - bdsDumpLocation, 38
 - bdsDumpPoleZeros, 34
 - bdsDumpSelection, 37
 - bdsDumpSelectionInfo, 37
 - bdsFileNameExpand, 40
 - bdsInfoFromDataInfo, 35
 - bdsLibInit, 34
 - bdsMetadataExportFix, 37
 - bdsMetadataImportFix, 36
 - bdsPoleZeroGain, 39
 - bdsPoleZeroGainPhase, 39
 - bdsPoleZeroToFap, 40
 - bdsSelectionChannelInfo, 39
 - bdsSpecialChannelIgnore, 41
 - bdsSpecialChannels, 41
 - bdsSpecialChannelsList, 47
 - bdsSpecialChannelsSet, 41
 - bdsStationAlias, 37
 - bdsUnCompressCm8, 42
 - bdsUnCompressSteim1, 42
 - bdsUnitCase, 42
 - bdsUnits, 42
 - bdsUnitsConvert, 36
 - ChannelAuxLen, 48
 - ChannelTypeLen, 48
 - cm6Table, 49
 - cm6TableRev, 49
 - crc, 43
 - crc64, 43
 - crclnit, 43
 - crclnitDone, 49
 - crcVec, 48
 - dataCalculateDifference, 44
 - dataCalculateUnDifference, 44
 - dataChecksum, 44
 - dataCompressCm6, 44
 - dataConvert, 45, 46
 - dataDeCompressCm6, 44
 - DataFlagClipDataToChannels, 30
 - DataFlagClipDataToTime, 30
 - DataFlagMergeSegments, 30
 - DataFlagNoMetadata, 30
 - DataFlagNone, 30
 - DataFlags, 30
 - dataFormatAll, 50
 - DataFormats, 28

- DataFormatSet, [32](#)
- DataFormatSetMetadataRead, [32](#)
- DataFormatSetMetadataWrite, [32](#)
- DataFormatSetNone, [32](#)
- DataFormatSetSensordataRead, [32](#)
- DataFormatSetSensordataWrite, [32](#)
- duplicateDump, [43](#)
- ErrorDataQuality, [29](#)
- ErrorNoMetaData, [29](#)
- Errors, [29](#)
- ErrorSlaveMode, [29](#)
- ErrorTimeStamp, [29](#)
- ErrorValidate, [29](#)
- ErrorValidateBdsFudge, [29](#)
- ErrorValidateDuplicate, [29](#)
- ErrorValidateFilenameTime, [29](#)
- ErrorValidateFix, [29](#)
- ErrorValidateMetaData, [29](#)
- ErrorValidateMissingBlocks, [29](#)
- ErrorValidateReorder, [29](#)
- ErrorValidateTimeBackwards, [29](#)
- FileHeaderType, [33](#)
- FileHeaderType_Standard, [33](#)
- FileHeaderType_TapeDigitiser, [33](#)
- fileNameTime, [40](#)
- FileSampleType, [33](#)
- FileSampleType_Float32, [33](#)
- FileSampleType_Float64, [33](#)
- FileSampleType_Int16, [33](#)
- FileSampleType_Int32, [33](#)
- FileSampleType_Unknown, [33](#)
- fixedString, [44](#)
- fixedWidthValue, [45](#)
- fromSeedTimeString, [46](#)
- getHexString, [43](#)
- LocationSelect, [32](#)
- LocationSelectAll, [33](#)
- LocationSelectChannel, [33](#)
- LocationSelectStation, [33](#)
- Mode, [29](#)
- ModeMaster, [30](#)
- ModeSlave, [30](#)
- NetworkNameLen, [47](#)
- node_types, [49](#)
- nullString, [43](#)
- Priority, [29](#)
- PriorityHigh, [29](#)
- PriorityLow, [29](#)
- PriorityNormal, [29](#)
- record_handler, [47](#)
- removeCR, [45](#)
- responseSort, [36](#)
- roundDigits, [45](#)
- SampleFormat, [30](#)
- SampleFormatFloat32, [32](#)
- SampleFormatFloat64, [32](#)
- SampleFormatInt16, [32](#)
- SampleFormatInt24, [32](#)
- SampleFormatInt32, [32](#)
- SampleFormatUnknown, [32](#)
- Scale, [49](#)
- seedChannelDataType, [41](#)
- seedChannelInstrumentCode, [41](#)
- seedCodeToDataTypes, [47](#)
- seedTime, [46](#)
- seedTimeString, [46](#)
- SelectionGroup, [30](#)
- SelectionGroupData, [30](#)
- SelectionGroupDataWithCount, [30](#)
- SelectionGroupMetaData, [30](#)
- SourceLen, [48](#)
- StationNameLen, [48](#)
- stringFormat, [45](#)
- unitsCode, [45](#)
- Bds::AccessGroup, [51](#)
- AccessGroup, [52](#)
- endTime, [53](#)
- getMember, [53](#)
- getMembers, [52](#)
- getType, [52](#)
- group, [53](#)
- id, [53](#)
- network, [54](#)
- setMember, [52](#)
- setMembers, [52](#)
- startTime, [53](#)
- station, [54](#)
- Bds::AdminAccess, [54](#)
- accessGroupDelete, [64](#)
- accessGroupGetList, [64](#)
- accessGroupUpdate, [64](#)
- AdminAccess, [60](#)
- calibrationDelete, [72](#)
- calibrationGetList, [72](#)
- calibrationUpdate, [72](#)
- changeDelete, [85](#)
- changeGetList, [85](#)
- changeGetListNumber, [85](#)
- changeGroupDelete, [85](#)
- changeGroupEnd, [84](#)
- changeGroupGetList, [85](#)
- changeGroupStart, [84](#)
- channelDelete, [68](#)
- channelGet, [67](#)
- channelGetList, [67](#)
- channelInstrumentDelete, [70](#)
- channelInstrumentGetList, [69](#)
- channelInstrumentUpdate, [70](#)
- channelUpdate, [68](#)
- clean, [90](#)
- connect, [60](#)
- dataAvailability, [77](#)
- databaseBackup, [92](#)
- databaseRestore, [92](#)
- dataChannelDelete, [77](#)
- dataChannelGetList, [76](#)

- dataChannelUpdate, 76
- dataClose, 83
- dataFileDelete, 76
- dataFileGetList, 75
- dataFileUpdate, 76
- dataFormatGetList, 89
- dataFormattedGetLength, 83
- dataFormattedRead, 83
- dataGetBlock, 81
- dataGetChannelInfo, 79
- dataGetInfo, 80
- dataGetNotes, 80
- dataGetWarnings, 81
- dataOpen, 79
- dataPutBlock, 82
- dataRealtimeConfig, 83
- dataRealtimeGet, 84
- dataSearch, 77
- dataSeekBlock, 82
- dataSetInfo, 82
- digitiserDelete, 71
- digitiserGet, 70
- digitiserGetList, 70
- digitiserUpdate, 70
- eventDelete, 74
- eventGetList, 73
- eventUpdate, 73
- extraCall, 93
- getSelectionInfo, 64
- getSelections, 65
- getVersion, 61
- groupDelete, 64
- groupGetList, 63
- groupUpdate, 63
- locationDelete, 67
- locationGetList, 66
- locationUpdate, 67
- logAppend, 88
- logDelete, 87
- logGetList, 87
- logUpdate, 87
- metadataGetChannelInfo, 74
- metadataGetFormatted, 75
- modeSet, 90
- modeSnapshotPause, 90
- networkDelete, 65
- networkGetList, 65
- networkUpdate, 65
- noteDelete, 86
- noteGetList, 86
- noteReadDocument, 87
- noteUpdate, 86
- noteWriteDocument, 86
- responseDelete, 73
- responseGetList, 72
- responseUpdate, 73
- sensorDelete, 72
- sensorGet, 71
- sensorGetList, 71
- sensorUpdate, 71
- serverConfigurationGet, 88
- setUser, 61
- setUserReal, 61
- sourceDelete, 68
- sourceGetList, 68
- sourcePriorityDelete, 69
- sourcePriorityGetList, 69
- sourcePriorityUpdate, 69
- sourceUpdate, 68
- specialChannelDelete, 74
- specialChannelGetList, 74
- specialChannelUpdate, 74
- sqlQuery, 92
- stationDelete, 66
- stationGetList, 66
- stationUpdate, 66
- statisticsGet, 88
- transactionEnd, 89
- transactionStart, 89
- userDelete, 62
- userGet, 62
- userGetFromId, 62
- userGetGroups, 63
- userGetList, 61
- userGetOptions, 63
- userSet, 62
- userSetOptions, 63
- userUpdate, 62
- validateUser, 60
- Bds::ArrayChannel, 93
 - ArrayChannel, 94
 - arrayOffsetEast, 95
 - arrayOffsetNorth, 95
 - channel, 95
 - network, 94
 - station, 94
- Bds::BdsDataBlock, 95
 - data, 96
 - header, 96
- Bds::BdsDataBlockHeader, 96
 - length, 97
 - packetOffset, 97
 - type, 97
- Bds::BdsDataBlockPos, 97
 - BdsDataBlockPos, 98
 - channel, 99
 - endTime, 98
 - numChannels, 99
 - numSamples, 99
 - operator<, 98
 - position, 99
 - segment, 99
 - startTime, 98
- Bds::BdsDataPacket, 99
 - ~BdsDataPacket, 100
 - BdsDataPacket, 100

- clear, 100
- dump, 101
- getHeader, 101
- reset, 100
- setChecksumAndLength, 101
- setHeader, 101
- validateChecksum, 101
- Bds::BdsDataPacketHeader, 101
 - checksum, 103
 - endTime, 103
 - length, 102
 - sequence, 102
 - startTime, 103
 - streamlet, 102
 - type, 102
- Bds::BdsDataSegment, 103
 - BdsDataSegment, 104
 - blocks, 105
 - endTime, 104
 - numBlocks, 104
 - numSamples, 105
 - operator<, 104
 - sampleRate, 105
 - startTime, 104
- Bds::BdsDataStreamlet, 105
 - BdsDataStreamlet, 106
 - blocks, 106
 - channel, 106
 - numChannels, 106
 - packetNumber, 106
 - position, 106
 - segments, 106
- Bds::BdsSeedType, 107
 - appendDouble, 109
 - appendExp, 109
 - appendInt, 108
 - appendString, 109
 - appendStringVariable, 109
 - BdsSeedType, 107
 - getDouble, 108
 - getInt, 108
 - getString, 108
 - getStringVariable, 108
 - getUInt, 108
- Bds::Calibration, 110
 - Calibration, 111
 - calibrationFactor, 114
 - calibrationFrequency, 114
 - calibrationUnits, 115
 - calibrationUnitsDesc, 115
 - channel, 114
 - depth, 115
 - endTime, 113
 - getMember, 112
 - getMembers, 112
 - getType, 112
 - horizontalAngle, 116
 - id, 113
 - name, 114
 - network, 113
 - rawCalibrationFactor, 115
 - rawCalibrationFrequency, 115
 - rawCalibrationUnits, 115
 - samplingFrequency, 114
 - setMember, 112
 - setMembers, 112
 - source, 114
 - startTime, 113
 - station, 113
 - verticalAngle, 116
 - waterLevel, 116
- Bds::CdChannel_1v0, 116
 - auth, 117
 - calibrationFactor, 117
 - calibrationPeriod, 117
 - channel, 118
 - channelName, 118
 - compress, 117
 - name, 117
 - spare0, 117
 - spare1, 117
 - stationName, 118
- Bds::CdDataChannel, 118
 - channel, 119
 - data, 120
 - dataSize, 120
 - mode, 119
 - numSamples, 119
 - period, 119
 - startTime, 119
 - station, 119
 - status, 119
- Bds::CdDataFormatFrame_1v0, 120
 - channels, 121
 - frameLength, 121
 - frameType, 120
 - maxFrameLength, 121
 - numChannels, 121
 - period, 121
- Bds::CdFlag, 121
 - CdFlag, 122
 - dead, 122
 - zeroed, 122
- Bds::CdPacketData, 122
 - auth, 124
 - authKey, 124
 - authSize, 124
 - channels, 124
 - crc, 125
 - creator, 123
 - destination, 123
 - frameType, 123
 - numChannels, 124
 - period, 124
 - sequenceNum, 123
 - series, 124

- startTime, [124](#)
- trailerOffset, [123](#)
- Bds::Change, [125](#)
 - Change, [126](#)
 - changeGroupId, [127](#)
 - getMember, [127](#)
 - getMembers, [126](#)
 - getType, [126](#)
 - id, [127](#)
 - rowId, [128](#)
 - setMember, [126](#)
 - setMembers, [126](#)
 - table, [128](#)
 - time, [127](#)
 - type, [127](#)
- Bds::ChangeGroup, [128](#)
 - ChangeGroup, [129](#)
 - description, [131](#)
 - getMember, [130](#)
 - getMembers, [130](#)
 - getType, [129](#)
 - id, [130](#)
 - setMember, [130](#)
 - setMembers, [129](#)
 - time, [130](#)
 - title, [131](#)
 - type, [131](#)
 - user, [131](#)
- Bds::Channel, [132](#)
 - Channel, [133](#)
 - channel, [135](#)
 - channelAux, [135](#)
 - channelType, [135](#)
 - dataType, [135](#)
 - description, [136](#)
 - endTime, [134](#)
 - getMember, [134](#)
 - getMembers, [134](#)
 - getType, [133](#)
 - id, [134](#)
 - network, [135](#)
 - setMember, [133](#)
 - setMembers, [133](#)
 - startTime, [134](#)
 - station, [135](#)
- Bds::ChannelInfo, [136](#)
 - calibration, [139](#)
 - channel, [138](#)
 - ChannelInfo, [137](#)
 - channelLocation, [138](#)
 - digitiser, [138](#)
 - endTime, [137](#)
 - responses, [139](#)
 - sensor, [138](#)
 - source, [138](#)
 - startTime, [137](#)
 - station, [137](#)
 - stationLocation, [138](#)
- Bds::ChannelInfos, [139](#)
 - array, [140](#)
 - ChannelInfos, [140](#)
 - channels, [140](#)
- Bds::ChannelInstrument, [141](#)
 - channelId, [143](#)
 - ChannelInstrument, [142](#)
 - digitiserId, [144](#)
 - endTime, [143](#)
 - getMember, [143](#)
 - getMembers, [142](#)
 - getType, [142](#)
 - id, [143](#)
 - sensorId, [144](#)
 - setMember, [142](#)
 - setMembers, [142](#)
 - source, [144](#)
 - startTime, [143](#)
- Bds::ChannelName, [144](#)
 - channel, [145](#)
 - ChannelName, [145](#)
 - network, [145](#)
 - source, [146](#)
 - station, [145](#)
- Bds::CleanOptions, [146](#)
 - changes, [147](#)
 - CleanOptions, [146](#)
 - deletedFiles, [147](#)
 - logs, [147](#)
- Bds::CompressSteim1, [147](#)
 - clear, [148](#)
 - CompressSteim1, [148](#)
 - setByteOrder, [148](#)
 - unCompress, [148](#)
- Bds::DataAccess, [149](#)
 - calibrationGetList, [158](#)
 - channelGetList, [155](#)
 - channelInstrumentGetList, [157](#)
 - clean, [170](#)
 - connect, [152](#)
 - DataAccess, [152](#)
 - dataAvailability, [162](#)
 - databaseBackup, [170](#)
 - dataChannelGetList, [156](#)
 - dataClose, [166](#)
 - dataFileGetList, [156](#)
 - dataFormatGetList, [171](#)
 - dataFormattedGetLength, [167](#)
 - dataFormattedRead, [166](#)
 - dataGetBlock, [166](#)
 - dataGetChannelInfo, [163](#)
 - dataGetInfo, [164](#)
 - dataGetNotes, [164](#)
 - dataGetWarnings, [165](#)
 - dataOpen, [163](#)
 - dataRealtimeConfig, [167](#)
 - dataRealtimeGet, [167](#)
 - dataSearch, [162](#)

- dataSeekBlock, 165
- digitiserGet, 158
- digitiserGetList, 157
- eventGetList, 159
- getSelectionInfo, 161
- getSelections, 161
- getVersion, 153
- groupGetList, 155
- locationGetList, 159
- logAppend, 169
- logUpdate, 169
- metadataGetChannelInfo, 160
- metadataGetFormatted, 160
- modeSet, 169
- modeSnapshotPause, 170
- networkGetList, 155
- noteGetList, 168
- noteReadDocument, 168
- noteUpdate, 168
- noteWriteDocument, 168
- responseGetList, 158
- sensorGet, 158
- sensorGetList, 158
- serverConfigurationGet, 171
- setUser, 153
- setUserReal, 153
- sourceGetList, 156
- sourcePriorityGetList, 156
- specialChannelGetList, 160
- stationGetList, 155
- statisticsGet, 171
- userGet, 154
- userGetFromId, 153
- userGetGroups, 154
- userGetOptions, 154
- userSet, 154
- userSetOptions, 154
- validateUser, 152
- Bds::DataAddAccess, 172
 - calibrationGetList, 181
 - channelGetList, 179
 - channelInstrumentGetList, 180
 - clean, 194
 - connect, 176
 - DataAddAccess, 175
 - dataAvailability, 185
 - databaseBackup, 194
 - dataChannelGetList, 180
 - dataClose, 190
 - dataFileGetList, 179
 - dataFormatGetList, 195
 - dataFormattedGetLength, 191
 - dataFormattedRead, 190
 - dataGetBlock, 189
 - dataGetChannelInfo, 186
 - dataGetInfo, 188
 - dataGetNotes, 188
 - dataGetWarnings, 188
 - dataOpen, 187
 - dataPutBlock, 190
 - dataRealtimeConfig, 191
 - dataRealtimeGet, 191
 - dataSearch, 186
 - dataSeekBlock, 189
 - dataSetInfo, 190
 - digitiserGet, 181
 - digitiserGetList, 180
 - eventDelete, 183
 - eventGetList, 182
 - eventUpdate, 183
 - getSelectionInfo, 185
 - getSelections, 185
 - getVersion, 177
 - groupGetList, 178
 - locationGetList, 182
 - logAppend, 193
 - logUpdate, 193
 - metadataGetChannelInfo, 184
 - metadataGetFormatted, 184
 - modeSet, 193
 - modeSnapshotPause, 194
 - networkGetList, 178
 - noteGetList, 192
 - noteReadDocument, 192
 - noteUpdate, 192
 - noteWriteDocument, 192
 - responseGetList, 182
 - sensorGet, 181
 - sensorGetList, 181
 - serverConfigurationGet, 195
 - setUser, 176
 - setUserReal, 177
 - sourceGetList, 179
 - sourcePriorityGetList, 179
 - specialChannelGetList, 183
 - stationGetList, 178
 - statisticsGet, 195
 - userGet, 177
 - userGetFromId, 177
 - userGetGroups, 177
 - userGetOptions, 178
 - userSet, 177
 - userSetOptions, 178
 - validateUser, 176
- Bds::DataAvail, 196
 - availType, 197
 - DataAvail, 196
 - endTime, 197
 - startTime, 197
- Bds::DataAvailChan, 197
 - channel, 199
 - DataAvailChan, 198
 - endTime, 199
 - network, 199
 - segments, 200
 - source, 199

- startTime, [199](#)
- station, [199](#)
- Bds::DataBlock, [200](#)
 - channelData, [202](#)
 - channelNumber, [201](#)
 - DataBlock, [201](#)
 - endTime, [201](#)
 - info, [202](#)
 - segmentNumber, [202](#)
 - startTime, [201](#)
- Bds::DataBlockChannel, [202](#)
 - channel, [203](#)
 - DataBlockChannel, [203](#)
 - network, [203](#)
 - source, [204](#)
 - station, [203](#)
- Bds::DataBlockPos, [204](#)
 - DataBlockPos, [204](#)
 - endTime, [205](#)
 - numSamples, [206](#)
 - operator<, [205](#)
 - order, [205](#)
 - position, [205](#)
 - ref, [205](#)
 - startTime, [205](#)
- Bds::DataChannel, [206](#)
 - channel, [210](#)
 - DataChannel, [207](#)
 - dataFileChannel, [211](#)
 - dataFileId, [211](#)
 - endTime, [209](#)
 - getMember, [209](#)
 - getMembers, [208](#)
 - getType, [208](#)
 - id, [209](#)
 - importFilename, [211](#)
 - importFormat, [211](#)
 - importStartTime, [211](#)
 - info, [212](#)
 - network, [209](#)
 - numBlocks, [210](#)
 - numSamples, [210](#)
 - sampleFormat, [211](#)
 - sampleRate, [210](#)
 - setMember, [208](#)
 - setMembers, [208](#)
 - source, [210](#)
 - startTime, [209](#)
 - station, [210](#)
- Bds::DataCollate, [212](#)
 - ~DataCollate, [212](#)
 - addSource, [213](#)
 - DataCollate, [212](#)
 - readData, [213](#)
- Bds::DataError, [213](#)
 - DataError, [215](#)
 - getErrorNumber, [216](#)
 - getString, [216](#)
- getTitle, [216](#)
- mergeDataInfo, [215](#)
- num, [216](#)
- ochannel, [218](#)
- odescription, [217](#)
- oendTime, [218](#)
- oerrorNumber, [217](#)
- ofilename, [217](#)
- onetwork, [218](#)
- operator int, [217](#)
- osource, [218](#)
- ostartTime, [218](#)
- ostation, [218](#)
- otitle, [217](#)
- ouser, [219](#)
- set, [215](#)
- setString, [216](#)
- setStringUser, [216](#)
- str, [217](#)
- Bds::DataFile, [219](#)
 - ~DataFile, [223](#)
 - close, [223](#)
 - dataErrorFixup, [229](#)
 - DataFile, [222](#)
 - DataOrder, [221](#)
 - DataOrderAll, [221](#)
 - DataOrderChannel, [221](#)
 - DataOrderSample, [221](#)
 - DataOrderUnknown, [221](#)
 - duplicateCheck, [229](#)
 - end, [226](#)
 - FeatureCanRead, [222](#)
 - FeatureCanWrite, [222](#)
 - FeatureNone, [222](#)
 - Features, [221](#)
 - fileNameProcess, [226](#)
 - flush, [226](#)
 - getDataOrder, [224](#)
 - getFeatures, [224](#)
 - getFileName, [224](#)
 - getFilePosition, [230](#)
 - getFixesInfo, [224](#)
 - getFormat, [227](#)
 - getFormats, [230](#)
 - getInfo, [227](#)
 - getMetaData, [229](#)
 - init, [223](#)
 - ofile, [230](#)
 - ofileName, [230](#)
 - ofileNameTime, [230](#)
 - oformat, [231](#)
 - omode, [230](#)
 - open, [223](#)
 - readData, [228](#)
 - ReadOptionDeleteDuplicates, [222](#)
 - ReadOptionFileNameProcess, [222](#)
 - ReadOptionFixCorruptions, [222](#)
 - ReadOptionFixSampleRate, [222](#)

- ReadOptionIgnoreSamplerate, 222
- ReadOptionInfoExtra, 222
- ReadOptionNone, 222
- ReadOptionPrintBlocks, 222
- ReadOptionReorder, 222
- ReadOptionsList, 222
- ReadOptionValidate, 222
- seekBlock, 227
- setFormat, 223
- setInfo, 225
- start, 225
- timeCompare, 229
- writeData, 226
- WriteOptionNoMetadata, 222
- WriteOptionNone, 222
- WriteOptionSensorData, 222
- WriteOptionsList, 222
- Bds::DataFileAd22, 231
 - DataFileAd22, 232
 - getDataOrder, 232
 - getFeatures, 232
 - getFixesInfo, 232
 - getFormats, 233
 - getInfo, 232
 - readData, 233
- Bds::DataFileAscii, 234
 - DataFileAscii, 234
 - end, 237
 - getDataOrder, 235
 - getFeatures, 235
 - getFormats, 237
 - open, 235
 - setFormat, 235
 - setInfo, 236
 - start, 236
 - writeData, 236
- Bds::DataFileBdrs, 237
 - DataFileBdrs, 238
 - getDataOrder, 238
 - getFeatures, 238
 - getFixesInfo, 239
 - getFormats, 240
 - getInfo, 239
 - readData, 239
- Bds::DataFileBds, 240
 - ~DataFileBds, 242
 - close, 243
 - DataFileBds, 242
 - DefaultBlockSize, 242
 - flush, 243
 - getDataOrder, 244
 - getDiskBlockSize, 245
 - getFormats, 245
 - getInfo, 244
 - open, 243
 - packetRead, 246
 - packetWrite, 246
 - PackFormat, 242
 - PackFormat_CM, 242
 - PackFormat_SM, 242
 - PackFormat_SM_CC, 242
 - PackFormat_Unknown, 242
 - readData, 245
 - seekBlock, 244
 - setDiskBlockSize, 245
 - setFormat, 243
 - setInfo, 243
 - setReadPositionToStart, 246
 - setWritePositionForAppend, 246
 - streamletToChannel, 245
 - StreamsMax, 242
 - writeData, 244
- Bds::DataFileBknas, 247
 - DataFileBknas, 247
 - getFormats, 248
 - open, 247
 - setInfo, 248
 - writeData, 248
- Bds::DataFileCd, 249
 - DataFileCd, 250
 - getDataOrder, 250
 - getFeatures, 250
 - getFixesInfo, 250
 - getFormats, 251
 - getInfo, 250
 - readData, 251
- Bds::DataFileCss, 252
 - DataFileCss, 252
 - getDataOrder, 253
 - getFeatures, 253
 - getFormats, 254
 - getInfo, 253
 - readData, 254
- Bds::DataFileCssData, 254
 - ~DataFileCssData, 255
 - calibrationFactor, 257
 - calibrationFreq, 257
 - chan, 256
 - chanid, 256
 - clip, 258
 - commId, 258
 - DataFileCssData, 255
 - datatype, 258
 - dirName, 258
 - endTime, 257
 - file, 258
 - fileName, 258
 - fileOffset, 258
 - instType, 257
 - jdate, 257
 - loadDate, 258
 - nsamp, 257
 - sampleBigEndian, 259
 - sampleFormat, 259
 - sampleRate, 257
 - sampleSize, 259

- segtype, [257](#)
- set, [256](#)
- sta, [256](#)
- startTime, [256](#)
- wfid, [256](#)
- Bds::DataFileGcf, [259](#)
 - DataFileGcf, [260](#)
 - getDataOrder, [260](#)
 - getFeatures, [260](#)
 - getFixesInfo, [261](#)
 - getFormats, [262](#)
 - getInfo, [261](#)
 - readData, [261](#)
- Bds::DataFileIdc, [262](#)
 - DataFileIdc, [263](#)
 - getFeatures, [263](#)
 - getFormats, [264](#)
 - getMetaData, [263](#)
 - setInfo, [264](#)
- Bds::DataFileIm, [265](#)
 - close, [266](#)
 - DataFileIm, [266](#)
 - end, [268](#)
 - getDataOrder, [266](#)
 - getFeatures, [266](#)
 - getFormats, [269](#)
 - getMetaData, [268](#)
 - open, [266](#)
 - setInfo, [267](#)
 - start, [267](#)
 - writeData, [268](#)
- Bds::DataFileInfo, [269](#)
 - comment, [273](#)
 - DataFileInfo, [270](#)
 - endTime, [272](#)
 - format, [272](#)
 - getMember, [271](#)
 - getMembers, [271](#)
 - getType, [271](#)
 - id, [272](#)
 - importTime, [273](#)
 - importUserId, [273](#)
 - location, [272](#)
 - setMember, [271](#)
 - setMembers, [271](#)
 - startTime, [272](#)
 - state, [273](#)
 - stream, [273](#)
 - url, [272](#)
- Bds::DataFileLac, [274](#)
 - DataFileLac, [274](#)
 - getDataOrder, [275](#)
 - getFeatures, [275](#)
 - getFixesInfo, [275](#)
 - getFormats, [276](#)
 - getInfo, [275](#)
 - readData, [276](#)
- Bds::DataFileLog, [276](#)
 - DataFileLog, [277](#)
 - end, [281](#)
 - getDataOrder, [278](#)
 - getFeatures, [278](#)
 - getFormats, [281](#)
 - getInfo, [278](#)
 - open, [277](#)
 - readData, [279](#)
 - setFormat, [279](#)
 - setInfo, [279](#)
 - start, [280](#)
 - writeData, [280](#)
- Bds::DataFileOptions, [281](#)
 - DataFileOptions, [282](#)
 - oignoreBlockList, [282](#)
 - ooptionList, [282](#)
 - operator int, [282](#)
 - operator | =, [282](#)
- Bds::DataFileResponse, [283](#)
 - DataFileResponse, [283](#)
 - getFeatures, [283](#)
 - getFormats, [284](#)
 - getMetaData, [284](#)
 - setInfo, [284](#)
- Bds::DataFileSac, [285](#)
 - DataFileSac, [285](#)
 - getFeatures, [285](#)
 - getFormats, [286](#)
 - getMetaData, [286](#)
 - setInfo, [286](#)
- Bds::DataFileSeed, [287](#)
 - ~DataFileSeed, [288](#)
 - close, [288](#)
 - DataFileSeed, [288](#)
 - end, [292](#)
 - getDataOrder, [288](#)
 - getFeatures, [289](#)
 - getFixesInfo, [289](#)
 - getFormats, [292](#)
 - getInfo, [289](#)
 - getMetaData, [290](#)
 - msrFileWrite, [292](#)
 - omsrErr, [292](#)
 - onoLock, [293](#)
 - readData, [290](#)
 - setFormat, [289](#)
 - setInfo, [291](#)
 - start, [291](#)
 - writeData, [291](#)
- Bds::DataFileStationXml, [293](#)
 - DataFileStationXml, [294](#)
 - getFeatures, [294](#)
 - getFormats, [295](#)
 - getMetaData, [295](#)
 - setInfo, [294](#)
- Bds::DataFileTapeDigitiser, [295](#)
 - DataFileTapeDigitiser, [296](#)
 - getFormats, [297](#)

- getInfo, 296
 - open, 296
 - readData, 297
- Bds::DataFileWra, 297
 - DataFileWra, 298
 - getDataOrder, 299
 - getFeatures, 299
 - getFixesInfo, 299
 - getFormats, 300
 - getInfo, 299
 - readData, 300
 - setFormat, 298
- Bds::DataFileWraAgso, 301
 - DataFileWraAgso, 301
 - getDataOrder, 302
 - getFeatures, 302
 - getFormats, 303
 - getInfo, 302
 - readData, 303
- Bds::DataFormat, 303
 - DataFormat, 304
 - dataRead, 304
 - dataWrite, 305
 - description, 305
 - extension, 305
 - metadataRead, 305
 - metadataWrite, 305
 - names, 304
- Bds::DataFormatAll, 306
 - ~DataFormatAll, 306
 - DataFormatAll, 306
 - findFormat, 307
 - formatGet, 307
 - formatGetExtension, 307
 - formatList, 306
- Bds::DataHandle, 307
 - dataFileId, 308
 - DataHandle, 308
 - handle, 308
- Bds::DataInfo, 309
 - array, 310
 - channels, 311
 - DataInfo, 309
 - description, 310
 - endTime, 310
 - info, 311
 - infoExtra, 311
 - startTime, 310
 - synchronous, 310
 - warnings, 311
- Bds::Digitiser, 312
 - baseSamplingFrequency, 315
 - Digitiser, 313
 - endTime, 314
 - gain, 316
 - getMember, 314
 - getMembers, 314
 - getType, 313
 - id, 314
 - initialSamplingFrequency, 315
 - name, 315
 - numberChannels, 315
 - serialNumber, 315
 - setMember, 313
 - setMembers, 313
 - shared, 316
 - startTime, 314
 - type, 315
- Bds::Event, 316
 - dataChannels, 321
 - description, 321
 - elevation, 320
 - endTime, 319
 - Event, 318
 - eventTime, 319
 - extra, 321
 - id, 318
 - latitude, 320
 - longitude, 320
 - magnitude, 320
 - magnitudeUnits, 320
 - network, 319
 - notes, 321
 - source, 319
 - startTime, 319
 - title, 319
 - type, 318
 - userId, 318
 - waterDepth, 320
- Bds::Fap, 321
 - amplitude, 322
 - Fap, 322
 - frequency, 322
 - phase, 322
- Bds::Fir, 323
 - a, 324
 - b, 324
 - Fir, 323
- Bds::FirEntry, 324
 - coefficient, 325
 - error, 325
 - FirEntry, 325
- Bds::GcfChannel, 325
 - channel, 326
 - format, 326
 - sampleRate, 326
 - streamId, 326
 - systemId, 326
 - type, 326
- Bds::Group, 327
 - description, 329
 - getMember, 328
 - getMembers, 328
 - getType, 328
 - Group, 327
 - group, 329

- id, [329](#)
- setMember, [328](#)
- setMembers, [328](#)
- Bds::ListRange, [329](#)
 - getMember, [331](#)
 - getMembers, [331](#)
 - getType, [330](#)
 - ListRange, [330](#)
 - number, [331](#)
 - reverse, [332](#)
 - setMember, [331](#)
 - setMembers, [330](#)
 - start, [331](#)
- Bds::Location, [332](#)
 - arrayOffsetEast, [336](#)
 - arrayOffsetNorth, [337](#)
 - channel, [336](#)
 - datum, [336](#)
 - elevation, [336](#)
 - endTime, [335](#)
 - getMember, [334](#)
 - getMembers, [334](#)
 - getType, [334](#)
 - id, [335](#)
 - latitude, [336](#)
 - Location, [333](#)
 - longitude, [336](#)
 - network, [335](#)
 - setMember, [334](#)
 - setMembers, [334](#)
 - startTime, [335](#)
 - station, [335](#)
- Bds::Log, [337](#)
 - description, [340](#)
 - getMember, [339](#)
 - getMembers, [339](#)
 - getType, [338](#)
 - id, [339](#)
 - Log, [338](#)
 - priority, [340](#)
 - setMember, [339](#)
 - setMembers, [338](#)
 - subSystem, [340](#)
 - time, [339](#)
 - title, [340](#)
 - type, [340](#)
- Bds::LogSelect, [341](#)
 - LogSelect, [341](#)
 - priority, [342](#)
 - startTime, [341](#)
 - subSystem, [342](#)
 - type, [342](#)
- Bds::Network, [342](#)
 - description, [345](#)
 - getMember, [344](#)
 - getMembers, [344](#)
 - getType, [343](#)
 - id, [344](#)
 - Network, [343](#)
 - network, [344](#)
 - setMember, [344](#)
 - setMembers, [343](#)
 - stations, [345](#)
- Bds::Note, [345](#)
 - channel, [349](#)
 - dataFileId, [350](#)
 - description, [350](#)
 - docFormat, [350](#)
 - docUrl, [350](#)
 - endTime, [348](#)
 - errorNumber, [350](#)
 - eventId, [351](#)
 - getMember, [348](#)
 - getMembers, [347](#)
 - getType, [347](#)
 - id, [348](#)
 - importFilename, [351](#)
 - network, [348](#)
 - Note, [346](#)
 - setMember, [347](#)
 - setMembers, [347](#)
 - source, [349](#)
 - startTime, [348](#)
 - station, [349](#)
 - timeAdded, [349](#)
 - title, [350](#)
 - type, [349](#)
 - user, [349](#)
- Bds::Point, [351](#)
 - Point, [352](#)
 - x, [352](#)
 - y, [352](#)
- Bds::PoleZero, [352](#)
 - poles, [353](#)
 - PoleZero, [353](#)
 - zeros, [353](#)
- Bds::Polynomial, [353](#)
 - approximationLowerBound, [355](#)
 - approximationType, [355](#)
 - approximationUpperBound, [356](#)
 - coefficients, [356](#)
 - frequencyLowerBound, [355](#)
 - frequencyUpperBound, [355](#)
 - maximumError, [356](#)
 - Polynomial, [354](#)
 - transferType, [355](#)
 - validFrequencyUnits, [355](#)
- Bds::PolynomialEntry, [356](#)
 - coefficient, [357](#)
 - measurementMethod, [358](#)
 - minusError, [357](#)
 - plusError, [357](#)
 - PolynomialEntry, [357](#)
- Bds::Response, [358](#)
 - channel, [362](#)
 - decimation, [364](#)

- decimationCorr, [364](#)
- decimationDelay, [364](#)
- decimationOffset, [364](#)
- description, [364](#)
- endTime, [361](#)
- faps, [363](#)
- fir, [363](#)
- gain, [363](#)
- gainFrequency, [363](#)
- id, [361](#)
- inputUnits, [365](#)
- inputUnitsDesc, [365](#)
- measured, [365](#)
- name, [362](#)
- network, [361](#)
- outputUnits, [365](#)
- outputUnitsDesc, [365](#)
- poleZeros, [362](#)
- polynomial, [363](#)
- Response, [360](#)
- sampleRate, [365](#)
- source, [362](#)
- stage, [362](#)
- stageType, [363](#)
- startTime, [361](#)
- station, [361](#)
- symmetry, [364](#)
- type, [362](#)
- Bds::ResponseObj, [366](#)
 - ~ResponseObj, [366](#)
 - getString, [367](#)
 - ResponseObj, [366](#)
 - setString, [367](#)
- Bds::Selection, [367](#)
 - array, [371](#)
 - calibrationName, [371](#)
 - channelId, [370](#)
 - channels, [370](#)
 - completeSegments, [370](#)
 - dataTypes, [371](#)
 - digitiserId, [370](#)
 - endTime, [369](#)
 - eventId, [371](#)
 - excludeChannels, [372](#)
 - id, [369](#)
 - locationSelect, [371](#)
 - name, [371](#)
 - range, [369](#)
 - Selection, [368](#)
 - sensorId, [370](#)
 - sensorOldId, [370](#)
 - startTime, [369](#)
- Bds::SelectionChannel, [372](#)
 - channel, [373](#)
 - network, [373](#)
 - SelectionChannel, [372](#)
 - source, [373](#)
 - station, [373](#)
- Bds::SelectionInfo, [373](#)
 - arrays, [375](#)
 - arraysAndStations, [375](#)
 - channels, [376](#)
 - endTime, [375](#)
 - networks, [375](#)
 - numDataChannels, [376](#)
 - SelectionInfo, [374](#)
 - sources, [376](#)
 - startTime, [375](#)
 - stations, [375](#)
- Bds::Sensor, [376](#)
 - endTime, [379](#)
 - gain, [380](#)
 - gainUnits, [380](#)
 - getMember, [378](#)
 - getMembers, [378](#)
 - getType, [378](#)
 - id, [379](#)
 - name, [379](#)
 - numberChannels, [380](#)
 - oldId, [380](#)
 - Sensor, [377](#)
 - serialNumber, [380](#)
 - setMember, [378](#)
 - setMembers, [378](#)
 - shared, [380](#)
 - startTime, [379](#)
 - type, [379](#)
- Bds::Source, [381](#)
 - alias, [383](#)
 - description, [384](#)
 - getMember, [383](#)
 - getMembers, [382](#)
 - getType, [382](#)
 - id, [383](#)
 - setMember, [382](#)
 - setMembers, [382](#)
 - Source, [382](#)
 - source, [383](#)
 - sourceMeta, [383](#)
- Bds::SourcePriority, [384](#)
 - endTime, [387](#)
 - getMember, [386](#)
 - getMembers, [386](#)
 - getType, [385](#)
 - id, [386](#)
 - priority, [387](#)
 - setMember, [386](#)
 - setMembers, [385](#)
 - source, [387](#)
 - SourcePriority, [385](#)
 - startTime, [386](#)
- Bds::SpecialChannel, [387](#)
 - channel, [390](#)
 - dataType, [391](#)
 - description, [391](#)
 - endTime, [390](#)

- getMember, [389](#)
- getMembers, [389](#)
- getType, [389](#)
- id, [390](#)
- network, [390](#)
- setMember, [389](#)
- setMembers, [389](#)
- SpecialChannel, [388](#)
- startTime, [390](#)
- station, [390](#)
- Bds::Station, [391](#)
 - alias, [392](#)
 - channels, [393](#)
 - description, [393](#)
 - id, [392](#)
 - name, [392](#)
 - network, [392](#)
 - Station, [392](#)
 - type, [393](#)
- Bds::TimePeriod, [393](#)
 - endTime, [395](#)
 - getMember, [395](#)
 - getMembers, [395](#)
 - getType, [394](#)
 - setMember, [395](#)
 - setMembers, [394](#)
 - startTime, [395](#)
 - TimePeriod, [394](#)
- Bds::User, [396](#)
 - address, [399](#)
 - email, [399](#)
 - enabled, [399](#)
 - getMember, [398](#)
 - getMembers, [397](#)
 - getType, [397](#)
 - groups, [399](#)
 - id, [398](#)
 - name, [398](#)
 - password, [398](#)
 - setMember, [397](#)
 - setMembers, [397](#)
 - telephone, [399](#)
 - User, [397](#)
 - user, [398](#)
- BdsC.cc, [450](#)
- BdsC.d, [451](#)
- BdsC.h, [451](#)
- bdsChannelGetName
 - Bds, [35](#)
- bdsChannelGetTypeAux
 - Bds, [34](#)
- BdsD.cc, [455](#)
- BdsD.d, [455](#)
- BdsD.h, [455](#)
- BdsDataBlockPos
 - Bds::BdsDataBlockPos, [98](#)
- bdsDataChannelInfo
 - Bds, [38](#)
- bdsDataChannelOverallResponse
 - Bds, [39](#)
- bdsDataChannelRef
 - Bds, [38](#), [39](#)
- BdsDataFileAd22.cpp
 - DEBUG_VELATRACK, [406](#)
- BdsDataFileBds.cpp
 - ALLOW_TIMESTAMP_JITTER, [411](#)
 - dl2printf, [411](#)
 - dl3printf, [411](#)
 - dlprintf, [411](#)
 - LDEBUG, [411](#)
 - LDEBUG2, [411](#)
 - LDEBUG3, [411](#)
 - TIMESTAMP_JITTER, [412](#)
- BdsDataFileBknas.cpp
 - clip, [416](#)
- BdsDataFileCd.cpp
 - ALLOW_TIMESTAMP_JITTER, [418](#)
 - dprintf, [418](#)
 - ErrorFormatNoDataFormat, [419](#)
 - htonll, [419](#)
 - INCLUDE_CHANNEL_AUTH, [418](#)
 - LDEBUG, [418](#)
 - MULTIPLE_SEGMENT, [418](#)
 - ntohl, [418](#)
 - SEGMENT_GAP, [418](#)
 - TIMESTAMP_JITTER, [418](#)
- BdsDataFileGcf.cpp
 - DEBUG, [423](#)
 - TEST_REORDER, [423](#)
- BdsDataFileIdc.cpp
 - dprintf, [425](#)
 - LDEBUG, [425](#)
- BdsDataFileResponse.cpp
 - dprintf, [431](#)
 - LDEBUG, [431](#)
- BdsDataFileSac.cpp
 - BDEBUGL1, [433](#)
- BdsDataFileSeed.cpp
 - BDEBUGL1, [443](#)
 - BDEBUGL2, [443](#)
 - BDEBUGL3, [443](#)
 - DEBUG, [443](#)
 - DEBUG_BLOCKETTE, [444](#)
 - DEBUG_BLOCKS, [444](#)
 - FILL_BLOCKS, [444](#)
 - ROUND_TIMESTAMPS_US, [444](#)
- BdsDataFileSeed.h
 - MSRecord, [445](#)
- bdsDataFileSeedLogError
 - Bds, [46](#)
- bdsDataFileSeedLogWarning
 - Bds, [46](#)
- BdsDataFileStationXml.cpp
 - BDEBUGL1, [435](#)
 - BDEBUGL2, [435](#)
- BdsDataFileVersion

- Bds, [48](#)
- BdsDataFileWraAgso.cpp
 - parseStringFixedFields, [439](#)
- bdsDataInfoFlatten
 - Bds, [35](#)
- bdsDataInfoFromChannelInfos
 - Bds, [36](#)
- bdsDataInfoFromInfo
 - Bds, [35](#)
- bdsDataInfoMergeFlatten
 - Bds, [36](#)
- bdsDataInfoSetTimeRange
 - Bds, [35](#)
- BdsDataPacket
 - Bds::BdsDataPacket, [100](#)
- BdsDataSegment
 - Bds::BdsDataSegment, [104](#)
- BdsDataStreamlet
 - Bds::BdsDataStreamlet, [106](#)
- BdsDataType
 - Bds, [33](#)
- BdsDataTypeBlock
 - Bds, [33](#)
- BdsDataTypeData
 - Bds, [33](#)
- BdsDataTypeInfo
 - Bds, [33](#)
- BdsDataTypeInfoExtra
 - Bds, [33](#)
- bdsDataTypes
 - Bds, [42](#)
- bdsDumpChannelInfos
 - Bds, [38](#)
- bdsDumpData
 - Bds, [38](#)
- bdsDumpDataInfo
 - Bds, [37](#)
- bdsDumpLocation
 - Bds, [38](#)
- bdsDumpPoleZeros
 - Bds, [34](#)
- bdsDumpSelection
 - Bds, [37](#)
- bdsDumpSelectionInfo
 - Bds, [37](#)
- bdsFileNameExpand
 - Bds, [40](#)
- bdsInfoFromDataInfo
 - Bds, [35](#)
- BdsLib.cpp, [469](#)
- BdsLib.d, [471](#)
- BdsLib.dox, [471](#)
- BdsLib.h, [471](#)
- bdsLibInit
 - Bds, [34](#)
- bdsMetadataExportFix
 - Bds, [37](#)
- bdsMetadataImportFix
 - Bds, [36](#)
- bdsPoleZeroGain
 - Bds, [39](#)
- bdsPoleZeroGainPhase
 - Bds, [39](#)
- bdsPoleZeroToFap
 - Bds, [40](#)
- BdsS.cc, [475](#)
- BdsS.d, [476](#)
- BdsSeedType
 - Bds::BdsSeedType, [107](#)
- bdsSelectionChannelInfo
 - Bds, [39](#)
- bdsSpecialChannelIgnore
 - Bds, [41](#)
- bdsSpecialChannels
 - Bds, [41](#)
- bdsSpecialChannelsList
 - Bds, [47](#)
- bdsSpecialChannelsSet
 - Bds, [41](#)
- bdsStationAlias
 - Bds, [37](#)
- BdsT.cc, [476](#)
- bdsUnCompressCm8
 - Bds, [42](#)
- bdsUnCompressSteim1
 - Bds, [42](#)
- bdsUnitCase
 - Bds, [42](#)
- bdsUnits
 - Bds, [42](#)
- bdsUnitsConvert
 - Bds, [36](#)
- blocks
 - Bds::BdsDataSegment, [105](#)
 - Bds::BdsDataStreamlet, [106](#)
- Calibration
 - Bds::Calibration, [111](#)
- calibration
 - Bds::ChannelInfo, [139](#)
- calibrationDelete
 - Bds::AdminAccess, [72](#)
- calibrationFactor
 - Bds::Calibration, [114](#)
 - Bds::CdChannel_1v0, [117](#)
 - Bds::DataFileCssData, [257](#)
- calibrationFreq
 - Bds::DataFileCssData, [257](#)
- calibrationFrequency
 - Bds::Calibration, [114](#)
- calibrationGetList
 - Bds::AdminAccess, [72](#)
 - Bds::DataAccess, [158](#)
 - Bds::DataAddAccess, [181](#)
- calibrationName
 - Bds::Selection, [371](#)
- calibrationPeriod

- Bds::CdChannel_1v0, 117
- calibrationUnits
 - Bds::Calibration, 115
- calibrationUnitsDesc
 - Bds::Calibration, 115
- calibrationUpdate
 - Bds::AdminAccess, 72
- canada_compress
 - canada_compress.h, 449
- canada_compress.h
 - canada_compress, 449
 - canada_uncompress, 449
 - CANCOMP_CORRUPT, 449
 - CANCOMP_ERR, 448
 - CANCOMP_EXCEED, 449
 - CANCOMP_NOT_20, 449
 - CANCOMP_SUCCESS, 449
- canada_uncompress
 - canada_compress.h, 449
- CANCOMP_CORRUPT
 - canada_compress.h, 449
- CANCOMP_ERR
 - canada_compress.h, 448
- CANCOMP_EXCEED
 - canada_compress.h, 449
- CANCOMP_NOT_20
 - canada_compress.h, 449
- CANCOMP_SUCCESS
 - canada_compress.h, 449
- CdFlag
 - Bds::CdFlag, 122
- chan
 - Bds::DataFileCssData, 256
- Change
 - Bds::Change, 126
- changeDelete
 - Bds::AdminAccess, 85
- changeGetList
 - Bds::AdminAccess, 85
- changeGetListNumber
 - Bds::AdminAccess, 85
- ChangeGroup
 - Bds::ChangeGroup, 129
- changeGroupDelete
 - Bds::AdminAccess, 85
- changeGroupEnd
 - Bds::AdminAccess, 84
- changeGroupGetList
 - Bds::AdminAccess, 85
- changeGroupId
 - Bds::Change, 127
- changeGroupStart
 - Bds::AdminAccess, 84
- changes
 - Bds::CleanOptions, 147
- chanid
 - Bds::DataFileCssData, 256
- Channel
 - Bds::Channel, 133
- channel
 - Bds::ArrayChannel, 95
 - Bds::BdsDataBlockPos, 99
 - Bds::BdsDataStreamlet, 106
 - Bds::Calibration, 114
 - Bds::CdChannel_1v0, 118
 - Bds::CdDataChannel, 119
 - Bds::Channel, 135
 - Bds::ChannelInfo, 138
 - Bds::ChannelName, 145
 - Bds::DataAvailChan, 199
 - Bds::DataBlockChannel, 203
 - Bds::DataChannel, 210
 - Bds::GcfChannel, 326
 - Bds::Location, 336
 - Bds::Note, 349
 - Bds::Response, 362
 - Bds::SelectionChannel, 373
 - Bds::SpecialChannel, 390
- channelAux
 - Bds::Channel, 135
- ChannelAuxLen
 - Bds, 48
- channelData
 - Bds::DataBlock, 202
- channelDelete
 - Bds::AdminAccess, 68
- channelGet
 - Bds::AdminAccess, 67
- channelGetList
 - Bds::AdminAccess, 67
 - Bds::DataAccess, 155
 - Bds::DataAddAccess, 179
- channelId
 - Bds::ChannelInstrument, 143
 - Bds::Selection, 370
- ChannelInfo
 - Bds::ChannelInfo, 137
- ChannelInfos
 - Bds::ChannelInfos, 140
- ChannelInstrument
 - Bds::ChannelInstrument, 142
- channelInstrumentDelete
 - Bds::AdminAccess, 70
- channelInstrumentGetList
 - Bds::AdminAccess, 69
 - Bds::DataAccess, 157
 - Bds::DataAddAccess, 180
- channelInstrumentUpdate
 - Bds::AdminAccess, 70
- channelLocation
 - Bds::ChannelInfo, 138
- ChannelName
 - Bds::ChannelName, 145
- channelName
 - Bds::CdChannel_1v0, 118
- channelNumber

- Bds::DataBlock, 201
- channels
 - Bds::CdDataFormatFrame_1v0, 121
 - Bds::CdPacketData, 124
 - Bds::ChannelInfos, 140
 - Bds::DataInfo, 311
 - Bds::Selection, 370
 - Bds::SelectionInfo, 376
 - Bds::Station, 393
- channelType
 - Bds::Channel, 135
- ChannelTypeLen
 - Bds, 48
- channelUpdate
 - Bds::AdminAccess, 68
- checksum
 - Bds::BdsDataPacketHeader, 103
- clean
 - Bds::AdminAccess, 90
 - Bds::DataAccess, 170
 - Bds::DataAddAccess, 194
- CleanOptions
 - Bds::CleanOptions, 146
- clear
 - Bds::BdsDataPacket, 100
 - Bds::CompressSteim1, 148
- clip
 - Bds::DataFileCssData, 258
 - BdsDataFileBknas.cpp, 416
- close
 - Bds::DataFile, 223
 - Bds::DataFileBds, 243
 - Bds::DataFileImS, 266
 - Bds::DataFileSeed, 288
- cm6Table
 - Bds, 49
- cm6TableRev
 - Bds, 49
- coefficient
 - Bds::FirEntry, 325
 - Bds::PolynomialEntry, 357
- coefficients
 - Bds::Polynomial, 356
- comment
 - Bds::DataFileInfo, 273
- commlId
 - Bds::DataFileCssData, 258
- completeSegments
 - Bds::Selection, 370
- compress
 - Bds::CdChannel_1v0, 117
- CompressSteim1
 - Bds::CompressSteim1, 148
- connect
 - Bds::AdminAccess, 60
 - Bds::DataAccess, 152
 - Bds::DataAddAccess, 176
- crc
 - Bds, 43
 - Bds::CdPacketData, 125
- crc64
 - Bds, 43
- crcInit
 - Bds, 43
- crcInitDone
 - Bds, 49
- crcVec
 - Bds, 48
- creator
 - Bds::CdPacketData, 123
- data
 - Bds::BdsDataBlock, 96
 - Bds::CdDataChannel, 120
- DataAccess
 - Bds::DataAccess, 152
- DataAddAccess
 - Bds::DataAddAccess, 175
- DataAvail
 - Bds::DataAvail, 196
- dataAvailability
 - Bds::AdminAccess, 77
 - Bds::DataAccess, 162
 - Bds::DataAddAccess, 185
- DataAvailChan
 - Bds::DataAvailChan, 198
- databaseBackup
 - Bds::AdminAccess, 92
 - Bds::DataAccess, 170
 - Bds::DataAddAccess, 194
- databaseRestore
 - Bds::AdminAccess, 92
- DataBlock
 - Bds::DataBlock, 201
- DataBlockChannel
 - Bds::DataBlockChannel, 203
- DataBlockPos
 - Bds::DataBlockPos, 204
- dataCalculateDifference
 - Bds, 44
- dataCalculateUnDifference
 - Bds, 44
- DataChannel
 - Bds::DataChannel, 207
- dataChannelDelete
 - Bds::AdminAccess, 77
- dataChannelGetList
 - Bds::AdminAccess, 76
 - Bds::DataAccess, 156
 - Bds::DataAddAccess, 180
- dataChannels
 - Bds::Event, 321
- dataChannelUpdate
 - Bds::AdminAccess, 76
- dataChecksum
 - Bds, 44
- dataClose

- Bds::AdminAccess, [83](#)
- Bds::DataAccess, [166](#)
- Bds::DataAddAccess, [190](#)
- DataCollate
 - Bds::DataCollate, [212](#)
- dataCompressCm6
 - Bds, [44](#)
- dataConvert
 - Bds, [45](#), [46](#)
- dataDeCompressCm6
 - Bds, [44](#)
- DataError
 - Bds::DataError, [215](#)
- dataErrorFixup
 - Bds::DataFile, [229](#)
- DataFile
 - Bds::DataFile, [222](#)
- DataFileAd22
 - Bds::DataFileAd22, [232](#)
- DataFileAscii
 - Bds::DataFileAscii, [234](#)
- DataFileBdrs
 - Bds::DataFileBdrs, [238](#)
- DataFileBds
 - Bds::DataFileBds, [242](#)
- DataFileBknas
 - Bds::DataFileBknas, [247](#)
- DataFileCd
 - Bds::DataFileCd, [250](#)
- dataFileChannel
 - Bds::DataChannel, [211](#)
- DataFileCss
 - Bds::DataFileCss, [252](#)
- DataFileCssData
 - Bds::DataFileCssData, [255](#)
- dataFileDelete
 - Bds::AdminAccess, [76](#)
- DataFileGcf
 - Bds::DataFileGcf, [260](#)
- dataFileGetList
 - Bds::AdminAccess, [75](#)
 - Bds::DataAccess, [156](#)
 - Bds::DataAddAccess, [179](#)
- dataFileId
 - Bds::DataChannel, [211](#)
 - Bds::DataHandle, [308](#)
 - Bds::Note, [350](#)
- DataFileIdc
 - Bds::DataFileIdc, [263](#)
- DataFileImS
 - Bds::DataFileImS, [266](#)
- DataFileInfo
 - Bds::DataFileInfo, [270](#)
- DataFileLac
 - Bds::DataFileLac, [274](#)
- DataFileLog
 - Bds::DataFileLog, [277](#)
- DataFileOptions
 - Bds::DataFileOptions, [282](#)
- DataFileResponse
 - Bds::DataFileResponse, [283](#)
- DataFileSac
 - Bds::DataFileSac, [285](#)
- DataFileSeed
 - Bds::DataFileSeed, [288](#)
- DataFileStationXml
 - Bds::DataFileStationXml, [294](#)
- DataFileTapeDigitiser
 - Bds::DataFileTapeDigitiser, [296](#)
- dataFileUpdate
 - Bds::AdminAccess, [76](#)
- DataFileWra
 - Bds::DataFileWra, [298](#)
- DataFileWraAgso
 - Bds::DataFileWraAgso, [301](#)
- DataFlagClipDataToChannels
 - Bds, [30](#)
- DataFlagClipDataToTime
 - Bds, [30](#)
- DataFlagMergeSegments
 - Bds, [30](#)
- DataFlagNoMetadata
 - Bds, [30](#)
- DataFlagNone
 - Bds, [30](#)
- DataFlags
 - Bds, [30](#)
- DataFormat
 - Bds::DataFormat, [304](#)
- DataFormatAll
 - Bds::DataFormatAll, [306](#)
- dataFormatAll
 - Bds, [50](#)
- dataFormatGetList
 - Bds::AdminAccess, [89](#)
 - Bds::DataAccess, [171](#)
 - Bds::DataAddAccess, [195](#)
- DataFormats
 - Bds, [28](#)
- DataFormatSet
 - Bds, [32](#)
- DataFormatSetMetadataRead
 - Bds, [32](#)
- DataFormatSetMetadataWrite
 - Bds, [32](#)
- DataFormatSetNone
 - Bds, [32](#)
- DataFormatSetSensordataRead
 - Bds, [32](#)
- DataFormatSetSensordataWrite
 - Bds, [32](#)
- dataFormattedGetLength
 - Bds::AdminAccess, [83](#)
 - Bds::DataAccess, [167](#)
 - Bds::DataAddAccess, [191](#)
- dataFormattedRead

- Bds::AdminAccess, [83](#)
- Bds::DataAccess, [166](#)
- Bds::DataAddAccess, [190](#)
- dataGetBlock
 - Bds::AdminAccess, [81](#)
 - Bds::DataAccess, [166](#)
 - Bds::DataAddAccess, [189](#)
- dataGetChannelInfo
 - Bds::AdminAccess, [79](#)
 - Bds::DataAccess, [163](#)
 - Bds::DataAddAccess, [186](#)
- dataGetInfo
 - Bds::AdminAccess, [80](#)
 - Bds::DataAccess, [164](#)
 - Bds::DataAddAccess, [188](#)
- dataGetNotes
 - Bds::AdminAccess, [80](#)
 - Bds::DataAccess, [164](#)
 - Bds::DataAddAccess, [188](#)
- dataGetWarnings
 - Bds::AdminAccess, [81](#)
 - Bds::DataAccess, [165](#)
 - Bds::DataAddAccess, [188](#)
- DataHandle
 - Bds::DataHandle, [308](#)
- DataInfo
 - Bds::DataInfo, [309](#)
- dataOpen
 - Bds::AdminAccess, [79](#)
 - Bds::DataAccess, [163](#)
 - Bds::DataAddAccess, [187](#)
- DataOrder
 - Bds::DataFile, [221](#)
- DataOrderAll
 - Bds::DataFile, [221](#)
- DataOrderChannel
 - Bds::DataFile, [221](#)
- DataOrderSample
 - Bds::DataFile, [221](#)
- DataOrderUnknown
 - Bds::DataFile, [221](#)
- dataPutBlock
 - Bds::AdminAccess, [82](#)
 - Bds::DataAddAccess, [190](#)
- dataRead
 - Bds::DataFormat, [304](#)
- dataRealtimeConfig
 - Bds::AdminAccess, [83](#)
 - Bds::DataAccess, [167](#)
 - Bds::DataAddAccess, [191](#)
- dataRealtimeGet
 - Bds::AdminAccess, [84](#)
 - Bds::DataAccess, [167](#)
 - Bds::DataAddAccess, [191](#)
- dataSearch
 - Bds::AdminAccess, [77](#)
 - Bds::DataAccess, [162](#)
 - Bds::DataAddAccess, [186](#)
- dataSeekBlock
 - Bds::AdminAccess, [82](#)
 - Bds::DataAccess, [165](#)
 - Bds::DataAddAccess, [189](#)
- dataSetInfo
 - Bds::AdminAccess, [82](#)
 - Bds::DataAddAccess, [190](#)
- dataSize
 - Bds::CdDataChannel, [120](#)
- dataType
 - Bds::Channel, [135](#)
 - Bds::SpecialChannel, [391](#)
- datatype
 - Bds::DataFileCssData, [258](#)
- dataTypes
 - Bds::Selection, [371](#)
- dataWrite
 - Bds::DataFormat, [305](#)
- datum
 - Bds::Location, [336](#)
- dead
 - Bds::CdFlag, [122](#)
- DEBUG
 - BdsDataFileGcf.cpp, [423](#)
 - BdsDataFileSeed.cpp, [443](#)
- DEBUG_BLOCKETTE
 - BdsDataFileSeed.cpp, [444](#)
- DEBUG_BLOCKS
 - BdsDataFileSeed.cpp, [444](#)
- DEBUG_VELATRACK
 - BdsDataFileAd22.cpp, [406](#)
- decimation
 - Bds::Response, [364](#)
- decimationCorr
 - Bds::Response, [364](#)
- decimationDelay
 - Bds::Response, [364](#)
- decimationOffset
 - Bds::Response, [364](#)
- DefaultBlockSize
 - Bds::DataFileBds, [242](#)
- deletedFiles
 - Bds::CleanOptions, [147](#)
- depth
 - Bds::Calibration, [115](#)
- description
 - Bds::ChangeGroup, [131](#)
 - Bds::Channel, [136](#)
 - Bds::DataFormat, [305](#)
 - Bds::DataInfo, [310](#)
 - Bds::Event, [321](#)
 - Bds::Group, [329](#)
 - Bds::Log, [340](#)
 - Bds::Network, [345](#)
 - Bds::Note, [350](#)
 - Bds::Response, [364](#)
 - Bds::Source, [384](#)
 - Bds::SpecialChannel, [391](#)

- Bds::Station, [393](#)
- destination
 - Bds::CdPacketData, [123](#)
- Digitiser
 - Bds::Digitiser, [313](#)
- digitiser
 - Bds::ChannelInfo, [138](#)
- digitiserDelete
 - Bds::AdminAccess, [71](#)
- digitiserGet
 - Bds::AdminAccess, [70](#)
 - Bds::DataAccess, [158](#)
 - Bds::DataAddAccess, [181](#)
- digitiserGetList
 - Bds::AdminAccess, [70](#)
 - Bds::DataAccess, [157](#)
 - Bds::DataAddAccess, [180](#)
- digitiserId
 - Bds::ChannelInstrument, [144](#)
 - Bds::Selection, [370](#)
- digitiserUpdate
 - Bds::AdminAccess, [70](#)
- dirName
 - Bds::DataFileCssData, [258](#)
- dl2printf
 - BdsDataFileBds.cpp, [411](#)
- dl3printf
 - BdsDataFileBds.cpp, [411](#)
- dlprintf
 - BdsDataFileBds.cpp, [411](#)
- docFormat
 - Bds::Note, [350](#)
- docUrl
 - Bds::Note, [350](#)
- dprintf
 - BdsDataFileCd.cpp, [418](#)
 - BdsDataFileIdc.cpp, [425](#)
 - BdsDataFileResponse.cpp, [431](#)
- dump
 - Bds::BdsDataPacket, [101](#)
- duplicateCheck
 - Bds::DataFile, [229](#)
- duplicateDump
 - Bds, [43](#)
- elevation
 - Bds::Event, [320](#)
 - Bds::Location, [336](#)
- email
 - Bds::User, [399](#)
- enabled
 - Bds::User, [399](#)
- end
 - Bds::DataFile, [226](#)
 - Bds::DataFileAscii, [237](#)
 - Bds::DataFileIms, [268](#)
 - Bds::DataFileLog, [281](#)
 - Bds::DataFileSeed, [292](#)
- endTime
 - Bds::AccessGroup, [53](#)
 - Bds::BdsDataBlockPos, [98](#)
 - Bds::BdsDataPacketHeader, [103](#)
 - Bds::BdsDataSegment, [104](#)
 - Bds::Calibration, [113](#)
 - Bds::Channel, [134](#)
 - Bds::ChannelInfo, [137](#)
 - Bds::ChannelInstrument, [143](#)
 - Bds::DataAvail, [197](#)
 - Bds::DataAvailChan, [199](#)
 - Bds::DataBlock, [201](#)
 - Bds::DataBlockPos, [205](#)
 - Bds::DataChannel, [209](#)
 - Bds::DataFileCssData, [257](#)
 - Bds::DataFileInfo, [272](#)
 - Bds::DataInfo, [310](#)
 - Bds::Digitiser, [314](#)
 - Bds::Event, [319](#)
 - Bds::Location, [335](#)
 - Bds::Note, [348](#)
 - Bds::Response, [361](#)
 - Bds::Selection, [369](#)
 - Bds::SelectionInfo, [375](#)
 - Bds::Sensor, [379](#)
 - Bds::SourcePriority, [387](#)
 - Bds::SpecialChannel, [390](#)
 - Bds::TimePeriod, [395](#)
- error
 - Bds::FirEntry, [325](#)
- ErrorDataQuality
 - Bds, [29](#)
- ErrorFormatNoDataFormat
 - BdsDataFileCd.cpp, [419](#)
- ErrorNoMetaData
 - Bds, [29](#)
- errorNumber
 - Bds::Note, [350](#)
- Errors
 - Bds, [29](#)
- ErrorSlaveMode
 - Bds, [29](#)
- ErrorTimeStamp
 - Bds, [29](#)
- ErrorValidate
 - Bds, [29](#)
- ErrorValidateBdsFudge
 - Bds, [29](#)
- ErrorValidateDuplicate
 - Bds, [29](#)
- ErrorValidateFilenameTime
 - Bds, [29](#)
- ErrorValidateFix
 - Bds, [29](#)
- ErrorValidateMetaData
 - Bds, [29](#)
- ErrorValidateMissingBlocks
 - Bds, [29](#)
- ErrorValidateReorder

- Bds, [29](#)
- ErrorValidateTimeBackwards
 - Bds, [29](#)
- Event
 - Bds::Event, [318](#)
- eventDelete
 - Bds::AdminAccess, [74](#)
 - Bds::DataAddAccess, [183](#)
- eventGetList
 - Bds::AdminAccess, [73](#)
 - Bds::DataAccess, [159](#)
 - Bds::DataAddAccess, [182](#)
- eventId
 - Bds::Note, [351](#)
 - Bds::Selection, [371](#)
- eventTime
 - Bds::Event, [319](#)
- eventUpdate
 - Bds::AdminAccess, [73](#)
 - Bds::DataAddAccess, [183](#)
- excludeChannels
 - Bds::Selection, [372](#)
- extension
 - Bds::DataFormat, [305](#)
- extra
 - Bds::Event, [321](#)
- extraCall
 - Bds::AdminAccess, [93](#)
- Fap
 - Bds::Fap, [322](#)
- faps
 - Bds::Response, [363](#)
- FeatureCanRead
 - Bds::DataFile, [222](#)
- FeatureCanWrite
 - Bds::DataFile, [222](#)
- FeatureNone
 - Bds::DataFile, [222](#)
- Features
 - Bds::DataFile, [221](#)
- file
 - Bds::DataFileCssData, [258](#)
- FileHeaderType
 - Bds, [33](#)
- FileHeaderType_Standard
 - Bds, [33](#)
- FileHeaderType_TapeDigitiser
 - Bds, [33](#)
- fileName
 - Bds::DataFileCssData, [258](#)
- fileNameProcess
 - Bds::DataFile, [226](#)
- fileNameTime
 - Bds, [40](#)
- fileOffset
 - Bds::DataFileCssData, [258](#)
- FileSampleType
 - Bds, [33](#)
- FileSampleType_Float32
 - Bds, [33](#)
- FileSampleType_Float64
 - Bds, [33](#)
- FileSampleType_Int16
 - Bds, [33](#)
- FileSampleType_Int32
 - Bds, [33](#)
- FileSampleType_Unknown
 - Bds, [33](#)
- FILL_BLOCKS
 - BdsDataFileSeed.cpp, [444](#)
- findFormat
 - Bds::DataFormatAll, [307](#)
- Fir
 - Bds::Fir, [323](#)
- fir
 - Bds::Response, [363](#)
- FirEntry
 - Bds::FirEntry, [325](#)
- fixedString
 - Bds, [44](#)
- fixedWidthValue
 - Bds, [45](#)
- flush
 - Bds::DataFile, [226](#)
 - Bds::DataFileBds, [243](#)
- format
 - Bds::DataFileInfo, [272](#)
 - Bds::GcfChannel, [326](#)
- formatGet
 - Bds::DataFormatAll, [307](#)
- formatGetExtension
 - Bds::DataFormatAll, [307](#)
- formatList
 - Bds::DataFormatAll, [306](#)
- frameLength
 - Bds::CdDataFormatFrame_1v0, [121](#)
- frameType
 - Bds::CdDataFormatFrame_1v0, [120](#)
 - Bds::CdPacketData, [123](#)
- frequency
 - Bds::Fap, [322](#)
- frequencyLowerBound
 - Bds::Polynomial, [355](#)
- frequencyUpperBound
 - Bds::Polynomial, [355](#)
- fromSeedTimeString
 - Bds, [46](#)
- gain
 - Bds::Digitiser, [316](#)
 - Bds::Response, [363](#)
 - Bds::Sensor, [380](#)
- gainFrequency
 - Bds::Response, [363](#)
- gainUnits
 - Bds::Sensor, [380](#)
- getDataOrder

- Bds::DataFile, 224
- Bds::DataFileAd22, 232
- Bds::DataFileAscii, 235
- Bds::DataFileBdrs, 238
- Bds::DataFileBds, 244
- Bds::DataFileCd, 250
- Bds::DataFileCss, 253
- Bds::DataFileGcf, 260
- Bds::DataFileIms, 266
- Bds::DataFileLac, 275
- Bds::DataFileLog, 278
- Bds::DataFileSeed, 288
- Bds::DataFileWra, 299
- Bds::DataFileWraAgso, 302
- getDiskBlockSize
 - Bds::DataFileBds, 245
- getDouble
 - Bds::BdsSeedType, 108
- getErrorNumber
 - Bds::DataError, 216
- getFeatures
 - Bds::DataFile, 224
 - Bds::DataFileAd22, 232
 - Bds::DataFileAscii, 235
 - Bds::DataFileBdrs, 238
 - Bds::DataFileCd, 250
 - Bds::DataFileCss, 253
 - Bds::DataFileGcf, 260
 - Bds::DataFileIdc, 263
 - Bds::DataFileIms, 266
 - Bds::DataFileLac, 275
 - Bds::DataFileLog, 278
 - Bds::DataFileResponse, 283
 - Bds::DataFileSac, 285
 - Bds::DataFileSeed, 289
 - Bds::DataFileStationXml, 294
 - Bds::DataFileWra, 299
 - Bds::DataFileWraAgso, 302
- getFileName
 - Bds::DataFile, 224
- getFilePosition
 - Bds::DataFile, 230
- getFixesInfo
 - Bds::DataFile, 224
 - Bds::DataFileAd22, 232
 - Bds::DataFileBdrs, 239
 - Bds::DataFileCd, 250
 - Bds::DataFileGcf, 261
 - Bds::DataFileLac, 275
 - Bds::DataFileSeed, 289
 - Bds::DataFileWra, 299
- getFormat
 - Bds::DataFile, 227
- getFormats
 - Bds::DataFile, 230
 - Bds::DataFileAd22, 233
 - Bds::DataFileAscii, 237
 - Bds::DataFileBdrs, 240
 - Bds::DataFileBds, 245
 - Bds::DataFileBknas, 248
 - Bds::DataFileCd, 251
 - Bds::DataFileCss, 254
 - Bds::DataFileGcf, 262
 - Bds::DataFileIdc, 264
 - Bds::DataFileIms, 269
 - Bds::DataFileLac, 276
 - Bds::DataFileLog, 281
 - Bds::DataFileResponse, 284
 - Bds::DataFileSac, 286
 - Bds::DataFileSeed, 292
 - Bds::DataFileStationXml, 295
 - Bds::DataFileTapeDigitiser, 297
 - Bds::DataFileWra, 300
 - Bds::DataFileWraAgso, 303
- getHeader
 - Bds::BdsDataPacket, 101
- getHexString
 - Bds, 43
- getInfo
 - Bds::DataFile, 227
 - Bds::DataFileAd22, 232
 - Bds::DataFileBdrs, 239
 - Bds::DataFileBds, 244
 - Bds::DataFileCd, 250
 - Bds::DataFileCss, 253
 - Bds::DataFileGcf, 261
 - Bds::DataFileLac, 275
 - Bds::DataFileLog, 278
 - Bds::DataFileSeed, 289
 - Bds::DataFileTapeDigitiser, 296
 - Bds::DataFileWra, 299
 - Bds::DataFileWraAgso, 302
- getInt
 - Bds::BdsSeedType, 108
- getMember
 - Bds::AccessGroup, 53
 - Bds::Calibration, 112
 - Bds::Change, 127
 - Bds::ChangeGroup, 130
 - Bds::Channel, 134
 - Bds::ChannelInstrument, 143
 - Bds::DataChannel, 209
 - Bds::DataFileInfo, 271
 - Bds::Digitiser, 314
 - Bds::Group, 328
 - Bds::ListRange, 331
 - Bds::Location, 334
 - Bds::Log, 339
 - Bds::Network, 344
 - Bds::Note, 348
 - Bds::Sensor, 378
 - Bds::Source, 383
 - Bds::SourcePriority, 386
 - Bds::SpecialChannel, 389
 - Bds::TimePeriod, 395
 - Bds::User, 398

- getMembers
 - Bds::AccessGroup, [52](#)
 - Bds::Calibration, [112](#)
 - Bds::Change, [126](#)
 - Bds::ChangeGroup, [130](#)
 - Bds::Channel, [134](#)
 - Bds::ChannellInstrument, [142](#)
 - Bds::DataChannel, [208](#)
 - Bds::DataFileInfo, [271](#)
 - Bds::Digitiser, [314](#)
 - Bds::Group, [328](#)
 - Bds::ListRange, [331](#)
 - Bds::Location, [334](#)
 - Bds::Log, [339](#)
 - Bds::Network, [344](#)
 - Bds::Note, [347](#)
 - Bds::Sensor, [378](#)
 - Bds::Source, [382](#)
 - Bds::SourcePriority, [386](#)
 - Bds::SpecialChannel, [389](#)
 - Bds::TimePeriod, [395](#)
 - Bds::User, [397](#)
- getMetaData
 - Bds::DataFile, [229](#)
 - Bds::DataFileIdc, [263](#)
 - Bds::DataFileIms, [268](#)
 - Bds::DataFileResponse, [284](#)
 - Bds::DataFileSac, [286](#)
 - Bds::DataFileSeed, [290](#)
 - Bds::DataFileStationXml, [295](#)
- getSelectionInfo
 - Bds::AdminAccess, [64](#)
 - Bds::DataAccess, [161](#)
 - Bds::DataAddAccess, [185](#)
- getSelections
 - Bds::AdminAccess, [65](#)
 - Bds::DataAccess, [161](#)
 - Bds::DataAddAccess, [185](#)
- getString
 - Bds::BdsSeedType, [108](#)
 - Bds::DataError, [216](#)
 - Bds::ResponseObj, [367](#)
- getStringVariable
 - Bds::BdsSeedType, [108](#)
- getTitle
 - Bds::DataError, [216](#)
- getType
 - Bds::AccessGroup, [52](#)
 - Bds::Calibration, [112](#)
 - Bds::Change, [126](#)
 - Bds::ChangeGroup, [129](#)
 - Bds::Channel, [133](#)
 - Bds::ChannellInstrument, [142](#)
 - Bds::DataChannel, [208](#)
 - Bds::DataFileInfo, [271](#)
 - Bds::Digitiser, [313](#)
 - Bds::Group, [328](#)
 - Bds::ListRange, [330](#)
 - Bds::Location, [334](#)
 - Bds::Log, [338](#)
 - Bds::Network, [343](#)
 - Bds::Note, [347](#)
 - Bds::Sensor, [378](#)
 - Bds::Source, [382](#)
 - Bds::SourcePriority, [385](#)
 - Bds::SpecialChannel, [389](#)
 - Bds::TimePeriod, [394](#)
 - Bds::User, [397](#)
- getUInt
 - Bds::BdsSeedType, [108](#)
- getVersion
 - Bds::AdminAccess, [61](#)
 - Bds::DataAccess, [153](#)
 - Bds::DataAddAccess, [177](#)
- Group
 - Bds::Group, [327](#)
- group
 - Bds::AccessGroup, [53](#)
 - Bds::Group, [329](#)
- groupDelete
 - Bds::AdminAccess, [64](#)
- groupGetList
 - Bds::AdminAccess, [63](#)
 - Bds::DataAccess, [155](#)
 - Bds::DataAddAccess, [178](#)
- groups
 - Bds::User, [399](#)
- groupUpdate
 - Bds::AdminAccess, [63](#)
- handle
 - Bds::DataHandle, [308](#)
- header
 - Bds::BdsDataBlock, [96](#)
- horizontalAngle
 - Bds::Calibration, [116](#)
- htonll
 - BdsDataFileCd.cpp, [419](#)
- id
 - Bds::AccessGroup, [53](#)
 - Bds::Calibration, [113](#)
 - Bds::Change, [127](#)
 - Bds::ChangeGroup, [130](#)
 - Bds::Channel, [134](#)
 - Bds::ChannellInstrument, [143](#)
 - Bds::DataChannel, [209](#)
 - Bds::DataFileInfo, [272](#)
 - Bds::Digitiser, [314](#)
 - Bds::Event, [318](#)
 - Bds::Group, [329](#)
 - Bds::Location, [335](#)
 - Bds::Log, [339](#)
 - Bds::Network, [344](#)
 - Bds::Note, [348](#)
 - Bds::Response, [361](#)
 - Bds::Selection, [369](#)

- Bds::Sensor, [379](#)
- Bds::Source, [383](#)
- Bds::SourcePriority, [386](#)
- Bds::SpecialChannel, [390](#)
- Bds::Station, [392](#)
- Bds::User, [398](#)
- importFilename
 - Bds::DataChannel, [211](#)
 - Bds::Note, [351](#)
- importFormat
 - Bds::DataChannel, [211](#)
- importStartTime
 - Bds::DataChannel, [211](#)
- importTime
 - Bds::DataFileInfo, [273](#)
- importUserId
 - Bds::DataFileInfo, [273](#)
- INCLUDE_CHANNEL_AUTH
 - BdsDataFileCd.cpp, [418](#)
- info
 - Bds::DataBlock, [202](#)
 - Bds::DataChannel, [212](#)
 - Bds::DataInfo, [311](#)
- infoExtra
 - Bds::DataInfo, [311](#)
- init
 - Bds::DataFile, [223](#)
- initialSamplingFrequency
 - Bds::Digitiser, [315](#)
- inputUnits
 - Bds::Response, [365](#)
- inputUnitsDesc
 - Bds::Response, [365](#)
- instType
 - Bds::DataFileCssData, [257](#)
- jdate
 - Bds::DataFileCssData, [257](#)
- latitude
 - Bds::Event, [320](#)
 - Bds::Location, [336](#)
- LDEBUG
 - BdsDataFileBds.cpp, [411](#)
 - BdsDataFileCd.cpp, [418](#)
 - BdsDataFileIdc.cpp, [425](#)
 - BdsDataFileResponse.cpp, [431](#)
- LDEBUG2
 - BdsDataFileBds.cpp, [411](#)
- LDEBUG3
 - BdsDataFileBds.cpp, [411](#)
- length
 - Bds::BdsDataBlockHeader, [97](#)
 - Bds::BdsDataPacketHeader, [102](#)
- ListRange
 - Bds::ListRange, [330](#)
- loadDate
 - Bds::DataFileCssData, [258](#)
- Location
 - Bds::Location, [333](#)
- location
 - Bds::DataFileInfo, [272](#)
- locationDelete
 - Bds::AdminAccess, [67](#)
- locationGetList
 - Bds::AdminAccess, [66](#)
 - Bds::DataAccess, [159](#)
 - Bds::DataAddAccess, [182](#)
- LocationSelect
 - Bds, [32](#)
- locationSelect
 - Bds::Selection, [371](#)
- LocationSelectAll
 - Bds, [33](#)
- LocationSelectChannel
 - Bds, [33](#)
- LocationSelectStation
 - Bds, [33](#)
- locationUpdate
 - Bds::AdminAccess, [67](#)
- Log
 - Bds::Log, [338](#)
- logAppend
 - Bds::AdminAccess, [88](#)
 - Bds::DataAccess, [169](#)
 - Bds::DataAddAccess, [193](#)
- logDelete
 - Bds::AdminAccess, [87](#)
- logGetList
 - Bds::AdminAccess, [87](#)
- logs
 - Bds::CleanOptions, [147](#)
- LogSelect
 - Bds::LogSelect, [341](#)
- logUpdate
 - Bds::AdminAccess, [87](#)
 - Bds::DataAccess, [169](#)
 - Bds::DataAddAccess, [193](#)
- longitude
 - Bds::Event, [320](#)
 - Bds::Location, [336](#)
- magnitude
 - Bds::Event, [320](#)
- magnitudeUnits
 - Bds::Event, [320](#)
- maxFrameLength
 - Bds::CdDataFormatFrame_1v0, [121](#)
- maximumError
 - Bds::Polynomial, [356](#)
- measured
 - Bds::Response, [365](#)
- measurementMethod
 - Bds::PolynomialEntry, [358](#)
- mergeDataInfo
 - Bds::DataError, [215](#)
- metadataGetChannelInfo
 - Bds::AdminAccess, [74](#)

- Bds::DataAccess, 160
- Bds::DataAddAccess, 184
- metadataGetFormatted
 - Bds::AdminAccess, 75
 - Bds::DataAccess, 160
 - Bds::DataAddAccess, 184
- metadataRead
 - Bds::DataFormat, 305
- metadataWrite
 - Bds::DataFormat, 305
- minusError
 - Bds::PolynomialEntry, 357
- Mode
 - Bds, 29
- mode
 - Bds::CdDataChannel, 119
- ModeMaster
 - Bds, 30
- modeSet
 - Bds::AdminAccess, 90
 - Bds::DataAccess, 169
 - Bds::DataAddAccess, 193
- ModeSlave
 - Bds, 30
- modeSnapshotPause
 - Bds::AdminAccess, 90
 - Bds::DataAccess, 170
 - Bds::DataAddAccess, 194
- MSRecord
 - BdsDataFileSeed.h, 445
- msrFileWrite
 - Bds::DataFileSeed, 292
- MULTIPLE_SEGMENT
 - BdsDataFileCd.cpp, 418
- name
 - Bds::Calibration, 114
 - Bds::CdChannel_1v0, 117
 - Bds::Digitiser, 315
 - Bds::Response, 362
 - Bds::Selection, 371
 - Bds::Sensor, 379
 - Bds::Station, 392
 - Bds::User, 398
- names
 - Bds::DataFormat, 304
- Network
 - Bds::Network, 343
- network
 - Bds::AccessGroup, 54
 - Bds::ArrayChannel, 94
 - Bds::Calibration, 113
 - Bds::Channel, 135
 - Bds::ChannelName, 145
 - Bds::DataAvailChan, 199
 - Bds::DataBlockChannel, 203
 - Bds::DataChannel, 209
 - Bds::Event, 319
 - Bds::Location, 335
 - Bds::Network, 344
 - Bds::Note, 348
 - Bds::Response, 361
 - Bds::SelectionChannel, 373
 - Bds::SpecialChannel, 390
 - Bds::Station, 392
- networkDelete
 - Bds::AdminAccess, 65
- networkGetList
 - Bds::AdminAccess, 65
 - Bds::DataAccess, 155
 - Bds::DataAddAccess, 178
- NetworkNameLen
 - Bds, 47
- networks
 - Bds::SelectionInfo, 375
- networkUpdate
 - Bds::AdminAccess, 65
- node_types
 - Bds, 49
- Note
 - Bds::Note, 346
- noteDelete
 - Bds::AdminAccess, 86
- noteGetList
 - Bds::AdminAccess, 86
 - Bds::DataAccess, 168
 - Bds::DataAddAccess, 192
- noteReadDocument
 - Bds::AdminAccess, 87
 - Bds::DataAccess, 168
 - Bds::DataAddAccess, 192
- notes
 - Bds::Event, 321
- noteUpdate
 - Bds::AdminAccess, 86
 - Bds::DataAccess, 168
 - Bds::DataAddAccess, 192
- noteWriteDocument
 - Bds::AdminAccess, 86
 - Bds::DataAccess, 168
 - Bds::DataAddAccess, 192
- nsamp
 - Bds::DataFileCssData, 257
- ntohl
 - BdsDataFileCd.cpp, 418
- nullString
 - Bds, 43
- num
 - Bds::DataError, 216
- number
 - Bds::ListRange, 331
- numberChannels
 - Bds::Digitiser, 315
 - Bds::Sensor, 380
- numBlocks
 - Bds::BdsDataSegment, 104
 - Bds::DataChannel, 210

- numChannels
 - Bds::BdsDataBlockPos, [99](#)
 - Bds::BdsDataStreamlet, [106](#)
 - Bds::CdDataFormatFrame_1v0, [121](#)
 - Bds::CdPacketData, [124](#)
- numDataChannels
 - Bds::SelectionInfo, [376](#)
- numSamples
 - Bds::BdsDataBlockPos, [99](#)
 - Bds::BdsDataSegment, [105](#)
 - Bds::CdDataChannel, [119](#)
 - Bds::DataBlockPos, [206](#)
 - Bds::DataChannel, [210](#)
- ochannel
 - Bds::DataError, [218](#)
- odescription
 - Bds::DataError, [217](#)
- oendTime
 - Bds::DataError, [218](#)
- oerrorNumber
 - Bds::DataError, [217](#)
- ofile
 - Bds::DataFile, [230](#)
- ofileName
 - Bds::DataFile, [230](#)
- ofilename
 - Bds::DataError, [217](#)
- ofilenameTime
 - Bds::DataFile, [230](#)
- oformat
 - Bds::DataFile, [231](#)
- oignoreBlockList
 - Bds::DataFileOptions, [282](#)
- oldId
 - Bds::Sensor, [380](#)
- omode
 - Bds::DataFile, [230](#)
- omsrErr
 - Bds::DataFileSeed, [292](#)
- onetwork
 - Bds::DataError, [218](#)
- onoLock
 - Bds::DataFileSeed, [293](#)
- ooptionList
 - Bds::DataFileOptions, [282](#)
- open
 - Bds::DataFile, [223](#)
 - Bds::DataFileAscii, [235](#)
 - Bds::DataFileBds, [243](#)
 - Bds::DataFileBknas, [247](#)
 - Bds::DataFileIms, [266](#)
 - Bds::DataFileLog, [277](#)
 - Bds::DataFileTapeDigitiser, [296](#)
- operator int
 - Bds::DataError, [217](#)
 - Bds::DataFileOptions, [282](#)
- operator<
 - Bds::BdsDataBlockPos, [98](#)
 - Bds::BdsDataSegment, [104](#)
 - Bds::DataBlockPos, [205](#)
- operator |=
 - Bds::DataFileOptions, [282](#)
- order
 - Bds::DataBlockPos, [205](#)
- osource
 - Bds::DataError, [218](#)
- ostartTime
 - Bds::DataError, [218](#)
- ostation
 - Bds::DataError, [218](#)
- otitle
 - Bds::DataError, [217](#)
- ouser
 - Bds::DataError, [219](#)
- outputUnits
 - Bds::Response, [365](#)
- outputUnitsDesc
 - Bds::Response, [365](#)
- packetNumber
 - Bds::BdsDataStreamlet, [106](#)
- packetOffset
 - Bds::BdsDataBlockHeader, [97](#)
- packetRead
 - Bds::DataFileBds, [246](#)
- packetWrite
 - Bds::DataFileBds, [246](#)
- PackFormat
 - Bds::DataFileBds, [242](#)
- PackFormat_CM
 - Bds::DataFileBds, [242](#)
- PackFormat_SM
 - Bds::DataFileBds, [242](#)
- PackFormat_SM_CC
 - Bds::DataFileBds, [242](#)
- PackFormat_Unknown
 - Bds::DataFileBds, [242](#)
- parseStringFixedFields
 - BdsDataFileWraAgso.cpp, [439](#)
- password
 - Bds::User, [398](#)
- period
 - Bds::CdDataChannel, [119](#)
 - Bds::CdDataFormatFrame_1v0, [121](#)
 - Bds::CdPacketData, [124](#)
- phase
 - Bds::Fap, [322](#)
- plusError
 - Bds::PolynomialEntry, [357](#)
- Point
 - Bds::Point, [352](#)
- poles
 - Bds::PoleZero, [353](#)
- PoleZero
 - Bds::PoleZero, [353](#)
- poleZeros
 - Bds::Response, [362](#)

- Polynomial
 - Bds::Polynomial, [354](#)
- polynomial
 - Bds::Response, [363](#)
- PolynomialEntry
 - Bds::PolynomialEntry, [357](#)
- position
 - Bds::BdsDataBlockPos, [99](#)
 - Bds::BdsDataStreamlet, [106](#)
 - Bds::DataBlockPos, [205](#)
- Priority
 - Bds, [29](#)
- priority
 - Bds::Log, [340](#)
 - Bds::LogSelect, [342](#)
 - Bds::SourcePriority, [387](#)
- PriorityHigh
 - Bds, [29](#)
- PriorityLow
 - Bds, [29](#)
- PriorityNormal
 - Bds, [29](#)
- range
 - Bds::Selection, [369](#)
- rawCalibrationFactor
 - Bds::Calibration, [115](#)
- rawCalibrationFrequency
 - Bds::Calibration, [115](#)
- rawCalibrationUnits
 - Bds::Calibration, [115](#)
- readData
 - Bds::DataCollate, [213](#)
 - Bds::DataFile, [228](#)
 - Bds::DataFileAd22, [233](#)
 - Bds::DataFileBdrs, [239](#)
 - Bds::DataFileBds, [245](#)
 - Bds::DataFileCd, [251](#)
 - Bds::DataFileCss, [254](#)
 - Bds::DataFileGcf, [261](#)
 - Bds::DataFileLac, [276](#)
 - Bds::DataFileLog, [279](#)
 - Bds::DataFileSeed, [290](#)
 - Bds::DataFileTapeDigitiser, [297](#)
 - Bds::DataFileWra, [300](#)
 - Bds::DataFileWraAgso, [303](#)
- ReadOptionDeleteDuplicates
 - Bds::DataFile, [222](#)
- ReadOptionFileNameProcess
 - Bds::DataFile, [222](#)
- ReadOptionFixCorruptions
 - Bds::DataFile, [222](#)
- ReadOptionFixSampleRate
 - Bds::DataFile, [222](#)
- ReadOptionIgnoreSamplerate
 - Bds::DataFile, [222](#)
- ReadOptionInfoExtra
 - Bds::DataFile, [222](#)
- ReadOptionNone
 - Bds::DataFile, [222](#)
- ReadOptionPrintBlocks
 - Bds::DataFile, [222](#)
- ReadOptionReorder
 - Bds::DataFile, [222](#)
- ReadOptionsList
 - Bds::DataFile, [222](#)
- ReadOptionValidate
 - Bds::DataFile, [222](#)
- record_handler
 - Bds, [47](#)
- ref
 - Bds::DataBlockPos, [205](#)
- removeCR
 - Bds, [45](#)
- reset
 - Bds::BdsDataPacket, [100](#)
- Response
 - Bds::Response, [360](#)
- responseDelete
 - Bds::AdminAccess, [73](#)
- responseGetList
 - Bds::AdminAccess, [72](#)
 - Bds::DataAccess, [158](#)
 - Bds::DataAddAccess, [182](#)
- ResponseObj
 - Bds::ResponseObj, [366](#)
- responses
 - Bds::ChannelInfo, [139](#)
- responseSort
 - Bds, [36](#)
- responseUpdate
 - Bds::AdminAccess, [73](#)
- reverse
 - Bds::ListRange, [332](#)
- ROUND_TIMESTAMPS_US
 - BdsDataFileSeed.cpp, [444](#)
- roundDigits
 - Bds, [45](#)
- rowId
 - Bds::Change, [128](#)
- sampleBigEndian
 - Bds::DataFileCssData, [259](#)
- SampleFormat
 - Bds, [30](#)
- sampleFormat
 - Bds::DataChannel, [211](#)
 - Bds::DataFileCssData, [259](#)
- SampleFormatFloat32
 - Bds, [32](#)
- SampleFormatFloat64
 - Bds, [32](#)
- SampleFormatInt16
 - Bds, [32](#)
- SampleFormatInt24
 - Bds, [32](#)
- SampleFormatInt32
 - Bds, [32](#)

- SampleFormatUnknown
 - Bds, [32](#)
- sampleRate
 - Bds::BdsDataSegment, [105](#)
 - Bds::DataChannel, [210](#)
 - Bds::DataFileCssData, [257](#)
 - Bds::GcfChannel, [326](#)
 - Bds::Response, [365](#)
- sampleSize
 - Bds::DataFileCssData, [259](#)
- samplingFrequency
 - Bds::Calibration, [114](#)
- Scale
 - Bds, [49](#)
- seedChannelDataType
 - Bds, [41](#)
- seedChannellInstrumentCode
 - Bds, [41](#)
- seedIcodeToDataTypes
 - Bds, [47](#)
- seedTime
 - Bds, [46](#)
- seedTimeString
 - Bds, [46](#)
- seekBlock
 - Bds::DataFile, [227](#)
 - Bds::DataFileBds, [244](#)
- segment
 - Bds::BdsDataBlockPos, [99](#)
- SEGMENT_GAP
 - BdsDataFileCd.cpp, [418](#)
- segmentNumber
 - Bds::DataBlock, [202](#)
- segments
 - Bds::BdsDataStreamlet, [106](#)
 - Bds::DataAvailChan, [200](#)
- segtype
 - Bds::DataFileCssData, [257](#)
- Selection
 - Bds::Selection, [368](#)
- SelectionChannel
 - Bds::SelectionChannel, [372](#)
- SelectionGroup
 - Bds, [30](#)
- SelectionGroupData
 - Bds, [30](#)
- SelectionGroupDataWithCount
 - Bds, [30](#)
- SelectionGroupMetaData
 - Bds, [30](#)
- SelectionInfo
 - Bds::SelectionInfo, [374](#)
- Sensor
 - Bds::Sensor, [377](#)
- sensor
 - Bds::ChannellInfo, [138](#)
- sensorDelete
 - Bds::AdminAccess, [72](#)
- sensorGet
 - Bds::AdminAccess, [71](#)
 - Bds::DataAccess, [158](#)
 - Bds::DataAddAccess, [181](#)
- sensorGetList
 - Bds::AdminAccess, [71](#)
 - Bds::DataAccess, [158](#)
 - Bds::DataAddAccess, [181](#)
- sensorId
 - Bds::ChannellInstrument, [144](#)
 - Bds::Selection, [370](#)
- sensorOldId
 - Bds::Selection, [370](#)
- sensorUpdate
 - Bds::AdminAccess, [71](#)
- sequence
 - Bds::BdsDataPacketHeader, [102](#)
- sequenceNum
 - Bds::CdPacketData, [123](#)
- serialNumber
 - Bds::Digitiser, [315](#)
 - Bds::Sensor, [380](#)
- series
 - Bds::CdPacketData, [124](#)
- serverConfigurationGet
 - Bds::AdminAccess, [88](#)
 - Bds::DataAccess, [171](#)
 - Bds::DataAddAccess, [195](#)
- set
 - Bds::DataError, [215](#)
 - Bds::DataFileCssData, [256](#)
- setByteOrder
 - Bds::CompressSteim1, [148](#)
- setChecksumAndLength
 - Bds::BdsDataPacket, [101](#)
- setDiskBlockSize
 - Bds::DataFileBds, [245](#)
- setFormat
 - Bds::DataFile, [223](#)
 - Bds::DataFileAscii, [235](#)
 - Bds::DataFileBds, [243](#)
 - Bds::DataFileLog, [279](#)
 - Bds::DataFileSeed, [289](#)
 - Bds::DataFileWra, [298](#)
- setHeader
 - Bds::BdsDataPacket, [101](#)
- setInfo
 - Bds::DataFile, [225](#)
 - Bds::DataFileAscii, [236](#)
 - Bds::DataFileBds, [243](#)
 - Bds::DataFileBknas, [248](#)
 - Bds::DataFileIdc, [264](#)
 - Bds::DataFileIms, [267](#)
 - Bds::DataFileLog, [279](#)
 - Bds::DataFileResponse, [284](#)
 - Bds::DataFileSac, [286](#)
 - Bds::DataFileSeed, [291](#)
 - Bds::DataFileStationXml, [294](#)

- setMember
 - Bds::AccessGroup, 52
 - Bds::Calibration, 112
 - Bds::Change, 126
 - Bds::ChangeGroup, 130
 - Bds::Channel, 133
 - Bds::ChannelInstrument, 142
 - Bds::DataChannel, 208
 - Bds::DataFileInfo, 271
 - Bds::Digitiser, 313
 - Bds::Group, 328
 - Bds::ListRange, 331
 - Bds::Location, 334
 - Bds::Log, 339
 - Bds::Network, 344
 - Bds::Note, 347
 - Bds::Sensor, 378
 - Bds::Source, 382
 - Bds::SourcePriority, 386
 - Bds::SpecialChannel, 389
 - Bds::TimePeriod, 395
 - Bds::User, 397
- setMembers
 - Bds::AccessGroup, 52
 - Bds::Calibration, 112
 - Bds::Change, 126
 - Bds::ChangeGroup, 129
 - Bds::Channel, 133
 - Bds::ChannelInstrument, 142
 - Bds::DataChannel, 208
 - Bds::DataFileInfo, 271
 - Bds::Digitiser, 313
 - Bds::Group, 328
 - Bds::ListRange, 330
 - Bds::Location, 334
 - Bds::Log, 338
 - Bds::Network, 343
 - Bds::Note, 347
 - Bds::Sensor, 378
 - Bds::Source, 382
 - Bds::SourcePriority, 385
 - Bds::SpecialChannel, 389
 - Bds::TimePeriod, 394
 - Bds::User, 397
- setReadPositionToStart
 - Bds::DataFileBds, 246
- setString
 - Bds::DataError, 216
 - Bds::ResponseObj, 367
- setStringUser
 - Bds::DataError, 216
- setUser
 - Bds::AdminAccess, 61
 - Bds::DataAccess, 153
 - Bds::DataAddAccess, 176
- setUserReal
 - Bds::AdminAccess, 61
 - Bds::DataAccess, 153
 - Bds::DataAddAccess, 177
- setWritePositionForAppend
 - Bds::DataFileBds, 246
- shared
 - Bds::Digitiser, 316
 - Bds::Sensor, 380
- Source
 - Bds::Source, 382
- source
 - Bds::Calibration, 114
 - Bds::ChannelInfo, 138
 - Bds::ChannelInstrument, 144
 - Bds::ChannelName, 146
 - Bds::DataAvailChan, 199
 - Bds::DataBlockChannel, 204
 - Bds::DataChannel, 210
 - Bds::Event, 319
 - Bds::Note, 349
 - Bds::Response, 362
 - Bds::SelectionChannel, 373
 - Bds::Source, 383
 - Bds::SourcePriority, 387
- sourceDelete
 - Bds::AdminAccess, 68
- sourceGetList
 - Bds::AdminAccess, 68
 - Bds::DataAccess, 156
 - Bds::DataAddAccess, 179
- SourceLen
 - Bds, 48
- sourceMeta
 - Bds::Source, 383
- SourcePriority
 - Bds::SourcePriority, 385
- sourcePriorityDelete
 - Bds::AdminAccess, 69
- sourcePriorityGetList
 - Bds::AdminAccess, 69
 - Bds::DataAccess, 156
 - Bds::DataAddAccess, 179
- sourcePriorityUpdate
 - Bds::AdminAccess, 69
- sources
 - Bds::SelectionInfo, 376
- sourceUpdate
 - Bds::AdminAccess, 68
- spare0
 - Bds::CdChannel_1v0, 117
- spare1
 - Bds::CdChannel_1v0, 117
- SpecialChannel
 - Bds::SpecialChannel, 388
- specialChannelDelete
 - Bds::AdminAccess, 74
- specialChannelGetList
 - Bds::AdminAccess, 74
 - Bds::DataAccess, 160
 - Bds::DataAddAccess, 183

- specialChannelUpdate
 - Bds::AdminAccess, [74](#)
- sqlQuery
 - Bds::AdminAccess, [92](#)
- sta
 - Bds::DataFileCssData, [256](#)
- stage
 - Bds::Response, [362](#)
- stageType
 - Bds::Response, [363](#)
- start
 - Bds::DataFile, [225](#)
 - Bds::DataFileAscii, [236](#)
 - Bds::DataFileIms, [267](#)
 - Bds::DataFileLog, [280](#)
 - Bds::DataFileSeed, [291](#)
 - Bds::ListRange, [331](#)
- startTime
 - Bds::AccessGroup, [53](#)
 - Bds::BdsDataBlockPos, [98](#)
 - Bds::BdsDataPacketHeader, [103](#)
 - Bds::BdsDataSegment, [104](#)
 - Bds::Calibration, [113](#)
 - Bds::CdDataChannel, [119](#)
 - Bds::CdPacketData, [124](#)
 - Bds::Channel, [134](#)
 - Bds::ChannelInfo, [137](#)
 - Bds::ChannelInstrument, [143](#)
 - Bds::DataAvail, [197](#)
 - Bds::DataAvailChan, [199](#)
 - Bds::DataBlock, [201](#)
 - Bds::DataBlockPos, [205](#)
 - Bds::DataChannel, [209](#)
 - Bds::DataFileCssData, [256](#)
 - Bds::DataFileInfo, [272](#)
 - Bds::DataInfo, [310](#)
 - Bds::Digitiser, [314](#)
 - Bds::Event, [319](#)
 - Bds::Location, [335](#)
 - Bds::LogSelect, [341](#)
 - Bds::Note, [348](#)
 - Bds::Response, [361](#)
 - Bds::Selection, [369](#)
 - Bds::SelectionInfo, [375](#)
 - Bds::Sensor, [379](#)
 - Bds::SourcePriority, [386](#)
 - Bds::SpecialChannel, [390](#)
 - Bds::TimePeriod, [395](#)
- state
 - Bds::DataFileInfo, [273](#)
- Station
 - Bds::Station, [392](#)
- station
 - Bds::AccessGroup, [54](#)
 - Bds::ArrayChannel, [94](#)
 - Bds::Calibration, [113](#)
 - Bds::CdDataChannel, [119](#)
 - Bds::Channel, [135](#)
 - Bds::ChannelInfo, [137](#)
 - Bds::ChannelName, [145](#)
 - Bds::DataAvailChan, [199](#)
 - Bds::DataBlockChannel, [203](#)
 - Bds::DataChannel, [210](#)
 - Bds::Location, [335](#)
 - Bds::Note, [349](#)
 - Bds::Response, [361](#)
 - Bds::SelectionChannel, [373](#)
 - Bds::SpecialChannel, [390](#)
- stationDelete
 - Bds::AdminAccess, [66](#)
- stationGetList
 - Bds::AdminAccess, [66](#)
 - Bds::DataAccess, [155](#)
 - Bds::DataAddAccess, [178](#)
- stationLocation
 - Bds::ChannelInfo, [138](#)
- stationName
 - Bds::CdChannel_v0, [118](#)
- StationNameLen
 - Bds, [48](#)
- stations
 - Bds::Network, [345](#)
 - Bds::SelectionInfo, [375](#)
- stationUpdate
 - Bds::AdminAccess, [66](#)
- statisticsGet
 - Bds::AdminAccess, [88](#)
 - Bds::DataAccess, [171](#)
 - Bds::DataAddAccess, [195](#)
- status
 - Bds::CdDataChannel, [119](#)
- str
 - Bds::DataError, [217](#)
- stream
 - Bds::DataFileInfo, [273](#)
- streamId
 - Bds::GcfChannel, [326](#)
- streamlet
 - Bds::BdsDataPacketHeader, [102](#)
- streamletToChannel
 - Bds::DataFileBds, [245](#)
- StreamsMax
 - Bds::DataFileBds, [242](#)
- stringFormat
 - Bds, [45](#)
- subSystem
 - Bds::Log, [340](#)
 - Bds::LogSelect, [342](#)
- symmetry
 - Bds::Response, [364](#)
- synchronous
 - Bds::DataInfo, [310](#)
- systemId
 - Bds::GcfChannel, [326](#)
- table
 - Bds::Change, [128](#)

- telephone
 - Bds::User, [399](#)
- TEST_REORDER
 - BdsDataFileGcf.cpp, [423](#)
- time
 - Bds::Change, [127](#)
 - Bds::ChangeGroup, [130](#)
 - Bds::Log, [339](#)
- timeAdded
 - Bds::Note, [349](#)
- timeCompare
 - Bds::DataFile, [229](#)
- TimePeriod
 - Bds::TimePeriod, [394](#)
- TIMESTAMP_JITTER
 - BdsDataFileBds.cpp, [412](#)
 - BdsDataFileCd.cpp, [418](#)
- title
 - Bds::ChangeGroup, [131](#)
 - Bds::Event, [319](#)
 - Bds::Log, [340](#)
 - Bds::Note, [350](#)
- trailerOffset
 - Bds::CdPacketData, [123](#)
- transactionEnd
 - Bds::AdminAccess, [89](#)
- transactionStart
 - Bds::AdminAccess, [89](#)
- transferType
 - Bds::Polynomial, [355](#)
- type
 - Bds::BdsDataBlockHeader, [97](#)
 - Bds::BdsDataPacketHeader, [102](#)
 - Bds::Change, [127](#)
 - Bds::ChangeGroup, [131](#)
 - Bds::Digitiser, [315](#)
 - Bds::Event, [318](#)
 - Bds::GcfChannel, [326](#)
 - Bds::Log, [340](#)
 - Bds::LogSelect, [342](#)
 - Bds::Note, [349](#)
 - Bds::Response, [362](#)
 - Bds::Sensor, [379](#)
 - Bds::Station, [393](#)
- unCompress
 - Bds::CompressSteim1, [148](#)
- unitsCode
 - Bds, [45](#)
- url
 - Bds::DataFileInfo, [272](#)
- User
 - Bds::User, [397](#)
- user
 - Bds::ChangeGroup, [131](#)
 - Bds::Note, [349](#)
 - Bds::User, [398](#)
- userDelete
 - Bds::AdminAccess, [62](#)
- userGet
 - Bds::AdminAccess, [62](#)
 - Bds::DataAccess, [154](#)
 - Bds::DataAddAccess, [177](#)
- userGetFromId
 - Bds::AdminAccess, [62](#)
 - Bds::DataAccess, [153](#)
 - Bds::DataAddAccess, [177](#)
- userGetGroups
 - Bds::AdminAccess, [63](#)
 - Bds::DataAccess, [154](#)
 - Bds::DataAddAccess, [177](#)
- userGetList
 - Bds::AdminAccess, [61](#)
- userGetOptions
 - Bds::AdminAccess, [63](#)
 - Bds::DataAccess, [154](#)
 - Bds::DataAddAccess, [178](#)
- userId
 - Bds::Event, [318](#)
- userSet
 - Bds::AdminAccess, [62](#)
 - Bds::DataAccess, [154](#)
 - Bds::DataAddAccess, [177](#)
- userSetOptions
 - Bds::AdminAccess, [63](#)
 - Bds::DataAccess, [154](#)
 - Bds::DataAddAccess, [178](#)
- userUpdate
 - Bds::AdminAccess, [62](#)
- validateChecksum
 - Bds::BdsDataPacket, [101](#)
- validateUser
 - Bds::AdminAccess, [60](#)
 - Bds::DataAccess, [152](#)
 - Bds::DataAddAccess, [176](#)
- validFrequencyUnits
 - Bds::Polynomial, [355](#)
- verticalAngle
 - Bds::Calibration, [116](#)
- warnings
 - Bds::DataInfo, [311](#)
- waterDepth
 - Bds::Event, [320](#)
- waterLevel
 - Bds::Calibration, [116](#)
- wfid
 - Bds::DataFileCssData, [256](#)
- writeData
 - Bds::DataFile, [226](#)
 - Bds::DataFileAscii, [236](#)
 - Bds::DataFileBds, [244](#)
 - Bds::DataFileBknas, [248](#)
 - Bds::DataFileIms, [268](#)
 - Bds::DataFileLog, [280](#)
 - Bds::DataFileSeed, [291](#)
- WriteOptionNoMetadata

Bds::DataFile, [222](#)
WriteOptionNone
 Bds::DataFile, [222](#)
WriteOptionSensorData
 Bds::DataFile, [222](#)
WriteOptionsList
 Bds::DataFile, [222](#)

x
 Bds::Point, [352](#)

y
 Bds::Point, [352](#)

zeroed
 Bds::CdFlag, [122](#)
zeros
 Bds::PoleZero, [353](#)