# Fledge

Generated by Doxygen 1.8.17

**1 Contributing to Fledge** 1

**2 Hierarchical Index** 3

2.1 Class Hierarchy . . . 3

**3 Class Index** 7

3.1 Class List . . . 7

**4 Class Documentation** 15

4.1 ACL Class Reference . . . 15

4.1.1 Detailed Description . . . 15

4.2 ACLMalformed Class Reference . . . 16

4.2.1 Detailed Description . . . 16

4.3 ACL::ACLReason Class Reference . . . 17

4.3.1 Detailed Description . . . 17

4.3.2 Constructor & Destructor Documentation . . . 17

4.3.2.1 ACLReason() . . . 17

4.4 ACLReasonMalformed Class Reference . . . 18

4.4.1 Detailed Description . . . 18

4.5 Aggregate Class Reference . . . 19

4.5.1 Detailed Description . . . 19

4.6 AllCategoriesEx Class Reference . . . 19

4.6.1 Detailed Description . . . 20

4.7 AssetTracker Class Reference . . . 20

4.7.1 Detailed Description . . . 21

4.7.2 Constructor & Destructor Documentation . . . 21

4.7.2.1 AssetTracker() . . . 21

4.7.2.2 ~AssetTracker() . . . 22

4.7.3 Member Function Documentation . . . 22

4.7.3.1 addAssetTrackingTuple() . . . 22

4.7.3.2 addStorageAssetTrackingTuple() . . . 22

4.7.3.3 checkAssetTrackingCache() . . . 23

4.7.3.4 findAssetTrackingCache() . . . 23

4.7.3.5 findStorageAssetTrackingCache() . . . 23

4.7.3.6 getAssetTracker() . . . 24

4.7.3.7 getDeprecated() . . . 24

4.7.3.8 getStorageAssetTrackingCacheData() . . . 24

4.7.3.9 populateAssetTrackingCache() . . . 25

4.7.3.10 populateStorageAssetTrackingCache() . . . 25

4.7.3.11 tune() . . . 25

4.7.3.12 updateCache() . . . 26

4.8 AssetTrackingTable Class Reference . . . 26

4.8.1 Detailed Description . . . 26

4.8.2 Member Function Documentation . . . 26
4.8.2.1 add() . . . 26
4.8.2.2 find() . . . 27
4.9 AssetTrackingTuple Class Reference . . . 27
4.9.1 Detailed Description . . . 28
4.9.2 Member Function Documentation . . . 28
4.9.2.1 processData() . . . 29
4.10 AssetTrackingTuplePtrEqual Struct Reference . . . 29
4.11 AttachDbSync Class Reference . . . 29
4.11.1 Detailed Description . . . 30
4.12 AuditLogger Class Reference . . . 30
4.12.1 Detailed Description . . . 30
4.12.2 Constructor & Destructor Documentation . . . 30
4.12.2.1 AuditLogger() . . . 30
4.12.3 Member Function Documentation . . . 31
4.12.3.1 audit() . . . 31
4.13 BadRequest Class Reference . . . 31
4.13.1 Detailed Description . . . 32
4.14 Base64DataBuffer Class Reference . . . 32
4.14.1 Detailed Description . . . 33
4.14.2 Member Function Documentation . . . 33
4.14.2.1 encode() . . . 33
4.15 Base64DPImage Class Reference . . . 34
4.15.1 Detailed Description . . . 34
4.15.2 Member Function Documentation . . . 35
4.15.2.1 encode() . . . 35
4.16 BearerToken Class Reference . . . 35
4.16.1 Detailed Description . . . 35
4.16.2 Constructor & Destructor Documentation . . . 35
4.16.2.1 BearerToken() . . . 35
4.16.3 Member Function Documentation . . . 36
4.16.3.1 verify() . . . 36
4.17 BinaryPluginHandle Class Reference . . . 36
4.17.1 Detailed Description . . . 37
4.18 CategoryDetailsEx Class Reference . . . 37
4.18.1 Detailed Description . . . 38
4.19 ConfigCategory::CategoryItem Class Reference . . . 38
4.19.1 Member Function Documentation . . . 39
4.19.1.1 defaultToJSON() . . . 39
4.19.1.2 toJSON() . . . 39
4.20 CheckUpdates Class Reference . . . 40
4.20.1 Detailed Description . . . 41

4.21 ChildCategoriesEx Class Reference . . . 41
4.21.1 Detailed Description . . . 42
4.22 ResultSet::ColumnValue Class Reference . . . 42
4.22.1 Member Function Documentation . . . 42
4.22.1.1 getInteger() . . . 43
4.22.1.2 getNumber() . . . 43
4.22.1.3 getString() . . . 43
4.23 ConfigCategories Class Reference . . . 44
4.23.1 Constructor & Destructor Documentation . . . 44
4.23.1.1 ConfigCategories() . . . 44
4.23.2 Member Function Documentation . . . 44
4.23.2.1 addCategoryDescription() . . . 44
4.24 ConfigCategory Class Reference . . . 45
4.24.1 Constructor & Destructor Documentation . . . 48
4.24.1.1 ConfigCategory() [1/2] . . . 48
4.24.1.2 ConfigCategory() [2/2] . . . 48
4.24.2 Member Function Documentation . . . 48
4.24.2.1 checkDefaultValuesOnly() . . . 49
4.24.2.2 extractSubcategory() . . . 49
4.24.2.3 getBoolValue() . . . 49
4.24.2.4 getDefault() . . . 50
4.24.2.5 getItemAttribute() . . . 50
4.24.2.6 getLength() . . . 50
4.24.2.7 getMaximum() . . . 51
4.24.2.8 getMinimum() . . . 51
4.24.2.9 getOptions() . . . 52
4.24.2.10 getPermissions() . . . 52
4.24.2.11 getType() . . . 53
4.24.2.12 getValue() . . . 53
4.24.2.13 getValueKVList() . . . 54
4.24.2.14 getValueList() . . . 54
4.24.2.15 hasPermission() . . . 54
4.24.2.16 isBool() . . . 55
4.24.2.17 isDeprecated() . . . 55
4.24.2.18 isDouble() . . . 56
4.24.2.19 isEnumeration() . . . 56
4.24.2.20 isJSON() . . . 57
4.24.2.21 isKVList() . . . 57
4.24.2.22 isList() . . . 58
4.24.2.23 isNumber() . . . 58
4.24.2.24 isString() . . . 59
4.24.2.25 itemExists() . . . 59

4.24.2.26 itemsToJSON() . . . 59
4.24.2.27 itemToJSON() . . . 60
4.24.2.28 keepItemsType() . . . 60
4.24.2.29 parseBucketItemValue() . . . 60
4.24.2.30 removeItems() . . . 61
4.24.2.31 removeItemsType() . . . 61
4.24.2.32 setDefault() . . . 61
4.24.2.33 setDescription() . . . 62
4.24.2.34 setItemAttribute() . . . 62
4.24.2.35 setItemDisplayName() . . . 62
4.24.2.36 setValue() . . . 63
4.24.2.37 to_string() . . . 63
4.24.2.38 toJSON() . . . 63
4.25 ConfigCategoryChange Class Reference . . . 65
4.25.1 Constructor & Destructor Documentation . . . 66
4.25.1.1 ConfigCategoryChange() . . . 66
4.26 ConfigCategoryDefaultWithValue Class Reference . . . 66
4.26.1 Detailed Description . . . 67
4.27 ConfigCategoryDescription Class Reference . . . 67
4.28 ConfigCategoryEx Class Reference . . . 68
4.28.1 Detailed Description . . . 68
4.29 ConfigHandler Class Reference . . . 69
4.29.1 Detailed Description . . . 69
4.29.2 Member Function Documentation . . . 70
4.29.2.1 configChange() . . . 70
4.29.2.2 configChildCreate() . . . 70
4.29.2.3 configChildDelete() . . . 70
4.29.2.4 registerCategory() . . . 71
4.29.2.5 registerCategoryChild() . . . 71
4.29.2.6 unregisterCategory() . . . 71
4.30 ConfigItemAttributeNotFound Class Reference . . . 72
4.30.1 Detailed Description . . . 72
4.31 ConfigItemNotAList Class Reference . . . 73
4.31.1 Detailed Description . . . 73
4.32 ConfigItemNotFound Class Reference . . . 74
4.33 ConfigMalformed Class Reference . . . 75
4.34 ConfigurationManager Class Reference . . . 75
4.34.1 Member Function Documentation . . . 76
4.34.1.1 addChildCategory() . . . 76
4.34.1.2 createCategory() . . . 77
4.34.1.3 deleteCategory() . . . 77
4.34.1.4 deleteCategoryItemValue() . . . 79

4.34.1.5 deleteChildCategory() . . . 79
4.34.1.6 getAllCategoryNames() . . . 80
4.34.1.7 getCategoryAllItems() . . . 80
4.34.1.8 getCategoryItem() . . . 81
4.34.1.9 getCategoryItemValue() . . . 81
4.34.1.10 getChildCategories() . . . 82
4.34.1.11 getInstance() . . . 82
4.34.1.12 setCategoryItemValue() . . . 82
4.35 ConfigValueFoundWithDefault Class Reference . . . 83
4.35.1 Detailed Description . . . 84
4.36 Conflict Class Reference . . . 84
4.36.1 Detailed Description . . . 85
4.37 Connection Class Reference . . . 85
4.37.1 Constructor & Destructor Documentation . . . 87
4.37.1.1 Connection() [1/2] . . . 87
4.37.1.2 ~Connection() . . . 88
4.37.1.3 Connection() [2/2] . . . 88
4.37.2 Member Function Documentation . . . 88
4.37.2.1 create_schema() . . . 88
4.37.2.2 createSchema() . . . 88
4.37.2.3 fetchReadings() . . . 89
4.37.2.4 findSchemaFromDB() . . . 89
4.37.2.5 formatDate() . . . 90
4.37.2.6 parseDatabaseStorageSchema() . . . 90
4.37.2.7 purgeReadingsByRows() . . . 91
4.37.2.8 readingStream() . . . 91
4.37.2.9 setTrace() . . . 91
4.37.2.10 vacuum() . . . 92
4.38 ConnectionManager Class Reference . . . 92
4.38.1 Detailed Description . . . 94
4.38.2 Member Function Documentation . . . 94
4.38.2.1 allocate() . . . 94
4.38.2.2 allowMoreDatabases() . . . 94
4.38.2.3 attachNewDb() . . . 94
4.38.2.4 attachRequestNewDb() . . . 95
4.38.2.5 background() . . . 95
4.38.2.6 detachNewDb() . . . 95
4.38.2.7 getInstance() . . . 95
4.38.2.8 growPool() . . . 96
4.38.2.9 release() . . . 96
4.38.2.10 setError() . . . 96
4.38.2.11 setPurgeBlockSize() . . . 96

4.38.2.12 shrinkPool() . . . 97
4.38.2.13 shutdown() . . . 97
4.39 CoreManagementApi Class Reference . . . 97
4.39.1 Detailed Description . . . 99
4.39.2 Member Function Documentation . . . 99
4.39.2.1 addChildCategory() . . . 99
4.39.2.2 createCategory() . . . 100
4.39.2.3 getInstance() . . . 100
4.39.2.4 registerMicroService() . . . 100
4.40 DataBuffer Class Reference . . . 100
4.40.1 Detailed Description . . . 101
4.40.2 Constructor & Destructor Documentation . . . 101
4.40.2.1 DataBuffer() [1/2] . . . 101
4.40.2.2 DataBuffer() [2/2] . . . 102
4.40.3 Member Function Documentation . . . 102
4.40.3.1 populate() . . . 102
4.41 DataLoad Class Reference . . . 102
4.41.1 Detailed Description . . . 104
4.41.2 Constructor & Destructor Documentation . . . 104
4.41.2.1 DataLoad() . . . 105
4.41.2.2 ~DataLoad() . . . 105
4.41.3 Member Function Documentation . . . 105
4.41.3.1 configChange() . . . 105
4.41.3.2 fetchReadings() . . . 105
4.41.3.3 passToOnwardFilter() . . . 106
4.41.3.4 pipelineEnd() . . . 106
4.41.3.5 setDataSource() . . . 107
4.41.3.6 triggerRead() . . . 107
4.42 Datapoint Class Reference . . . 107
4.42.1 Detailed Description . . . 108
4.42.2 Member Function Documentation . . . 108
4.42.2.1 parseJson() . . . 108
4.42.2.2 recursiveJson() . . . 109
4.43 DatapointValue Class Reference . . . 109
4.43.1 Detailed Description . . . 110
4.43.2 Constructor & Destructor Documentation . . . 111
4.43.2.1 ~DatapointValue() . . . 111
4.43.3 Member Function Documentation . . . 111
4.43.3.1 setValue() [1/4] . . . 111
4.43.3.2 setValue() [2/4] . . . 111
4.43.3.3 setValue() [3/4] . . . 111
4.43.3.4 setValue() [4/4] . . . 112

4.43.3.5 toString() . . . 112
4.44 DataSender Class Reference . . . 112
4.44.1 Member Function Documentation . . . 113
4.44.1.1 configChange() . . . 113
4.44.1.2 isDryRun() . . . 113
4.44.1.3 pause() . . . 113
4.44.1.4 release() . . . 114
4.45 DefaultConfigCategory Class Reference . . . 114
4.45.1 Detailed Description . . . 115
4.45.2 Constructor & Destructor Documentation . . . 115
4.45.2.1 ~DefaultConfigCategory() . . . 115
4.46 DiskSpaceMonitor Class Reference . . . 115
4.46.1 Detailed Description . . . 116
4.46.2 Constructor & Destructor Documentation . . . 116
4.46.2.1 DiskSpaceMonitor() . . . 116
4.46.3 Member Function Documentation . . . 116
4.46.3.1 periodic() . . . 116
4.47 DPImage Class Reference . . . 117
4.47.1 Detailed Description . . . 118
4.47.2 Constructor & Destructor Documentation . . . 118
4.47.2.1 DPImage() [1/2] . . . 118
4.47.2.2 DPImage() [2/2] . . . 118
4.47.3 Member Function Documentation . . . 119
4.47.3.1 operator=() . . . 119
4.48 ExistingChildCategories Class Reference . . . 119
4.48.1 Detailed Description . . . 120
4.49 Expression Class Reference . . . 120
4.49.1 Detailed Description . . . 120
4.50 ExpressionValues Class Reference . . . 121
4.51 FormData::FieldValue Class Reference . . . 121
4.52 FilterPipeline Class Reference . . . 122
4.52.1 Detailed Description . . . 123
4.52.2 Constructor & Destructor Documentation . . . 123
4.52.2.1 FilterPipeline() . . . 124
4.52.3 Member Function Documentation . . . 124
4.52.3.1 attachDebugger() . . . 124
4.52.3.2 cleanupFilters() . . . 124
4.52.3.3 configChange() . . . 125
4.52.3.4 execute() . . . 125
4.52.3.5 getDebuggerBuffer() . . . 125
4.52.3.6 loadFilters() . . . 125
4.52.3.7 setDebuggerBuffer() . . . 126

4.52.3.8 setupFiltersPipeline() . . . 126
4.53 FilterPlugin Class Reference . . . 127
4.53.1 Constructor & Destructor Documentation . . . 128
4.53.1.1 FilterPlugin() . . . 128
4.53.2 Member Function Documentation . . . 128
4.53.2.1 ingest() . . . 128
4.53.2.2 init() . . . 129
4.53.2.3 reconfigure() . . . 129
4.53.2.4 shutdownSaveData() . . . 129
4.53.2.5 startData() . . . 130
4.54 FilterPythonPluginHandle Class Reference . . . 130
4.54.1 Detailed Description . . . 131
4.54.2 Constructor & Destructor Documentation . . . 131
4.54.2.1 FilterPythonPluginHandle() . . . 131
4.55 FledgeFilter Class Reference . . . 132
4.55.1 Constructor & Destructor Documentation . . . 132
4.55.1.1 FledgeFilter() . . . 133
4.55.2 Member Function Documentation . . . 134
4.55.2.1 setConfig() . . . 134
4.56 FledgeProcess Class Reference . . . 134
4.56.1 Detailed Description . . . 135
4.56.2 Member Function Documentation . . . 135
4.56.2.1 getArgValue() . . . 135
4.57 FormData Class Reference . . . 136
4.57.1 Detailed Description . . . 136
4.57.2 Member Function Documentation . . . 136
4.57.2.1 getUploadedData() . . . 136
4.57.2.2 getUploadedFile() . . . 137
4.57.2.3 saveFile() . . . 137
4.58 std::hash< AssetTrackingTuple * > Struct Reference . . . 137
4.59 std::hash< AssetTrackingTuple > Struct Reference . . . 138
4.60 std::hash< StorageAssetTrackingTuple * > Struct Reference . . . 138
4.61 std::hash< StorageAssetTrackingTuple > Struct Reference . . . 138
4.62 HttpSender Class Reference . . . 138
4.62.1 Constructor & Destructor Documentation . . . 139
4.62.1.1 HttpSender() . . . 139
4.62.2 Member Function Documentation . . . 139
4.62.2.1 createDebugTraceDirectory() . . . 140
4.62.2.2 getOMFTracePath() . . . 140
4.63 Ingest Class Reference . . . 140
4.63.1 Detailed Description . . . 142
4.63.2 Constructor & Destructor Documentation . . . 142

4.63.2.1 Ingest() . . . 143
4.63.2.2 ~Ingest() . . . 143
4.63.3 Member Function Documentation . . . 143
4.63.3.1 configChange() . . . 143
4.63.3.2 configureRateMonitor() . . . 144
4.63.3.3 createStatsDbEntry() . . . 144
4.63.3.4 flowControl() . . . 144
4.63.3.5 ingest() [1/2] . . . 145
4.63.3.6 ingest() [2/2] . . . 145
4.63.3.7 loadFilters() . . . 145
4.63.3.8 passToOnwardFilter() . . . 146
4.63.3.9 processQueue() . . . 146
4.63.3.10 running() . . . 146
4.63.3.11 setResourceLimit() . . . 146
4.63.3.12 setStatistics() . . . 147
4.63.3.13 start() . . . 147
4.63.3.14 unDeprecateAssetTrackingRecord() . . . 147
4.63.3.15 unDeprecateStorageAssetTrackingRecord() . . . 148
4.63.3.16 updateStats() . . . 148
4.63.3.17 useFilteredData() . . . 148
4.64 IngestRate Class Reference . . . 149
4.64.1 Detailed Description . . . 149
4.64.2 Constructor & Destructor Documentation . . . 150
4.64.2.1 IngestRate() . . . 150
4.64.3 Member Function Documentation . . . 150
4.64.3.1 ingest() . . . 150
4.64.3.2 periodic() . . . 150
4.64.3.3 updateConfig() . . . 150
4.65 InsertValue Class Reference . . . 151
4.65.1 Detailed Description . . . 151
4.66 InsertValues Class Reference . . . 152
4.67 Join Class Reference . . . 152
4.67.1 Detailed Description . . . 153
4.67.2 Member Function Documentation . . . 153
4.67.2.1 toJSON() . . . 153
4.68 JSONPath Class Reference . . . 153
4.68.1 Detailed Description . . . 154
4.68.2 Constructor & Destructor Documentation . . . 154
4.68.2.1 ~JSONPath() . . . 154
4.68.3 Member Function Documentation . . . 154
4.68.3.1 findNode() . . . 154
4.69 JSONProperties Class Reference . . . 155

4.69.1 Detailed Description . . . 155
4.70 JSONProperty Class Reference . . . 156
4.71 JSONProvider Class Reference . . . 156
4.72 JSONReading Class Reference . . . 156
4.72.1 Detailed Description . . . 157
4.72.2 Constructor & Destructor Documentation . . . 157
4.72.2.1 JSONReading() . . . 158
4.73 ACL::KeyValueItem Class Reference . . . 158
4.74 LALookup Class Reference . . . 158
4.74.1 Detailed Description . . . 159
4.74.2 Member Function Documentation . . . 159
4.74.2.1 containerSent() [1/2] . . . 159
4.74.2.2 containerSent() [2/2] . . . 160
4.75 LibcurlHttps Class Reference . . . 160
4.75.1 Member Function Documentation . . . 161
4.75.1.1 sendRequest() . . . 161
4.75.1.2 setProxy() . . . 162
4.76 Logger Class Reference . . . 162
4.76.1 Detailed Description . . . 163
4.76.2 Constructor & Destructor Documentation . . . 163
4.76.2.1 Logger() . . . 163
4.76.3 Member Function Documentation . . . 164
4.76.3.1 debug() . . . 164
4.76.3.2 error() . . . 164
4.76.3.3 fatal() . . . 164
4.76.3.4 info() . . . 165
4.76.3.5 printLongString() . . . 165
4.76.3.6 registerInterceptor() . . . 165
4.76.3.7 setMinLevel() . . . 166
4.76.3.8 unregisterInterceptor() . . . 166
4.76.3.9 warn() . . . 166
4.77 ManagementApi Class Reference . . . 167
4.77.1 Detailed Description . . . 169
4.77.2 Constructor & Destructor Documentation . . . 169
4.77.2.1 ManagementApi() . . . 169
4.77.3 Member Function Documentation . . . 169
4.77.3.1 configChange() . . . 169
4.77.3.2 configChildCreate() . . . 170
4.77.3.3 configChildDelete() . . . 170
4.77.3.4 getInstance() . . . 170
4.77.3.5 ping() . . . 170
4.77.3.6 registerProvider() . . . 171

4.77.3.7 registerStats() . . . 171
4.77.3.8 securityChange() . . . 171
4.77.3.9 shutdown() . . . 171
4.78 ManagementClient Class Reference . . . 172
4.78.1 Detailed Description . . . 174
4.78.2 Constructor & Destructor Documentation . . . 174
4.78.2.1 ManagementClient() . . . 174
4.78.3 Member Function Documentation . . . 174
4.78.3.1 addAssetTrackingTuple() . . . 175
4.78.3.2 addAuditEntry() . . . 175
4.78.3.3 addCategory() . . . 175
4.78.3.4 addChildCategories() . . . 176
4.78.3.5 addProxy() . . . 176
4.78.3.6 addStorageAssetTrackingTuple() . . . 177
4.78.3.7 clearAlert() . . . 177
4.78.3.8 deleteProxy() . . . 178
4.78.3.9 getACL() . . . 178
4.78.3.10 getAlertByKey() . . . 178
4.78.3.11 getAssetTrackingTuple() . . . 179
4.78.3.12 getAssetTrackingTuples() . . . 179
4.78.3.13 getCategories() . . . 180
4.78.3.14 getCategory() . . . 180
4.78.3.15 getChildCategories() . . . 180
4.78.3.16 getDeprecatedAssetTrackingTuples() . . . 181
4.78.3.17 getHttpClient() . . . 181
4.78.3.18 getService() . . . 181
4.78.3.19 getServices() . . . 182
4.78.3.20 getStorageAssetTrackingTuple() . . . 182
4.78.3.21 getStorageAssetTrackingTuples() . . . 183
4.78.3.22 raiseAlert() . . . 183
4.78.3.23 refreshBearerToken() . . . 183
4.78.3.24 registerCategory() . . . 184
4.78.3.25 registerCategoryChild() . . . 184
4.78.3.26 registerService() . . . 184
4.78.3.27 restartService() . . . 185
4.78.3.28 setCategoryItemValue() . . . 185
4.78.3.29 unregisterCategory() . . . 186
4.78.3.30 unregisterService() . . . 186
4.78.3.31 validateDatapoints() . . . 186
4.78.3.32 verifyAccessBearerToken() . . . 187
4.78.3.33 verifyBearerToken() . . . 187
4.79 MemConnectionManager Class Reference . . . 187

4.79.1 Detailed Description . . . 188
4.80 NorthApi Class Reference . . . 188
4.80.1 Constructor & Destructor Documentation . . . 189
4.80.1.1 NorthApi() . . . 189
4.80.1.2 ~NorthApi() . . . 189
4.80.2 Member Function Documentation . . . 189
4.80.2.1 attachDebugger() . . . 189
4.80.2.2 detachDebugger() . . . 189
4.80.2.3 getDebuggerBuffer() . . . 190
4.80.2.4 isolateDebugger() . . . 190
4.80.2.5 replayDebugger() . . . 190
4.80.2.6 setDebuggerBuffer() . . . 191
4.80.2.7 startServer() . . . 191
4.80.2.8 stateDebugger() . . . 191
4.80.2.9 stepDebugger() . . . 191
4.80.2.10 suspendDebugger() . . . 192
4.81 NorthFilterPipeline Class Reference . . . 192
4.81.1 Detailed Description . . . 193
4.81.2 Constructor & Destructor Documentation . . . 193
4.81.2.1 NorthFilterPipeline() . . . 193
4.81.3 Member Function Documentation . . . 194
4.81.3.1 setupFiltersPipeline() . . . 194
4.82 NorthPlugin Class Reference . . . 194
4.82.1 Detailed Description . . . 196
4.82.2 Constructor & Destructor Documentation . . . 196
4.82.2.1 NorthPlugin() [1/2] . . . 196
4.82.2.2 NorthPlugin() [2/2] . . . 196
4.82.3 Member Function Documentation . . . 197
4.82.3.1 init() . . . 197
4.82.3.2 send() . . . 197
4.83 NorthPythonPluginHandle Class Reference . . . 198
4.83.1 Detailed Description . . . 199
4.83.2 Constructor & Destructor Documentation . . . 199
4.83.2.1 NorthPythonPluginHandle() . . . 199
4.84 NorthService Class Reference . . . 199
4.84.1 Detailed Description . . . 201
4.84.2 Member Function Documentation . . . 201
4.84.2.1 alertFailures() . . . 202
4.84.2.2 clearFailures() . . . 202
4.84.2.3 debugState() . . . 202
4.85 NorthServiceProvider Class Reference . . . 202
4.85.1 Detailed Description . . . 203

4.86 NoSuchCategory Class Reference . . . 203
4.86.1 Detailed Description . . . 204
4.87 NoSuchCategoryItem Class Reference . . . 204
4.87.1 Detailed Description . . . 205
4.88 NoSuchCategoryItemValue Class Reference . . . 205
4.88.1 Detailed Description . . . 206
4.89 NotificationPythonPluginHandle Class Reference . . . 206
4.89.1 Detailed Description . . . 207
4.89.2 Constructor & Destructor Documentation . . . 207
4.89.2.1 NotificationPythonPluginHandle() . . . 207
4.90 NotSupportedDataType Class Reference . . . 208
4.90.1 Detailed Description . . . 208
4.91 OCS Class Reference . . . 209
4.91.1 Detailed Description . . . 209
4.91.2 Member Function Documentation . . . 209
4.91.2.1 extractToken() . . . 209
4.91.2.2 OCSRetrieveAuthToken() . . . 209
4.91.2.3 retrieveToken() . . . 210
4.92 OMF Class Reference . . . 210
4.92.1 Detailed Description . . . 212
4.92.2 Member Function Documentation . . . 212
4.92.2.1 ApplyPIServerNamingRulesInvalidChars() . . . 212
4.92.2.2 ApplyPIServerNamingRulesObj() . . . 213
4.92.2.3 ApplyPIServerNamingRulesPath() . . . 213
4.92.2.4 compress_string() . . . 214
4.92.2.5 createAssetAFH() . . . 214
4.92.2.6 deleteAssetAFH() . . . 215
4.92.2.7 extractVariable() . . . 215
4.92.2.8 generateAFHierarchyPrefixLevel() . . . 215
4.92.2.9 generateMeasurementId() . . . 216
4.92.2.10 generateSuffixType() . . . 216
4.92.2.11 getFormatType() . . . 216
4.92.2.12 getHashStored() . . . 217
4.92.2.13 getNamingScheme() . . . 217
4.92.2.14 getPathOrigStored() . . . 217
4.92.2.15 getPathStored() . . . 218
4.92.2.16 isDataTypeError() . . . 218
4.92.2.17 reportAsset() . . . 218
4.92.2.18 sendToServer() [1/4] . . . 219
4.92.2.19 sendToServer() [2/4] . . . 219
4.92.2.20 sendToServer() [3/4] . . . 220
4.92.2.21 sendToServer() [4/4] . . . 220

4.92.2.22 setFormatType() . . . 221
4.92.2.23 setMapObjectTypes() . . . 221
4.92.2.24 setPathStored() . . . 221
4.92.2.25 unsetMapObjectTypes() . . . 222
4.92.2.26 variableValueHandle() . . . 222
4.93 OMFAFLocationHint Class Reference . . . 223
4.93.1 Detailed Description . . . 223
4.94 OMFBuffer Class Reference . . . 224
4.94.1 Detailed Description . . . 224
4.94.2 Constructor & Destructor Documentation . . . 224
4.94.2.1 OMFBuffer() . . . 225
4.94.3 Member Function Documentation . . . 225
4.94.3.1 append() [1/7] . . . 225
4.94.3.2 append() [2/7] . . . 225
4.94.3.3 append() [3/7] . . . 225
4.94.3.4 append() [4/7] . . . 226
4.94.3.5 append() [5/7] . . . 226
4.94.3.6 append() [6/7] . . . 226
4.94.3.7 append() [7/7] . . . 226
4.94.3.8 coalesce() . . . 227
4.94.3.9 quote() . . . 227
4.95 OMFData Class Reference . . . 227
4.95.1 Detailed Description . . . 228
4.95.2 Constructor & Destructor Documentation . . . 228
4.95.2.1 OMFData() . . . 228
4.96 OMFDataTypes Class Reference . . . 228
4.96.1 Detailed Description . . . 229
4.97 OMFError Class Reference . . . 229
4.97.1 Detailed Description . . . 230
4.97.2 Member Function Documentation . . . 230
4.97.2.1 getEventExceptionMessage() . . . 230
4.97.2.2 getEventExceptionType() . . . 231
4.97.2.3 getEventReason() . . . 231
4.97.2.4 getEventSeverity() . . . 231
4.97.2.5 getHttpCode() . . . 232
4.97.2.6 getMessage() . . . 232
4.97.2.7 Log() . . . 232
4.97.2.8 setFromHttpResponse() . . . 233
4.98 OMFHint Class Reference . . . 233
4.98.1 Detailed Description . . . 235
4.99 OMFHints Class Reference . . . 235
4.99.1 Detailed Description . . . 235

4.99.2 Constructor & Destructor Documentation . . . 235
4.99.2.1 OMFHints() . . . 235
4.99.3 Member Function Documentation . . . 236
4.99.3.1 getHintForChecksum() . . . 236
4.100 OMFInformation Class Reference . . . 236
4.100.1 Detailed Description . . . 237
4.100.2 Constructor & Destructor Documentation . . . 237
4.100.2.1 OMFInformation() . . . 237
4.100.3 Member Function Documentation . . . 237
4.100.3.1 saveData() . . . 237
4.100.3.2 send() . . . 237
4.100.3.3 start() . . . 238
4.101 OMFIntegerHint Class Reference . . . 238
4.101.1 Detailed Description . . . 239
4.102 OMFInterpolationHint Class Reference . . . 239
4.102.1 Detailed Description . . . 240
4.103 OMFLegacyTypeHint Class Reference . . . 240
4.103.1 Detailed Description . . . 241
4.104 OMFLinkedData Class Reference . . . 241
4.104.1 Detailed Description . . . 242
4.104.2 Member Function Documentation . . . 242
4.104.2.1 buildLookup() . . . 242
4.104.2.2 clearLALookup() . . . 242
4.104.2.3 flushContainers() . . . 243
4.104.2.4 processReading() . . . 243
4.105 OMFMaximumHint Class Reference . . . 244
4.105.1 Detailed Description . . . 245
4.106 OMFMinimumHint Class Reference . . . 245
4.106.1 Detailed Description . . . 246
4.107 OMFNumberHint Class Reference . . . 246
4.107.1 Detailed Description . . . 247
4.108 OMFSourceHint Class Reference . . . 247
4.108.1 Detailed Description . . . 248
4.109 OMFTagHint Class Reference . . . 248
4.109.1 Detailed Description . . . 249
4.110 OMFTagNameDatapointHint Class Reference . . . 249
4.110.1 Detailed Description . . . 250
4.111 OMFTagNameHint Class Reference . . . 250
4.111.1 Detailed Description . . . 251
4.112 OMFTypeNameHint Class Reference . . . 251
4.112.1 Detailed Description . . . 252
4.113 OMFUOMHint Class Reference . . . 252

4.113.1 Detailed Description . . . 253
4.114 PerfMon Class Reference . . . 253
4.114.1 Constructor & Destructor Documentation . . . 253
4.114.1.1 PerfMon() . . . 253
4.114.2 Member Function Documentation . . . 253
4.114.2.1 addValue() . . . 253
4.114.2.2 getValues() . . . 255
4.115 PerformanceMonitor Class Reference . . . 255
4.115.1 Detailed Description . . . 256
4.115.2 Constructor & Destructor Documentation . . . 256
4.115.2.1 PerformanceMonitor() . . . 256
4.115.3 Member Function Documentation . . . 256
4.115.3.1 collect() . . . 256
4.115.3.2 setCollecting() . . . 256
4.116 PipelineBranch Class Reference . . . 257
4.116.1 Detailed Description . . . 258
4.116.2 Constructor & Destructor Documentation . . . 258
4.116.2.1 ∼PipelineBranch() . . . 258
4.116.3 Member Function Documentation . . . 258
4.116.3.1 ingest() . . . 258
4.116.3.2 init() . . . 259
4.116.3.3 setup() . . . 259
4.116.3.4 setupConfiguration() . . . 260
4.116.3.5 shutdown() . . . 260
4.117 PipelineDebugger Class Reference . . . 260
4.117.1 Detailed Description . . . 261
4.117.2 Member Function Documentation . . . 261
4.117.2.1 fetchBuffer() . . . 261
4.117.2.2 process() . . . 261
4.117.2.3 setBuffer() . . . 262
4.118 PipelineElement Class Reference . . . 262
4.118.1 Detailed Description . . . 264
4.118.2 Member Function Documentation . . . 264
4.118.2.1 attachDebugger() . . . 264
4.118.2.2 getDebuggerBuffer() . . . 264
4.118.2.3 setDebuggerBuffer() . . . 264
4.119 PipelineFilter Class Reference . . . 265
4.119.1 Detailed Description . . . 266
4.119.2 Constructor & Destructor Documentation . . . 266
4.119.2.1 PipelineFilter() . . . 266
4.119.3 Member Function Documentation . . . 266
4.119.3.1 init() . . . 266

4.119.3.2 setup() . . . 267
4.119.3.3 setupConfiguration() . . . 267
4.119.3.4 shutdown() . . . 267
4.120 PipelineWriter Class Reference . . . 268
4.120.1 Detailed Description . . . 269
4.121 PIWebAPI Class Reference . . . 270
4.121.1 Detailed Description . . . 270
4.121.2 Member Function Documentation . . . 270
4.121.2.1 errorMessageHandler() . . . 270
4.121.2.2 ExtractVersion() . . . 271
4.121.2.3 GetVersion() . . . 271
4.122 Plugin Class Reference . . . 272
4.122.1 Detailed Description . . . 273
4.123 PLUGIN_ERROR Struct Reference . . . 273
4.123.1 Detailed Description . . . 273
4.124 PLUGIN_INFORMATION Struct Reference . . . 273
4.124.1 Detailed Description . . . 274
4.125 plugin_parameter Struct Reference . . . 274
4.125.1 Detailed Description . . . 274
4.126 PluginData Class Reference . . . 274
4.126.1 Constructor & Destructor Documentation . . . 274
4.126.1.1 PluginData() . . . 274
4.126.2 Member Function Documentation . . . 275
4.126.2.1 loadStoredData() . . . 275
4.126.2.2 persistPluginData() . . . 275
4.127 PluginHandle Class Reference . . . 276
4.127.1 Detailed Description . . . 276
4.128 PluginManager Class Reference . . . 276
4.128.1 Detailed Description . . . 277
4.128.2 Member Function Documentation . . . 277
4.128.2.1 getInstalledPlugins() . . . 277
4.128.2.2 getPluginsByFlags() . . . 278
4.129 PluginNotImplementedException Class Reference . . . 278
4.129.1 Detailed Description . . . 279
4.130 ProfileItem Class Reference . . . 279
4.131 PurgeConfiguration Class Reference . . . 280
4.131.1 Member Function Documentation . . . 280
4.131.1.1 exclude() . . . 280
4.131.1.2 getInstance() . . . 280
4.131.1.3 isExcluded() . . . 281
4.131.1.4 minimumRetained() . . . 281
4.132 PurgeResult Class Reference . . . 281

4.133 PurgeSystem Class Reference . . . 282
4.134 PythonConfigCategory Class Reference . . . 283
4.134.1 Detailed Description . . . 283
4.134.2 Constructor & Destructor Documentation . . . 284
4.134.2.1 PythonConfigCategory() . . . 284
4.134.3 Member Function Documentation . . . 284
4.134.3.1 toPython() . . . 284
4.135 PythonModule Class Reference . . . 284
4.135.1 Detailed Description . . . 285
4.136 PythonPluginHandle Class Reference . . . 285
4.136.1 Detailed Description . . . 286
4.136.2 Constructor & Destructor Documentation . . . 286
4.136.2.1 ~PythonPluginHandle() . . . 286
4.136.3 Member Function Documentation . . . 287
4.136.3.1 ResolveSymbol() . . . 287
4.137 PythonReading Class Reference . . . 287
4.137.1 Detailed Description . . . 288
4.137.2 Constructor & Destructor Documentation . . . 288
4.137.2.1 PythonReading() . . . 288
4.137.3 Member Function Documentation . . . 289
4.137.3.1 errorMessage() . . . 289
4.137.3.2 isArray() . . . 289
4.137.3.3 toPython() . . . 289
4.138 PythonReadingSet Class Reference . . . 290
4.138.1 Detailed Description . . . 291
4.138.2 Constructor & Destructor Documentation . . . 291
4.138.2.1 PythonReadingSet() . . . 291
4.138.3 Member Function Documentation . . . 291
4.138.3.1 toPython() . . . 291
4.139 PythonRuntime Class Reference . . . 292
4.139.1 Member Function Documentation . . . 292
4.139.1.1 execute() . . . 292
4.139.1.2 getPythonRuntime() . . . 292
4.139.1.3 importModule() . . . 293
4.140 Query Class Reference . . . 293
4.140.1 Detailed Description . . . 294
4.140.2 Constructor & Destructor Documentation . . . 294
4.140.2.1 Query() [1/8] . . . 294
4.140.2.2 Query() [2/8] . . . 294
4.140.2.3 Query() [3/8] . . . 295
4.140.2.4 Query() [4/8] . . . 295
4.140.2.5 Query() [5/8] . . . 295

4.140.2.6 Query() [6/8] . . . 296
4.140.2.7 Query() [7/8] . . . 296
4.140.2.8 Query() [8/8] . . . 296
4.140.3 Member Function Documentation . . . 296
4.140.3.1 aggregate() . . . 297
4.140.3.2 group() . . . 297
4.140.3.3 join() . . . 297
4.140.3.4 limit() . . . 297
4.140.3.5 returns() [1/2] . . . 298
4.140.3.6 returns() [2/2] . . . 298
4.140.3.7 sort() . . . 298
4.140.3.8 timebucket() . . . 299
4.141 QueryProfile Class Reference . . . 299
4.142 RDSAcknowledge Struct Reference . . . 299
4.143 RDSBlockHeader Struct Reference . . . 300
4.144 RDSConnectHeader Struct Reference . . . 300
4.145 RDSReadingHeader Struct Reference . . . 300
4.146 Reading Class Reference . . . 301
4.146.1 Detailed Description . . . 303
4.146.2 Member Function Documentation . . . 303
4.146.2.1 escape() . . . 303
4.146.2.2 getAssetDateTime() . . . 303
4.146.2.3 getAssetDateUserTime() . . . 304
4.146.2.4 getDatapoint() . . . 304
4.146.2.5 getFormattedDateTimeStr() . . . 304
4.146.2.6 JSONtoDatapoints() . . . 305
4.146.2.7 removeDatapoint() . . . 305
4.146.2.8 stringToTimestamp() . . . 305
4.146.2.9 substitute() . . . 306
4.146.3 Member Data Documentation . . . 306
4.146.3.1 m_dateTypes . . . 306
4.147 ReadingCircularBuffer Class Reference . . . 306
4.147.1 Detailed Description . . . 307
4.147.2 Constructor & Destructor Documentation . . . 307
4.147.2.1 ReadingCircularBuffer() . . . 307
4.147.3 Member Function Documentation . . . 307
4.147.3.1 extract() . . . 307
4.147.3.2 insert() [1/3] . . . 308
4.147.3.3 insert() [2/3] . . . 308
4.147.3.4 insert() [3/3] . . . 308
4.148 ReadingsCatalogue::ReadingReference Struct Reference . . . 309
4.149 ReadingsCatalogue Class Reference . . . 309

4.149.1 Detailed Description . . . 311
4.149.2 Member Function Documentation . . . 312
4.149.2.1 attachDb() . . . 312
4.149.2.2 attachDbsToAllConnections() . . . 313
4.149.2.3 connectionAttachAllDbs() . . . 313
4.149.2.4 connectionAttachDbList() . . . 313
4.149.2.5 createReadingsOverflowTable() . . . 314
4.149.2.6 detachDb() . . . 314
4.149.2.7 evaluateGlobalId() . . . 314
4.149.2.8 extractDbIdFromName() . . . 315
4.149.2.9 extractReadingsIdFromName() . . . 315
4.149.2.10 generateDbAlias() . . . 315
4.149.2.11 generateDbFileName() . . . 316
4.149.2.12 generateDbName() . . . 316
4.149.2.13 generateDbNameFromTableId() . . . 317
4.149.2.14 generateReadingsName() . . . 317
4.149.2.15 getAllDbs() . . . 317
4.149.2.16 getEmptyReadingTableReference() . . . 318
4.149.2.17 getMaxReadingsId() . . . 318
4.149.2.18 getMinGlobalId() . . . 318
4.149.2.19 getNewDbs() . . . 319
4.149.2.20 getReadingPosition() . . . 319
4.149.2.21 getReadingReference() . . . 319
4.149.2.22 getReadingsCount() . . . 320
4.149.2.23 latestDbUpdate() . . . 320
4.149.2.24 loadAssetReadingCatalogue() . . . 320
4.149.2.25 loadEmptyAssetReadingCatalogue() . . . 321
4.149.2.26 multipleReadingsInit() . . . 321
4.149.2.27 preallocateNewDbsRange() . . . 321
4.149.2.28 preallocateReadingsTables() . . . 321
4.149.2.29 purgeAllReadings() . . . 322
4.149.2.30 setUsedDbId() . . . 322
4.149.2.31 sqlConstructMultiDb() . . . 322
4.149.2.32 sqlConstructOverflow() . . . 323
4.149.2.33 SQLExec() . . . 323
4.149.2.34 storeGlobalId() . . . 324
4.150 ReadingSet Class Reference . . . 324
4.150.1 Detailed Description . . . 325
4.150.2 Constructor & Destructor Documentation . . . 325
4.150.2.1 ReadingSet() [1/2] . . . 326
4.150.2.2 ReadingSet() [2/2] . . . 326
4.150.3 Member Function Documentation . . . 326

4.150.3.1 append() [1/3] . . . 326
4.150.3.2 append() [2/3] . . . 327
4.150.3.3 append() [3/3] . . . 327
4.150.3.4 copy() . . . 327
4.150.3.5 getReadingId() . . . 328
4.150.3.6 merge() . . . 328
4.151 ReadingSetCircularBuffer Class Reference . . . 328
4.151.1 Detailed Description . . . 329
4.151.2 Constructor & Destructor Documentation . . . 329
4.151.2.1 ReadingSetCircularBuffer() . . . 329
4.151.3 Member Function Documentation . . . 329
4.151.3.1 extract() . . . 329
4.151.3.2 insert() [1/2] . . . 329
4.151.3.3 insert() [2/2] . . . 331
4.152 ReadingSetException Class Reference . . . 331
4.153 ReadingStream Struct Reference . . . 332
4.154 ResultException Class Reference . . . 333
4.155 ResultIncorrectTypeException Class Reference . . . 334
4.156 ResultNoMoreRowsException Class Reference . . . 335
4.157 ResultNoSuchColumnException Class Reference . . . 336
4.158 ResultSet Class Reference . . . 336
4.158.1 Detailed Description . . . 337
4.158.2 Constructor & Destructor Documentation . . . 337
4.158.2.1 ResultSet() . . . 337
4.158.3 Member Function Documentation . . . 338
4.158.3.1 columnName() . . . 338
4.158.3.2 columnType() . . . 338
4.158.3.3 findColumn() . . . 339
4.158.3.4 firstRow() . . . 339
4.158.3.5 hasNextRow() . . . 339
4.158.3.6 isLastRow() . . . 340
4.158.3.7 nextRow() . . . 340
4.159 Returns Class Reference . . . 341
4.159.1 Detailed Description . . . 341
4.160 ResultSet::Row Class Reference . . . 341
4.160.1 Member Function Documentation . . . 341
4.160.1.1 getColumn() . . . 341
4.160.1.2 getType() . . . 342
4.161 Schema Class Reference . . . 342
4.161.1 Detailed Description . . . 343
4.161.2 Constructor & Destructor Documentation . . . 343
4.161.2.1 Schema() . . . 343

4.161.3 Member Function Documentation . . . 343
4.161.3.1 attach() . . . 344
4.161.3.2 upgrade() . . . 344
4.162 SchemaManager Class Reference . . . 344
4.162.1 Detailed Description . . . 345
4.162.2 Member Function Documentation . . . 345
4.162.2.1 create() . . . 345
4.162.2.2 exists() . . . 346
4.162.2.3 getInstance() . . . 346
4.162.2.4 load() . . . 346
4.163 SendingProcess Class Reference . . . 347
4.163.1 Constructor & Destructor Documentation . . . 349
4.163.1.1 SendingProcess() . . . 349
4.163.2 Member Function Documentation . . . 349
4.163.2.1 createNewStream() . . . 349
4.163.2.2 createStream() . . . 349
4.163.2.3 getLastSentReadingId() . . . 350
4.163.2.4 getLoadBufferIndex() . . . 350
4.163.2.5 getLoadBufferIndexPtr() . . . 350
4.163.2.6 passToOnwardFilter() . . . 350
4.163.2.7 setLoadBufferIndex() . . . 351
4.163.2.8 useFilteredData() . . . 351
4.164 ServiceAuthHandler Class Reference . . . 351
4.164.1 Detailed Description . . . 353
4.164.2 Member Function Documentation . . . 353
4.164.2.1 AuthenticationMiddlewareACL() . . . 353
4.164.2.2 AuthenticationMiddlewareCommon() . . . 354
4.164.2.3 AuthenticationMiddlewarePUT() . . . 354
4.164.2.4 createSecurityCategories() . . . 355
4.164.2.5 getAuthenticatedCaller() . . . 355
4.164.2.6 refreshBearerToken() . . . 356
4.164.2.7 securityChange() . . . 356
4.164.2.8 setAuthenticatedCaller() . . . 356
4.164.2.9 updateSecurityCategory() . . . 356
4.165 ServiceHandler Class Reference . . . 357
4.165.1 Detailed Description . . . 357
4.166 ServiceRecord Class Reference . . . 358
4.167 ServiceRegistry Class Reference . . . 359
4.167.1 Detailed Description . . . 359
4.167.2 Member Function Documentation . . . 359
4.167.2.1 findService() . . . 359
4.167.2.2 getUUID() . . . 360

4.167.2.3 registerService() . . . 360
4.167.2.4 unRegisterService() . . . 361
4.168 SimpleHttp Class Reference . . . 361
4.168.1 Member Function Documentation . . . 362
4.168.1.1 sendRequest() . . . 362
4.168.1.2 setProxy() . . . 363
4.169 SimpleHttps Class Reference . . . 363
4.169.1 Member Function Documentation . . . 364
4.169.1.1 sendRequest() . . . 365
4.169.1.2 setProxy() . . . 365
4.170 Sort Class Reference . . . 366
4.170.1 Detailed Description . . . 366
4.171 SouthApi Class Reference . . . 366
4.171.1 Constructor & Destructor Documentation . . . 367
4.171.1.1 SouthApi() . . . 367
4.171.1.2 ∼SouthApi() . . . 367
4.171.2 Member Function Documentation . . . 367
4.171.2.1 attachDebugger() . . . 367
4.171.2.2 detachDebugger() . . . 368
4.171.2.3 getDebuggerBuffer() . . . 368
4.171.2.4 isolateDebugger() . . . 368
4.171.2.5 operation() . . . 369
4.171.2.6 replayDebugger() . . . 369
4.171.2.7 setDebuggerBuffer() . . . 369
4.171.2.8 setPoint() . . . 370
4.171.2.9 startServer() . . . 370
4.171.2.10 stateDebugger() . . . 370
4.171.2.11 stepDebugger() . . . 370
4.171.2.12 suspendDebugger() . . . 371
4.172 SouthPlugin Class Reference . . . 371
4.172.1 Detailed Description . . . 373
4.172.2 Constructor & Destructor Documentation . . . 373
4.172.2.1 SouthPlugin() . . . 373
4.172.3 Member Function Documentation . . . 373
4.172.3.1 operation() . . . 373
4.172.3.2 write() . . . 374
4.173 SouthPythonPluginHandle Class Reference . . . 374
4.173.1 Detailed Description . . . 375
4.173.2 Constructor & Destructor Documentation . . . 375
4.173.2.1 SouthPythonPluginHandle() . . . 375
4.174 SouthService Class Reference . . . 376
4.174.1 Detailed Description . . . 378

4.174.2 Member Function Documentation . . . 378
4.174.2.1 configChange() . . . 378
4.174.2.2 debugState() . . . 378
4.174.2.3 operation() . . . 379
4.174.2.4 processConfigChange() . . . 379
4.174.2.5 setPoint() . . . 379
4.175 SouthServiceProvider Class Reference . . . 380
4.175.1 Detailed Description . . . 381
4.176 SQLBuffer Class Reference . . . 381
4.176.1 Detailed Description . . . 382
4.176.2 Constructor & Destructor Documentation . . . 382
4.176.2.1 SQLBuffer() . . . 382
4.176.3 Member Function Documentation . . . 382
4.176.3.1 append() [1/7] . . . 382
4.176.3.2 append() [2/7] . . . 382
4.176.3.3 append() [3/7] . . . 382
4.176.3.4 append() [4/7] . . . 383
4.176.3.5 append() [5/7] . . . 383
4.176.3.6 append() [6/7] . . . 383
4.176.3.7 append() [7/7] . . . 384
4.176.3.8 coalesce() . . . 384
4.176.3.9 quote() . . . 384
4.177 StatsHistory Class Reference . . . 384
4.177.1 Detailed Description . . . 385
4.178 STORAGE_CONFIGURATION Struct Reference . . . 386
4.178.1 Detailed Description . . . 386
4.179 StorageApi Class Reference . . . 386
4.179.1 Detailed Description . . . 389
4.179.2 Constructor & Destructor Documentation . . . 389
4.179.2.1 ~StorageApi() . . . 389
4.179.3 Member Function Documentation . . . 389
4.179.3.1 commonDelete() . . . 389
4.179.3.2 commonInsert() . . . 389
4.179.3.3 commonQuery() . . . 390
4.179.3.4 commonSimpleQuery() . . . 390
4.179.3.5 commonUpdate() . . . 390
4.179.3.6 createStorageSchema() . . . 391
4.179.3.7 createStorageStream() . . . 391
4.179.3.8 queue() . . . 391
4.179.3.9 readingAppend() . . . 392
4.179.3.10 readingFetch() . . . 392
4.179.3.11 readingPurge() . . . 392

4.179.3.12 readingQuery() . . . 393
4.179.3.13 readingStream() . . . 393
4.179.3.14 storageTableDelete() . . . 393
4.179.3.15 storageTableInsert() . . . 394
4.179.3.16 storageTableQuery() . . . 394
4.179.3.17 storageTableSimpleQuery() . . . 394
4.179.3.18 storageTableUpdate() . . . 395
4.180 StorageAssetTrackingTuple Class Reference . . . 395
4.180.1 Member Function Documentation . . . 396
4.180.1.1 processData() . . . 396
4.181 StorageAssetTrackingTuplePtrEqual Struct Reference . . . 397
4.182 StorageClient Class Reference . . . 397
4.182.1 Detailed Description . . . 399
4.182.2 Member Function Documentation . . . 399
4.182.2.1 deleteTable() [1/2] . . . 399
4.182.2.2 deleteTable() [2/2] . . . 400
4.182.2.3 queryTable() [1/2] . . . 400
4.182.2.4 queryTable() [2/2] . . . 400
4.182.2.5 queryTableToReadings() . . . 401
4.182.2.6 readingAppend() . . . 401
4.182.2.7 readingFetch() . . . 401
4.182.2.8 readingPurgeByAge() . . . 402
4.182.2.9 readingPurgeByAsset() . . . 402
4.182.2.10 readingPurgeBySize() . . . 403
4.182.2.11 readingQuery() . . . 403
4.182.2.12 readingQueryToReadings() . . . 403
4.182.2.13 registerAssetNotification() . . . 404
4.182.2.14 registerTableNotification() . . . 404
4.182.2.15 unregisterAssetNotification() . . . 405
4.182.2.16 unregisterTableNotification() . . . 405
4.183 StorageConfiguration Class Reference . . . 406
4.183.1 Detailed Description . . . 406
4.183.2 Constructor & Destructor Documentation . . . 406
4.183.2.1 StorageConfiguration() . . . 406
4.183.3 Member Function Documentation . . . 406
4.183.3.1 getDefaultCategory() . . . 407
4.184 StorageOperation Class Reference . . . 407
4.184.1 Detailed Description . . . 408
4.185 StoragePerformanceMonitor Class Reference . . . 408
4.185.1 Detailed Description . . . 409
4.186 StoragePlugin Class Reference . . . 409
4.186.1 Detailed Description . . . 411

4.186.2 Constructor & Destructor Documentation . . . 411
4.186.2.1 StoragePlugin() . . . 411
4.187 StoragePluginConfiguration Class Reference . . . 412
4.187.1 Detailed Description . . . 412
4.187.2 Member Function Documentation . . . 412
4.187.2.1 getConfiguration() . . . 412
4.187.2.2 getDefaultCategory() . . . 413
4.188 StorageRegistry Class Reference . . . 413
4.188.1 Detailed Description . . . 414
4.188.2 Constructor & Destructor Documentation . . . 414
4.188.2.1 StorageRegistry() . . . 414
4.188.3 Member Function Documentation . . . 414
4.188.3.1 process() . . . 414
4.188.3.2 processTableDelete() . . . 414
4.188.3.3 processTableInsert() . . . 416
4.188.3.4 processTableUpdate() . . . 416
4.188.3.5 registerAsset() . . . 416
4.188.3.6 registerTable() . . . 417
4.188.3.7 unregisterAsset() . . . 417
4.188.3.8 unregisterTable() . . . 417
4.189 StorageService Class Reference . . . 418
4.189.1 Detailed Description . . . 419
4.190 StorageStats Class Reference . . . 419
4.191 StreamHandler Class Reference . . . 420
4.191.1 Constructor & Destructor Documentation . . . 421
4.191.1.1 ∼StreamHandler() . . . 421
4.191.2 Member Function Documentation . . . 421
4.191.2.1 createStream() . . . 421
4.191.2.2 handler() . . . 421
4.192 TableReference Class Reference . . . 421
4.192.1 Detailed Description . . . 422
4.193 TableRegistration Struct Reference . . . 422
4.194 Timebucket Class Reference . . . 422
4.194.1 Detailed Description . . . 422
4.195 TrackingTuple Class Reference . . . 423
4.195.1 Detailed Description . . . 423
4.196 TransactionBoundary Class Reference . . . 423
4.196.1 Detailed Description . . . 424
4.196.2 Member Function Documentation . . . 424
4.196.2.1 ClearThreadTransaction() . . . 424
4.196.2.2 GetMinReadingId() . . . 424
4.196.2.3 SetThreadTransactionStart() . . . 424

4.197 Unauthorized Class Reference . . . . . 425
4.197.1 Detailed Description . . . . . 426
4.198 UpdateModifier Class Reference . . . . . 426
4.198.1 Detailed Description . . . . . 426
4.199 UpdateValue Class Reference . . . . . 426
4.199.1 Detailed Description . . . . . 427
4.200 ACL::UrlItem Class Reference . . . . . 427
4.201 Where Class Reference . . . . . 427
4.201.1 Detailed Description . . . . . 427

**5 Example Documentation** **429**
5.1 /home/ubuntu/fledge/C/common/include/utils.h . . . . . 429

**Index** **431**

# Chapter 1

# Contributing to Fledge

The project welcomes contributions of all types; documentation, code changes, new plugins, scripts, just simply reports of the way you use Fledge or suggestions of features you would like to see within Fledge.

The following is a set of guidelines for contributing to Fledge IoT project and its plugins, which are hosted in the `fledge-iot Organization` on GitHub.

To give us feedback or make suggestions use the fledge or fledge-help Slack Channel on `LFEdge`.

If you find a security vulnerability within Fledge or any of its plugins then we request that **you inform us via email rather than by opening an issue in GitHub**. This allows us to act on it without giving information that others might exploit. Any security vulnerability will be discussed at the project TSC and user will be informed of the need to upgrade via the Fledge Slack channel. The email address to which vulnerabilities should be reported is `security@dianomic.com`.

## Pull requests

**Please ask first** before embarking on any significant work (e.g. implementing new features, refactoring code etc.), otherwise you risk spending a lot of time working on something that might already be underway or is unlikely to be merged into the project.

Join the fledge or fledge-help Slack channel on `LFEdge`. This will allow you to talk to the wider fledge community and discuss your proposed changes and get help from the maintainers when needed.

Please adhere to the coding conventions used throughout the project and limit your changes to functional rather than aesthetic changes to make it as easy as possible to review your changes. We also encourage you to comment your code changes for the same reason and for the benefit of those that come after you.

Adhering to the following process is the best way to get your work included in the project:

1. `Fork` the project, clone your fork, and configure the remotes:

```
# Clone your fork of the repo into the current directory
git clone https://github.com/<your-username>/fledge.git
# Navigate to the newly cloned directory
cd fledge
# Assign the original repo to a remote called "upstream"
git remote add upstream https://github.com/fledge-iot/fledge.git
```

1. If you cloned a while ago, get the latest changes of develop branch from upstream:

```
git checkout develop
git pull --rebase upstream develop
```

1. Create a new topic branch from `develop`, if you are working a particular issue from the Project Jira then the convention for branch names is to use the Jira name, otherwise choose a descriptive branch name that contains your GitHub username in order to help us track the changes.

```
git checkout -b [topic-branch-name] upstream/develop
```

1. Commit your changes in logical chunks. When you are ready to commit, make sure to write a Good Commit Message™. Use `interactive rebase` to group your commits into logical units of working before making them public.

   Note that every commit you make must be signed. By signing off your work you indicate that you are accepting the `Developer Certificate of Origin`.

   Use your real name (sorry, no pseudonyms or anonymous contributions). If you set your `user.name` and `user.email` git configs, you can sign your commit automatically with `git commit -s`.

2. Locally merge (or rebase) the upstream development branch into your topic branch:

```
git pull --rebase upstream develop
```

1. Push your topic branch up to your fork:

```
git push -u origin [topic-branch-name]
```

1. `Open a Pull Request` with a clear title and detailed description. Also make sure always raise Pull request against develop base branch of upstream only. It must have at least one reviewer to expedite the review and also verify GitHub status checks which let you know if your commits meet the conditions set for the repository you're contributing to. GitHub Status checks are based on external processes, such as continuous integration builds, which run for each push you make to a repository. You can see the pending, passing, or failing state of status checks next to individual commits in your pull request.

## Plugins

The above addresses the main Fledge repository, however plugins each have a repository of their own which contains the code for the plugin and the documentation for the plugin. If you wish to work on an existing plugin then the process is similar to that above, just replace the "fledge.git" repository with the fledge-{plugin-type}-{plugin-name}.git repository, for example

```
# Clone your fork of the repo into the current directory
git clone https://github.com/<your-username>/fledge-south-sinusoid.git
# Navigate to the newly cloned directory
cd fledge-south-sinusoid
# Assign the original repo to a remote called "upstream"
git remote add upstream https://github.com/fledge-iot/fledge-south-sinusoid.git
```

Repeat further steps which we mentioned `here`

If you wish to create a new plugin then contact the maintainers and we will create a blank base repository for you to add your code into.

# Chapter 2

# Hierarchical Index

## 2.1 Class Hierarchy

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


ACL . . . 15
ACL::ACLReason . . . 17
Aggregate . . . 19
AssetTracker . . . 20
AssetTrackingTable . . . 26
AssetTrackingTuplePtrEqual . . . 29
AttachDbSync . . . 29
AuditLogger . . . 30
BearerToken . . . 35
ConfigCategory::CategoryItem . . . 38
ResultSet::ColumnValue . . . 42
ConfigCategories . . . 44
ConfigCategory . . . 45
    ConfigCategoryChange . . . 65
    DefaultConfigCategory . . . 114
    PythonConfigCategory . . . 283
ConfigCategoryDescription . . . 67
ConfigHandler . . . 69
ConfigurationManager . . . 75
Connection . . . 85
ConnectionManager . . . 92
DataBuffer . . . 100
    Base64DataBuffer . . . 32
Datapoint . . . 107
DatapointValue . . . 109
DataSender . . . 112
DiskSpaceMonitor . . . 115
DPImage . . . 117
    Base64DPImage . . . 34
exception
    ACLMalformed . . . 16
    ACLReasonMalformed . . . 18
    AllCategoriesEx . . . 19
    BadRequest . . . 31

CategoryDetailsEx . . . 37
ChildCategoriesEx . . . 41
ConfigCategoryDefaultWithValue . . . 66
ConfigCategoryEx . . . 68
ConfigItemAttributeNotFound . . . 72
ConfigItemNotAList . . . 73
ConfigItemNotFound . . . 74
ConfigMalformed . . . 75
ConfigValueFoundWithDefault . . . 83
Conflict . . . 84
ExistingChildCategories . . . 119
NoSuchCategory . . . 203
NoSuchCategoryItem . . . 204
NoSuchCategoryItemValue . . . 205
NotSupportedDataType . . . 208
PluginNotImplementedException . . . 278
ReadingSetException . . . 331
ResultException . . . 333
ResultIncorrectTypeException . . . 334
ResultNoMoreRowsException . . . 335
ResultNoSuchColumnException . . . 336
StorageOperation . . . 407
Unauthorized . . . 425
Expression . . . 120
FormData::FieldValue . . . 121
FilterPipeline . . . 122
NorthFilterPipeline . . . 192
FledgeFilter . . . 132
FledgeProcess . . . 134
CheckUpdates . . . 40
PurgeSystem . . . 282
SendingProcess . . . 347
StatsHistory . . . 384
FormData . . . 136
std::hash< AssetTrackingTuple * > . . . 137
std::hash< AssetTrackingTuple > . . . 138
std::hash< StorageAssetTrackingTuple * > . . . 138
std::hash< StorageAssetTrackingTuple > . . . 138
HttpSender . . . 138
LibcurlHttps . . . 160
SimpleHttp . . . 361
SimpleHttps . . . 363
IngestRate . . . 149
InsertValue . . . 151
Join . . . 152
JSONPath . . . 153
JSONProperty . . . 156
JSONProvider . . . 156
NorthServiceProvider . . . 202
ServiceRecord . . . 358
SouthServiceProvider . . . 380
StorageStats . . . 419
ACL::KeyValueItem . . . 158
LALookup . . . 158
Logger . . . 162
ManagementApi . . . 167
CoreManagementApi . . . 97

ManagementClient . . . 172
MemConnectionManager . . . 187
NorthApi . . . 188
OCS . . . 209
OMF . . . 210
OMFBuffer . . . 224
OMFData . . . 227
OMFDataTypes . . . 228
OMFError . . . 229
OMFHint . . . 233
- OMFAFLocationHint . . . 223
- OMFIntegerHint . . . 238
- OMFInterpolationHint . . . 239
- OMFLegacyTypeHint . . . 240
- OMFMaximumHint . . . 244
- OMFMinimumHint . . . 245
- OMFNumberHint . . . 246
- OMFSourceHint . . . 247
- OMFTagHint . . . 248
- OMFTagNameDatapointHint . . . 249
- OMFTagNameHint . . . 250
- OMFTypeNameHint . . . 251
- OMFUOMHint . . . 252

OMFHints . . . 235
OMFInformation . . . 236
OMFLinkedData . . . 241
PerfMon . . . 253
PerformanceMonitor . . . 255
- StoragePerformanceMonitor . . . 408

PipelineDebugger . . . 260
PipelineElement . . . 262
- PipelineBranch . . . 257
- PipelineFilter . . . 265
- PipelineWriter . . . 268

PIWebAPI . . . 270
Plugin . . . 272
- FilterPlugin . . . 127
- NorthPlugin . . . 194
- SouthPlugin . . . 371
- StoragePlugin . . . 409

PLUGIN_ERROR . . . 273
PLUGIN_INFORMATION . . . 273
plugin_parameter . . . 274
PluginData . . . 274
PluginHandle . . . 276
- BinaryPluginHandle . . . 36
- PythonPluginHandle . . . 285
  - FilterPythonPluginHandle . . . 130
  - NorthPythonPluginHandle . . . 198
  - NotificationPythonPluginHandle . . . 206
  - SouthPythonPluginHandle . . . 374

PluginManager . . . 276
ProfileItem . . . 279
PurgeConfiguration . . . 280
PurgeResult . . . 281
PythonModule . . . 284
PythonRuntime . . . 292

Query . . . 293
QueryProfile . . . 299
RDSAcknowledge . . . 299
RDSBlockHeader . . . 300
RDSConnectHeader . . . 300
RDSReadingHeader . . . 300
Reading . . . 301
  JSONReading . . . 156
  PythonReading . . . 287
ReadingCircularBuffer . . . 306
ReadingsCatalogue::ReadingReference . . . 309
ReadingsCatalogue . . . 309
ReadingSet . . . 324
  PythonReadingSet . . . 290
ReadingSetCircularBuffer . . . 328
ReadingStream . . . 332
ResultSet . . . 336
Returns . . . 341
ResultSet::Row . . . 341
Schema . . . 342
SchemaManager . . . 344
ServiceHandler . . . 357
  DataLoad . . . 102
  Ingest . . . 140
  ServiceAuthHandler . . . 351
    NorthService . . . 199
    SouthService . . . 376
  StorageService . . . 418
ServiceRegistry . . . 359
Sort . . . 366
SouthApi . . . 366
SQLBuffer . . . 381
STORAGE_CONFIGURATION . . . 386
StorageApi . . . 386
StorageAssetTrackingTuplePtrEqual . . . 397
StorageClient . . . 397
StorageConfiguration . . . 406
StoragePluginConfiguration . . . 412
StorageRegistry . . . 413
StreamHandler . . . 420
TableReference . . . 421
TableRegistration . . . 422
Timebucket . . . 422
TrackingTuple . . . 423
  AssetTrackingTuple . . . 27
  StorageAssetTrackingTuple . . . 395
TransactionBoundary . . . 423
UpdateModifier . . . 426
UpdateValue . . . 426
ACL::UrlItem . . . 427
vector
  ExpressionValues . . . 121
  InsertValues . . . 152
  JSONProperties . . . 155
Where . . . 427

# Chapter 3

# Class Index

## 3.1 Class List

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


ACL
This class represents the ACL (Access Control List) as JSON object fetched from Fledge Storage 15
ACLMalformed
Custom exception ACLMalformed . . . . . 16
ACL::ACLReason
This class represents the ACL security change request . . . . . 17
ACLReasonMalformed
Custom exception ACLReasonMalformed . . . . . 18
Aggregate
Aggregate clause in a selection of records . . . . . 19
AllCategoriesEx
AllCategoriesEx . . . . . 19
AssetTracker
Asset tracking functionality . . . . . 20
AssetTrackingTable
A class to hold a set of asset tracking tuples that allows lookup by name . . . . . 26
AssetTrackingTuple
Used to represent an asset tracking tuple . . . . . 27
AssetTrackingTuplePtrEqual . . . . . 29
AttachDbSync
Used to synchronize the attach database operation . . . . . 29
AuditLogger
A singleton class for access to the audit logger within services . . . . . 30
BadRequest
BadRequest exception . . . . . 31
Base64DataBuffer
Provide functionality on top of the simple DataBuffer class that is used to encode the buffer in base64 such that it may be stored as string data . . . . . 32
Base64DPImage
The Base64DPImage provide functionality on top of the simple DPImage class that is used to encode the buffer in base64 such that it may be stored as string data . . . . . 34
BearerToken
This class represents a JWT bearer token . . . . . 35
BinaryPluginHandle
Used to represent an interface to a plugin that is available in a binary format . . . . . 36

CategoryDetailsEx
CategoryDetailsEx . . . 37
ConfigCategory::CategoryItem . . . 38
CheckUpdates
CheckUpdates class . . . 40
ChildCategoriesEx
ChildCategoriesEx . . . 41
ResultSet::ColumnValue . . . 42
ConfigCategories . . . 44
ConfigCategory . . . 45
ConfigCategoryChange . . . 65
ConfigCategoryDefaultWithValue
ConfigCategoryDefaultWithValue . . . 66
ConfigCategoryDescription . . . 67
ConfigCategoryEx
ConfigCategoryEx . . . 68
ConfigHandler
Handler class within a service to manage configuration changes . . . 69
ConfigItemAttributeNotFound
This exception must be raised when a requested item attribute does not exist . . . 72
ConfigItemNotAList
An attempt has been made to access a configuration item as a list when the item is not of type list 73
ConfigItemNotFound . . . 74
ConfigMalformed . . . 75
ConfigurationManager . . . 75
ConfigValueFoundWithDefault
This exception must be raised when at least one of the JSON items of a new being created category have both "value" and "default" fields . . . 83
Conflict
Conflict exception . . . 84
Connection . . . 85
ConnectionManager
Singleton class to manage Postgres connection pool . . . 92
CoreManagementApi
Management API server for a C++ microservice . . . 97
DataBuffer
Buffer type for storage of arbitrary buffers of data within a datapoint . . . 100
DataLoad
A class used in the North service to load data from the buffer . . . 102
Datapoint
Name and value pair used to represent a data value within an asset reading . . . 107
DatapointValue
Class to hold an actual reading value . . . 109
DataSender . . . 112
DefaultConfigCategory
DefaultConfigCategory . . . 114
DiskSpaceMonitor
A class to monitor the free disk space used to store the various storage databases . . . 115
DPImage
Simple Image class that will be used within data points to store image data . . . 117
ExistingChildCategories
ExistingChildCategories . . . 119
Expression
Class that defines data to be inserted or updated in a column within the table . . . 120
ExpressionValues . . . 121
FormData::FieldValue . . . 121
FilterPipeline
Used to represent a pipeline of filters applicable to a task/service . . . 122

FilterPlugin . . . 127
FilterPythonPluginHandle
The PythonPluginHandle class is used to represent an interface to a plugin that is available as a python script . . . 130
FledgeFilter . . . 132
FledgeProcess
Fledge process base class . . . 134
FormData
This class represents a parsed HTTP form data uploaded to SimpleWeb::Server<SimpleWeb↩::HTTP . . . 136
std::hash< AssetTrackingTuple * > . . . 137
std::hash< AssetTrackingTuple > . . . 138
std::hash< StorageAssetTrackingTuple * > . . . 138
std::hash< StorageAssetTrackingTuple > . . . 138
HttpSender . . . 138
Ingest
The ingest class is used to ingest asset readings . . . 140
IngestRate
A class used to track and report on the ingest rates of a data stream . . . 149
InsertValue
Class that defines data to be inserted or updated in a column within the table . . . 151
InsertValues . . . 152
Join
Join clause representation . . . 152
JSONPath
A simple implementation of a JSON Path search mechanism to use alongside RapidJSON . . . 153
JSONProperties
Class that defines JSON properties for update . . . 155
JSONProperty . . . 156
JSONProvider . . . 156
JSONReading
JSONReading class . . . 156
ACL::KeyValueItem . . . 158
LALookup
Linked Asset Information class . . . 158
LibcurlHttps . . . 160
Logger
Fledge Logger class used to log to syslog . . . 162
ManagementApi
Management API server for a C++ microservice . . . 167
ManagementClient
The management client class used by services and tasks to communicate with the management API of the Fledge core microservice . . . 172
MemConnectionManager
Singleton class to manage SQLite3 Memory connection pool . . . 187
NorthApi . . . 188
NorthFilterPipeline
Derived from FilterPipeline class and is used to represent a pipeline of filter applicable to sending process . . . 192
NorthPlugin
Class that represents a north plugin . . . 194
NorthPythonPluginHandle
Used to represent an interface to a South plugin that is available as a python script . . . 198
NorthService
The NorthService class . . . 199
NorthServiceProvider
A data provider class to return data in the north service ping response . . . 202

NoSuchCategory
NoSuchCategory . . . 203
NoSuchCategoryItem
NoSuchItem . . . 204
NoSuchCategoryItemValue
NoSuchCategoryItemValue . . . 205
NotificationPythonPluginHandle
The PythonPluginHandle class is used to represent an interface to a plugin that is available as a python script . . . 206
NotSupportedDataType
NotSupportedDataType . . . 208
OCS
The OCS class . . . 209
OMF
The OMF class . . . 210
OMFAFLocationHint
A AFLocation hint, tells us in which Asset Framework hierarchy the asset should be created . . . 223
OMFBuffer
Buffer class designed to hold OMF payloads that can grow as required but have minimal copy semantics . . . 224
OMFData
The OMFData class . . . 227
OMFDataTypes
Per asset dataTypes - This class is used in a std::map where assetName is a key . . . 228
OMFError
An encapsulation of an error return from an OMF call . . . 229
OMFHint
Virtual base class for an OMF Hint . . . 233
OMFHints
A set of hints for a reading . . . 235
OMFInformation
A class that holds the configuration information for the OMF plugin . . . 236
OMFIntegerHint
A integer hint, defines how integer type should be defined, int64, int32 or int16 . . . 238
OMFInterpolationHint
A interpolation hint, defines the interpolation value for a property . . . 239
OMFLegacyTypeHint
A Legacy type hint, tells the OMF plugin to send complex types for this asset . . . 240
OMFLinkedData
The OMFLinkedData class . . . 241
OMFMaximumHint
A maximum hint, defines the maximum value for a property . . . 244
OMFMinimumHint
A minimum hint, defines the minimum value for a property . . . 245
OMFNumberHint
A number hint, defines how number type should be defined, float64 or float32 . . . 246
OMFSourceHint
A Source hint, defines the data source for the asset or datapoint . . . 247
OMFTagHint
A tag hint, used to define an existing OMF container or tag to use . . . 248
OMFTagNameDatapointHint
A tag name hint, defines which PI Tag to use for a Datapoint . . . 249
OMFTagNameHint
A tag name hint, tells us which Container name to use in PI . . . 250
OMFTypeNameHint
A Type name hint, tells us how to name the types we use . . . 251
OMFUOMHint
A unit of measurement hint, defines the unit of measurement for a datapoint . . . 252

PerfMon . . . 253
PerformanceMonitor
Class to handle the performance monitors . . . 255
PipelineBranch
A pipeline element that represents a branch in the pipeline . . . 257
PipelineDebugger
The debugger class for elements in a pipeline . . . 260
PipelineElement
The base pipeline element class . . . 262
PipelineFilter
A pipeline element the runs a filter plugin . . . 265
PipelineWriter
A pipeline element that writes to a storage service or buffer . . . 268
PIWebAPI
The PIWebAPI class . . . 270
Plugin
A generic representation of a plugin . . . 272
PLUGIN_ERROR
Structure used by plugins to return error information . . . 273
PLUGIN_INFORMATION
The plugin infiornation structure, used to return information from a plugin during the laod and configuration stage . . . 273
plugin_parameter
Pass a name/value pair to a plugin . . . 274
PluginData . . . 274
PluginHandle
Used to represent an opaque handle to a plugin instance . . . 276
PluginManager
The manager for plugins . . . 276
PluginNotImplementedException
Implementation of PluginNotImplementedException . . . 278
ProfileItem . . . 279
PurgeConfiguration . . . 280
PurgeResult . . . 281
PurgeSystem . . . 282
PythonConfigCategory
A wrapper class for a ConfigCategory to convert to and from Python objects . . . 283
PythonModule
This class represents the loaded Python module with interpreter initialisation flag . . . 284
PythonPluginHandle
Base class used to represent an interface to a plugin that is available as a python script . . . 285
PythonReading
A wrapper class for a Reading to convert to and from Python objects . . . 287
PythonReadingSet
A wrapper class for the ReadingSet class that allows conversion to and from Python objects . . . 290
PythonRuntime . . . 292
Query
Storage layer query container . . . 293
QueryProfile . . . 299
RDSAcknowledge . . . 299
RDSBlockHeader . . . 300
RDSConnectHeader . . . 300
RDSReadingHeader . . . 300
Reading
An asset reading represented as a class . . . 301
ReadingCircularBuffer
A circular buffer of readings . . . 306
ReadingsCatalogue::ReadingReference . . . 309

ReadingsCatalogue
Implements the handling of multiples readings tables stored among multiple SQLite databases 309
ReadingSet
Reading set class . . . 324
ReadingSetCircularBuffer
Reading set circular buffer class . . . 328
ReadingSetException . . . 331
ReadingStream . . . 332
ResultException . . . 333
ResultIncorrectTypeException . . . 334
ResultNoMoreRowsException . . . 335
ResultNoSuchColumnException . . . 336
ResultSet
Result set . . . 336
Returns
Control a returned column . . . 341
ResultSet::Row . . . 341
Schema
Representation of an extension schema . . . 342
SchemaManager
The singleton SchemaManager class used to interact with the extension schemas created by various extension services . . . 344
SendingProcess . . . 347
ServiceAuthHandler
ServiceAuthHandler adds security to the base class ServiceHandler . . . 351
ServiceHandler
ServiceHandler abstract class - the interface that services using the management API must provide . . . 357
ServiceRecord . . . 358
ServiceRegistry
ServiceRegistry Singleton class . . . 359
SimpleHttp . . . 361
SimpleHttps . . . 363
Sort
Sort clause in a selection of records . . . 366
SouthApi . . . 366
SouthPlugin
Class that represents a south plugin . . . 371
SouthPythonPluginHandle
Used to represent an interface to a South plugin that is available as a python script . . . 374
SouthService
The SouthService class . . . 376
SouthServiceProvider
A data provider class to return data in the south service ping response . . . 380
SQLBuffer
Buffer class designed to hold SQL statement that can as required but have minimal copy semantics . . . 381
StatsHistory
StatisticsHisotry class . . . 384
STORAGE_CONFIGURATION . . . 386
StorageApi
The Storage API class - this class is responsible for the registration of all API entry points in the storage API and the dispatch of those API calls to the internals of the storage service and the storage plugin itself . . . 386
StorageAssetTrackingTuple . . . 395
StorageAssetTrackingTuplePtrEqual . . . 397
StorageClient
Client for accessing the storage service . . . 397

StorageConfiguration
The storage service must handle its own configuration differently to other services as it is unable to read the configuration from the database . . . 406
StorageOperation
StorageOperation . . . 407
StoragePerformanceMonitor
StoragePerformanceMonitor is a derived class from PerformanceMonitor It allows direct writing of monitoring data to database . . . 408
StoragePlugin
Class that represents a storage plugin . . . 409
StoragePluginConfiguration
The storage service must handle its own configuration differently to other services as it is unable to read the configuration from the database . . . 412
StorageRegistry
StorageRegistry - a class that manages requests from other microservices to register interest in new readings being inserted into the storage layer that match a given asset code, or any asset code "∗" . . . 413
StorageService
The StorageService class . . . 418
StorageStats . . . 419
StreamHandler . . . 420
TableReference
Class used to store table references . . . 421
TableRegistration . . . 422
Timebucket
Timebucket clause in a selection of records . . . 422
TrackingTuple
Tracking abstract base class to be passed in the process data queue . . . 423
TransactionBoundary
This class handles per thread started transaction boundaries: . . . 423
Unauthorized
Unauthorized exception . . . 425
UpdateModifier
Update modifier . . . 426
UpdateValue
A value in an update statement . . . 426
ACL::UrlItem . . . 427
Where
Where clause in a selection of records . . . 427

# Chapter 4

# Class Documentation

## 4.1 ACL Class Reference

This class represents the ACL (Access Control List) as JSON object fetched from Fledge Storage.

```
#include <acl.h>
```

### Classes

- class ACLReason

  *This class represents the ACL security change request.*
- class KeyValueItem
- class UrlItem

### Public Member Functions

- **ACL** (const std::string &json)
- const std::string & **getName** ()
- const std::vector< KeyValueItem > & **getService** ()
- const std::vector< UrlItem > & **getURL** ()

### 4.1.1 Detailed Description

This class represents the ACL (Access Control List) as JSON object fetched from Fledge Storage.

There are utility methods along with ACLReason class for changes handking

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

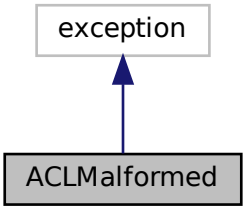
- C/common/include/acl.h

## 4.2 ACLMalformed Class Reference

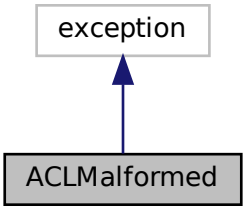
Custom exception ACLMalformed.

```
#include <acl.h>
```

Inheritance diagram for ACLMalformed:




Collaboration diagram for ACLMalformed:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.2.1 Detailed Description

Custom exception ACLMalformed.

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

- C/common/include/acl.h

## 4.3 ACL::ACLReason Class Reference

This class represents the ACL security change request.

```
#include <acl.h>
```

### Public Member Functions

- ACLReason (const std::string &reason)

  *ACLReason constructor: parse input JSON for ACL change reason.*
- const std::string & **getReason** ()
- const std::string & **getArgument** ()

### 4.3.1 Detailed Description

This class represents the ACL security change request.

Parsed JSON should have string attributes 'reason' and 'argument'

### 4.3.2 Constructor & Destructor Documentation

#### 4.3.2.1 ACLReason()

```
ACL::ACLReason::ACLReason (
            const std::string & reason )
```

ACLReason constructor: parse input JSON for ACL change reason.

JSON should have string attributes 'reason' and 'argument'

**Parameters**

| | |
|---|---|
| *json* | The JSON reason string to parse |

**Exceptions**

| | |
|---|---|
| *exception* | ACLReasonMalformed |

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

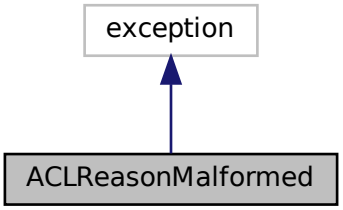
- C/common/include/acl.h
- C/common/acl.cpp

## 4.4 ACLReasonMalformed Class Reference

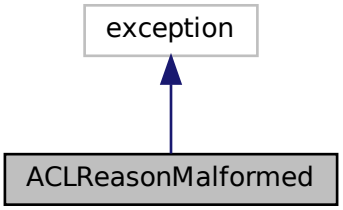
Custom exception ACLReasonMalformed.

```
#include <acl.h>
```

Inheritance diagram for ACLReasonMalformed:




Collaboration diagram for ACLReasonMalformed:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.4.1 Detailed Description

Custom exception ACLReasonMalformed.

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

- C/common/include/acl.h

## 4.5 Aggregate Class Reference

Aggregate clause in a selection of records.

```
#include <aggregate.h>
```

### Public Member Functions

- **Aggregate** (const std::string &operation, const std::string &column)
- std::string toJSON ()

  *Return the JSON payload for a where clause.*

### 4.5.1 Detailed Description

Aggregate clause in a selection of records.

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

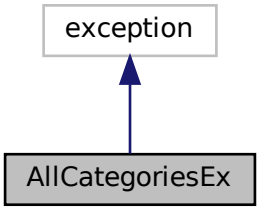
- C/common/include/aggregate.h
- C/common/aggregate.cpp

## 4.6 AllCategoriesEx Class Reference

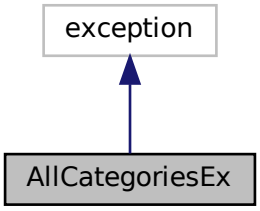
AllCategoriesEx.

```
#include <configuration_manager.h>
```

Inheritance diagram for AllCategoriesEx:



Collaboration diagram for AllCategoriesEx:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.6.1 Detailed Description

AllCategoriesEx.

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

- C/services/core/include/configuration_manager.h

## 4.7 AssetTracker Class Reference

The AssetTracker class provides the asset tracking functionality.

```
#include <asset_tracking.h>
```

### Public Member Functions

- AssetTracker (ManagementClient ∗mgtClient, std::string service)

  *AssetTracker class constructor.*
- ∼AssetTracker ()

  *Destructor for the asset tracker.*
- void populateAssetTrackingCache (std::string plugin, std::string event)

  *Fetch all asset tracking tuples from DB and populate local cache.*
- void populateStorageAssetTrackingCache ()

  *Fetch all storage asset tracking tuples from DB and populate local cache.*
- bool checkAssetTrackingCache (AssetTrackingTuple &tuple)

  *Check local cache for a given asset tracking tuple.*
- AssetTrackingTuple ∗ findAssetTrackingCache (AssetTrackingTuple &tuple)

  *Lookup tuple in the asset tracker cache.*

- void addAssetTrackingTuple (AssetTrackingTuple &tuple)

  *Add asset tracking tuple via microservice management API and in cache.*
- void **addAssetTrackingTuple** (std::string plugin, std::string asset, std::string event)
- void addStorageAssetTrackingTuple (StorageAssetTrackingTuple &tuple, std::set< std::string > &dpSet, bool addObj=false)

  *Add asset tracking tuple via microservice management API and in cache.*
- StorageAssetTrackingTuple * findStorageAssetTrackingCache (StorageAssetTrackingTuple &tuple)

  *Check if a StorageAssetTrackingTuple is in cache.*
- std::string **getIngestService** (const std::string &asset)
- std::string **getEgressService** (const std::string &asset)
- void workerThread ()

  *The worker thread that will flush any pending asset tuples to the database.*
- bool getDeprecated (StorageAssetTrackingTuple *ptr)

  *This function takes a StorageAssetTrackingTuple pointer and searches for it in cache, if found then returns its Deprecated status.*
- void updateCache (std::set< std::string > dpSet, StorageAssetTrackingTuple *ptr)

  *Updates datapoints present in the arg dpSet in the cache.*
- std::set< std::string > * getStorageAssetTrackingCacheData (StorageAssetTrackingTuple *tuple)

  *Get stored value in the StorageAssetTrackingTuple cache for the given tuple.*
- bool tune (unsigned long updateInterval)

  *Set the update interval for the asset tracker.*

## Static Public Member Functions

- static AssetTracker * getAssetTracker ()

  *Get asset tracker singleton instance for the current south service.*

### 4.7.1 Detailed Description

The AssetTracker class provides the asset tracking functionality.

There are methods to populate asset tracking cache from asset_tracker DB table, and methods to check/add asset tracking tuples to DB and to cache

### 4.7.2 Constructor & Destructor Documentation

#### 4.7.2.1 AssetTracker()

```
AssetTracker::AssetTracker (
            ManagementClient * mgtClient,
            std::string service )
```

AssetTracker class constructor.

**Parameters**

| | |
|---|---|
| *mgtClient* | Management client object for this south service |
| *service* | Service name |

#### 4.7.2.2 ~AssetTracker()

```
AssetTracker::~AssetTracker ( )
```

Destructor for the asset tracker.

We must make sure any pending tuples are written out before the asset tracker is destroyed.

### 4.7.3 Member Function Documentation

#### 4.7.3.1 addAssetTrackingTuple()

```
void AssetTracker::addAssetTrackingTuple (
            AssetTrackingTuple & tuple )
```

Add asset tracking tuple via microservice management API and in cache.

**Parameters**

| | |
|---|---|
| *tuple* | New tuple to add in DB and in cache |

#### 4.7.3.2 addStorageAssetTrackingTuple()

```
void AssetTracker::addStorageAssetTrackingTuple (
            StorageAssetTrackingTuple & tuple,
            std::set< std::string > & dpSet,
            bool addObj = false )
```

Add asset tracking tuple via microservice management API and in cache.

**Parameters**

| | |
|---|---|
| *tuple* | New tuple to add to the queue |
| *dpSet* | Set of datapoints to handle |
| *addObj* | Create a new obj for cache and queue if true. Otherwise just add current tuple to processing queue. |

#### 4.7.3.3 checkAssetTrackingCache()

```
bool AssetTracker::checkAssetTrackingCache (
            AssetTrackingTuple & tuple )
```

Check local cache for a given asset tracking tuple.

**Parameters**

| | |
|---|---|
| *tuple* | Tuple to find in cache |

**Returns**

Returns whether tuple is present in cache

#### 4.7.3.4 findAssetTrackingCache()

```
AssetTrackingTuple * AssetTracker::findAssetTrackingCache (
            AssetTrackingTuple & tuple )
```

Lookup tuple in the asset tracker cache.

**Parameters**

| | |
|---|---|
| *tuple* | The tuple to lookup |

**Returns**

NULL if the tuple is not in the cache or the tuple from the cache

#### 4.7.3.5 findStorageAssetTrackingCache()

```
StorageAssetTrackingTuple * AssetTracker::findStorageAssetTrackingCache (
            StorageAssetTrackingTuple & tuple )
```

Check if a StorageAssetTrackingTuple is in cache.

**Parameters**

| | |
|---|---|
| *tuple* | The StorageAssetTrackingTuple to find |

**Returns**

Pointer to found tuple or NULL

#### 4.7.3.6 getAssetTracker()

```
AssetTracker * AssetTracker::getAssetTracker ( ) [static]
```

Get asset tracker singleton instance for the current south service.

**Returns**

Singleton asset tracker instance

#### 4.7.3.7 getDeprecated()

```
bool AssetTracker::getDeprecated (
            StorageAssetTrackingTuple * ptr )
```

This function takes a StorageAssetTrackingTuple pointer and searches for it in cache, if found then returns its Deprecated status.

**Parameters**

| | |
|---|---|
| *ptr* | StorageAssetTrackingTuple∗ , as key in cache (map) |

**Returns**

bool Deprecation status

#### 4.7.3.8 getStorageAssetTrackingCacheData()

```
std::set< std::string > * AssetTracker::getStorageAssetTrackingCacheData (
            StorageAssetTrackingTuple * tuple )
```

Get stored value in the StorageAssetTrackingTuple cache for the given tuple.

**Parameters**

| | |
|---|---|
| *tuple* | The StorageAssetTrackingTuple to find |

**Returns**

Pointer to found std::set<std::string> result or NULL if tuble does not exist

#### 4.7.3.9 populateAssetTrackingCache()

```
void AssetTracker::populateAssetTrackingCache (
            std::string plugin,
            std::string event )
```

Fetch all asset tracking tuples from DB and populate local cache.

Return the vector of deprecated asset names

**Parameters**

| | |
|---|---|
| *plugin* | Plugin name |
| *event* | Event name |

#### 4.7.3.10 populateStorageAssetTrackingCache()

```
void AssetTracker::populateStorageAssetTrackingCache ( )
```

Fetch all storage asset tracking tuples from DB and populate local cache.

Return the vector of deprecated asset names

#### 4.7.3.11 tune()

```
bool AssetTracker::tune (
            unsigned long interval )
```

Set the update interval for the asset tracker.

**Parameters**

| | |
|---|---|
| *interval* | The number of milliseconds between update of the asset tracker |

**Returns**

bool Was the update accepted

#### 4.7.3.12 updateCache()

```
void AssetTracker::updateCache (
            std::set< std::string > dpSet,
            StorageAssetTrackingTuple * ptr )
```

Updates datapoints present in the arg dpSet in the cache.

**Parameters**

| | |
|---|---|
| *dpSet* | set of datapoints string values to be updated in cache |
| *ptr* | StorageAssetTrackingTuple∗ , as key in cache (map) Retval void |

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

- C/common/include/asset_tracking.h
- C/common/asset_tracking.cpp

## 4.8 AssetTrackingTable Class Reference

A class to hold a set of asset tracking tuples that allows lookup by name.

```
#include <asset_tracking.h>
```

### Public Member Functions

- AssetTrackingTable ()

  *Constructor for an asset tracking tuple table.*
- ∼AssetTrackingTable ()

  *Destructor for asset tracking tuple table.*
- void add (AssetTrackingTuple ∗tuple)

  *Add a tuple to an asset tracking table.*
- void remove (const std::string &name)

  *Remove an asset tracking tuple from the table.*
- AssetTrackingTuple ∗ find (const std::string &name)

  *Find the named asset tuple and return a pointer to te asset.*

### 4.8.1 Detailed Description

A class to hold a set of asset tracking tuples that allows lookup by name.

### 4.8.2 Member Function Documentation

#### 4.8.2.1 add()

```
void AssetTrackingTable::add (
            AssetTrackingTuple * tuple )
```

Add a tuple to an asset tracking table.

**Parameters**

| | |
|---|---|
| *tuple* | Pointer to the asset tracking tuple to add |

#### 4.8.2.2 find()

```
AssetTrackingTuple * AssetTrackingTable::find (
            const std::string & name )
```

Find the named asset tuple and return a pointer to te asset.

**Parameters**

| | |
|---|---|
| *name* | The name of the asset to lookup |

**Returns**

AssetTrackingTupple∗ The matchign tuple or NULL

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

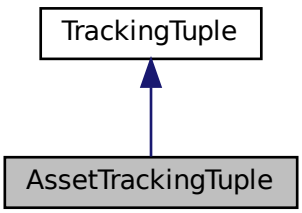
- C/common/include/asset_tracking.h
- C/common/asset_tracking.cpp

## 4.9 AssetTrackingTuple Class Reference

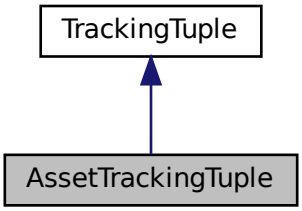
The AssetTrackingTuple class is used to represent an asset tracking tuple.

```
#include <asset_tracking.h>
```

Inheritance diagram for AssetTrackingTuple:



Collaboration diagram for AssetTrackingTuple:




## Public Member Functions

- std::string **assetToString** ()
- bool **operator==** (const AssetTrackingTuple &x) const
- **AssetTrackingTuple** (const std::string &service, const std::string &plugin, const std::string &asset, const std::string &event, const bool &deprecated=false)
- std::string & **getAssetName** ()
- std::string **getPluginName** ()
- std::string **getEventName** ()
- std::string **getServiceName** ()
- bool **isDeprecated** ()
- void **unDeprecate** ()
- InsertValues processData (bool storage_connected, ManagementClient *mgtClient, bool &warned, std::string &instanceName)

  *Insert AssetTrackingTuple data via Fledge core API or prepare InsertValues object for direct DB operation.*

## Public Attributes

- std::string **m_serviceName**
- std::string **m_pluginName**
- std::string **m_assetName**
- std::string **m_eventName**

### 4.9.1 Detailed Description

The AssetTrackingTuple class is used to represent an asset tracking tuple.

Hash function and '==' operator are defined for this class and pointer to this class that would be required to create an unordered_set of this class.

### 4.9.2 Member Function Documentation

#### 4.9.2.1 processData()

```
InsertValues AssetTrackingTuple::processData (
            bool storage_connected,
            ManagementClient * mgtClient,
            bool & warned,
            std::string & instanceName )  [virtual]
```

Insert AssetTrackingTuple data via Fledge core API or prepare InsertValues object for direct DB operation.

**Parameters**

| | |
|---|---|
| *storage* | Boolean for storage being available |
| *mgtClient* | ManagementClient object pointer |
| *warned* | Boolean ireference updated for logging operation |
| *instanceName* | Fledge instance name |

**Returns**

InsertValues object

Implements TrackingTuple.

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

- C/common/include/asset_tracking.h
- C/common/asset_tracking.cpp

## 4.10 AssetTrackingTuplePtrEqual Struct Reference

### Public Member Functions

- bool **operator()** (AssetTrackingTuple const ∗a, AssetTrackingTuple const ∗b) const

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

- C/common/include/asset_tracking.h

## 4.11 AttachDbSync Class Reference

Used to synchronize the attach database operation.

```
#include <readings_catalogue.h>
```

### Public Member Functions

- void **lock** ()
- void **unlock** ()

### Static Public Member Functions

- static AttachDbSync ∗ **getInstance** ()

### 4.11.1 Detailed Description

Used to synchronize the attach database operation.

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

- C/plugins/storage/sqlite/common/include/readings_catalogue.h

## 4.12 AuditLogger Class Reference

A singleton class for access to the audit logger within services.

```
#include <audit_logger.h>
```

### Public Member Functions

- AuditLogger (ManagementClient ∗mgmt)

  *Constructor for an audit logger that is passed the management client.*
- ∼AuditLogger ()

  *Destructor for an audit logger.*
- void audit (const std::string &code, const std::string &level, const std::string &data="")

  *Log an audit message.*

### Static Public Member Functions

- static AuditLogger ∗ getLogger ()

  *Get the audit logger singleton.*
- static void **auditLog** (const std::string &code, const std::string &level, const std::string &data="")

### 4.12.1 Detailed Description

A singleton class for access to the audit logger within services.

The service must create this with the maagement client before any access to it is used.

### 4.12.2 Constructor & Destructor Documentation

#### 4.12.2.1 AuditLogger()

```
AuditLogger::AuditLogger (
            ManagementClient * mgmt )
```

Constructor for an audit logger that is passed the management client.

This must be called early in a service or task creation before any audit logs are created.

**Parameters**

| | |
|---|---|
| *mgmt* | Pointer to the management client |

### 4.12.3 Member Function Documentation

#### 4.12.3.1 audit()

```
void AuditLogger::audit (
            const std::string & code,
            const std::string & level,
            const std::string & data = "" )
```

Log an audit message.

**Parameters**

| | |
|---|---|
| *code* | The audit code |
| *level* | The audit level |
| *data* | Optional data associated with the audit entry |

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

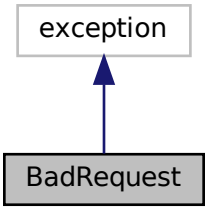
- C/common/include/audit_logger.h
- C/common/audit_logger.cpp

## 4.13 BadRequest Class Reference

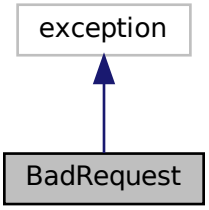
BadRequest exception.

```
#include <http_sender.h>
```

Inheritance diagram for BadRequest:



Collaboration diagram for BadRequest:




### Public Member Functions

- **BadRequest** (const std::string &serverReply)
- virtual const char ∗ **what** () const throw ()

### 4.13.1 Detailed Description

BadRequest exception.

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

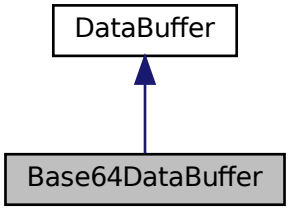
- C/plugins/common/include/http_sender.h

## 4.14 Base64DataBuffer Class Reference

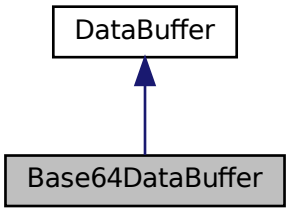
The Base64DataBuffer class provide functionality on top of the simple DataBuffer class that is used to encode the buffer in base64 such that it may be stored as string data.

```
#include <base64databuffer.h>
```

Inheritance diagram for Base64DataBuffer:



Collaboration diagram for Base64DataBuffer:




## Public Member Functions

- Base64DataBuffer (const std::string &encoded)

  *Construct a DataBuffer by decoding a Base64 encoded buffer.*
- std::string encode ()

  *Base 64 encode the DataBuffer.*

## Additional Inherited Members

### 4.14.1 Detailed Description

The Base64DataBuffer class provide functionality on top of the simple DataBuffer class that is used to encode the buffer in base64 such that it may be stored as string data.

### 4.14.2 Member Function Documentation

#### 4.14.2.1 encode()

```
string Base64DataBuffer::encode ( )
```

Base 64 encode the DataBuffer.

Not the first character is not the data itself but an unencoded value for itemSize

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

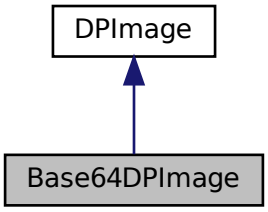
- C/common/include/base64databuffer.h
- C/common/base64databuffer.cpp

## 4.15 Base64DPImage Class Reference

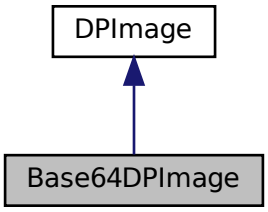
The Base64DPImage provide functionality on top of the simple DPImage class that is used to encode the buffer in base64 such that it may be stored as string data.

```
#include <base64dpimage.h>
```

Inheritance diagram for Base64DPImage:




Collaboration diagram for Base64DPImage:




### Public Member Functions

- Base64DPImage (const std::string &encoded)

  *Construct a DPImage by decoding a Base64 encoded buffer.*
- std::string encode ()

  *Base 64 encode the DPImage.*

### Additional Inherited Members

### 4.15.1 Detailed Description

The Base64DPImage provide functionality on top of the simple DPImage class that is used to encode the buffer in base64 such that it may be stored as string data.

### 4.15.2 Member Function Documentation

#### 4.15.2.1 encode()

```
string Base64DPImage::encode ( )
```

Base 64 encode the DPImage.

Note the first character is not the data itself but an unencoded value for itemSize

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

- C/common/include/base64dpimage.h
- C/common/base64image.cpp

## 4.16 BearerToken Class Reference

This class represents a JWT bearer token.

```
#include <bearer_token.h>
```

### Public Member Functions

- **BearerToken** (std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Request > request)
- BearerToken (std::string &token)

  *BearerToken constructor with string reference.*
- bool **exists** ()
- const std::string & **token** ()
- bool verify (const std::string &serverResponse)

  *BearerToken verification from JSON string reference.*
- unsigned long **getExpiration** ()
- const std::string & **getAudience** ()
- const std::string & **getSubject** ()
- const std::string & **getIssuer** ()

### 4.16.1 Detailed Description

This class represents a JWT bearer token.

The claims are stored after verification to core service API endpoint

### 4.16.2 Constructor & Destructor Documentation

#### 4.16.2.1 BearerToken()

```
BearerToken::BearerToken (
            std::string & token )
```

BearerToken constructor with string reference.

**Parameters**

| | |
|---|---|
| *token* | Bearer token string |

### 4.16.3 Member Function Documentation

#### 4.16.3.1 verify()

```
bool BearerToken::verify (
            const std::string & serverResponse )
```

BearerToken verification from JSON string reference.

Known token claims as stored as strings

**Parameters**

| | |
|---|---|
| *response* | JSON string from token verification endpoint |

**Returns**

True on success False otherwise

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

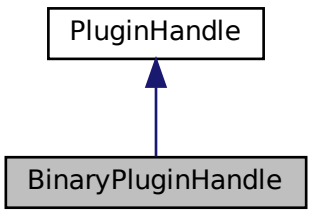
- C/common/include/bearer_token.h
- C/common/bearer_token.cpp

## 4.17 BinaryPluginHandle Class Reference

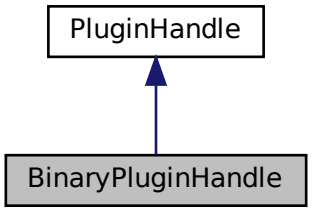
The BinaryPluginHandle class is used to represent an interface to a plugin that is available in a binary format.

`#include <binary_plugin_handle.h>`

Inheritance diagram for BinaryPluginHandle:



Collaboration diagram for BinaryPluginHandle:




### Public Member Functions

- **BinaryPluginHandle** (const char ∗name, const char ∗path, tPluginType type)
- **BinaryPluginHandle** (const char ∗name, const char ∗path)
- void ∗ **GetInfo** ()
- void ∗ **ResolveSymbol** (const char ∗sym)
- void ∗ **getHandle** ()

### 4.17.1 Detailed Description

The BinaryPluginHandle class is used to represent an interface to a plugin that is available in a binary format.

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

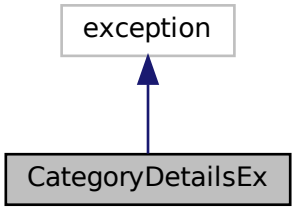
- C/services/common/include/binary_plugin_handle.h

## 4.18 CategoryDetailsEx Class Reference

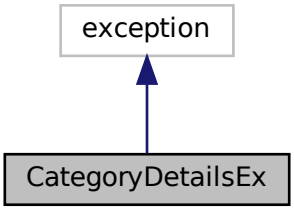
CategoryDetailsEx.

```
#include <configuration_manager.h>
```

Inheritance diagram for CategoryDetailsEx:



Collaboration diagram for CategoryDetailsEx:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.18.1 Detailed Description

CategoryDetailsEx.

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

- C/services/core/include/configuration_manager.h

## 4.19 ConfigCategory::CategoryItem Class Reference

### Public Member Functions

- **CategoryItem** (const std::string &name, const rapidjson::Value &item)
- **CategoryItem** (const std::string &name, const std::string &description, const std::string &type, const std↩ ::string def, const std::string &value)
- **CategoryItem** (const std::string &name, const std::string &description, const std::string def, const std::string &value, const std::vector< std::string > options)
- CategoryItem (const CategoryItem &rhs)

  *Copy constructor for configuration item.*
- std::string toJSON (const bool full=false) const

  *Create a JSON representation of the configuration item.*
- std::string defaultToJSON () const

  *Return only "default" item values.*

## Public Attributes

- std::string **m_name**
- std::string **m_displayName**
- std::string **m_type**
- std::string **m_default**
- std::string **m_value**
- std::string **m_description**
- std::string **m_order**
- std::string **m_readonly**
- std::string **m_mandatory**
- std::string **m_deprecated**
- std::string **m_length**
- std::string **m_minimum**
- std::string **m_maximum**
- std::string **m_filename**
- std::vector< std::string > **m_options**
- std::string **m_file**
- ItemType **m_itemType**
- std::string **m_validity**
- std::string **m_group**
- std::string **m_rule**
- std::string **m_bucketProperties**
- std::string **m_listSize**
- std::string **m_listItemType**
- std::string **m_listName**
- std::string **m_kvlistKeyName**
- std::string **m_kvlistKeyDescription**
- std::vector< std::string > **m_permissions**
- std::string **m_jsonSchema**

## 4.19.1 Member Function Documentation

### 4.19.1.1 defaultToJSON()

```
string ConfigCategory::CategoryItem::defaultToJSON ( ) const
```

Return only "default" item values.

NOTE: These data types must be all escaped. "default" items in the DefaultConfigCategory class are sent to ConfigurationManager interface which requires string values only:

examples: we must use "100" not 100 and for JSON "{\"pipeline":["scale"]}" not {"pipeline":["scale"]}

### 4.19.1.2 toJSON()

```
string ConfigCategory::CategoryItem::toJSON (
            const bool full = false ) const
```

Create a JSON representation of the configuration item.

**Parameters**

| | |
|---|---|
| *full* | false is the default, true evaluates all the members of the CategoryItem |

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

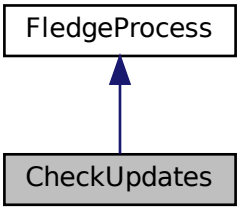
- C/common/include/config_category.h
- C/common/config_category.cpp

## 4.20 CheckUpdates Class Reference

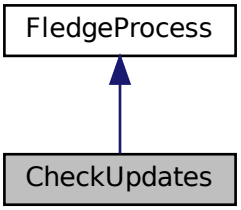
CheckUpdates class.

```
#include <check_updates.h>
```

Inheritance diagram for CheckUpdates:




Collaboration diagram for CheckUpdates:



### Public Member Functions

- CheckUpdates (int argc, char **argv)

  *Constructor for CheckUpdates.*
- ~CheckUpdates ()

  *Destructor for CheckUpdates.*
- void run ()

  *CheckUpdates run method, called by the base class to start the process and do the actual work.*

### Additional Inherited Members

### 4.20.1 Detailed Description

CheckUpdates class.

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

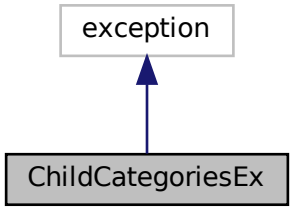
- C/tasks/check_updates/include/check_updates.h
- C/tasks/check_updates/check_updates.cpp

## 4.21 ChildCategoriesEx Class Reference

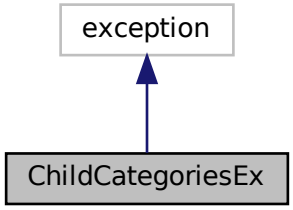
ChildCategoriesEx.

```
#include <configuration_manager.h>
```

Inheritance diagram for ChildCategoriesEx:



Collaboration diagram for ChildCategoriesEx:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.21.1 Detailed Description

ChildCategoriesEx.

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

- C/services/core/include/configuration_manager.h

## 4.22 ResultSet::ColumnValue Class Reference

### Public Member Functions

- **ColumnValue** (const std::string &value)
- **ColumnValue** (const int value)
- **ColumnValue** (const long value)
- **ColumnValue** (const double value)
- **ColumnValue** (const rapidjson::Value &value)
- ColumnType **getType** ()
- long getInteger () const

  *Retrieve a column value as an integer.*
- double getNumber () const

  *Retrieve a column value as a floating point number.*
- char ∗ getString () const

  *Retrieve a column value as a string.*
- const rapidjson::Value ∗ **getJSON** () const

### 4.22.1 Member Function Documentation

#### 4.22.1.1 getInteger()

```
long ResultSet::ColumnValue::getInteger ( ) const
```

Retrieve a column value as an integer.

**Returns**

long Integer value

**Exceptions**

| | |
|---|---|
| *ResultIncorrectTypeException* | The column can not be returned as an integer |

#### 4.22.1.2 getNumber()

```
double ResultSet::ColumnValue::getNumber ( ) const
```

Retrieve a column value as a floating point number.

**Returns**

double Floating point value

**Exceptions**

| | |
|---|---|
| *ResultIncorrectTypeException* | The column can not be returned as a double |

#### 4.22.1.3 getString()

```
char * ResultSet::ColumnValue::getString ( ) const
```

Retrieve a column value as a string.

**Returns**

double Floating point value

**Exceptions**

| | |
|---|---|
| *ResultIncorrectTypeException* | The column can not be returned as a double |

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

- C/common/include/resultset.h
- C/common/result_set.cpp

## 4.23 ConfigCategories Class Reference

### Public Member Functions

- ConfigCategories (const std::string &json)

  *Construct a ConfigCategories object from a JSON document returned from the Fledge configuratrion service.*
- ConfigCategories ()

  *ConfigCategories constructor without parameters.*
- ~ConfigCategories ()

  *ConfigCategories destructor.*
- unsigned int **length** ()
- ConfigCategoryDescription * **operator[]** (const unsigned int idx)
- void addCategoryDescription (ConfigCategoryDescription *elem)

  *Add a ConfigCategoryDescription element.*
- std::string toJSON () const

  *Return the JSON string of all ConfigCategoryDescription elements in m_categories.*

### 4.23.1 Constructor & Destructor Documentation

#### 4.23.1.1 ConfigCategories()

```
ConfigCategories::ConfigCategories ( )
```

ConfigCategories constructor without parameters.

Elements can be added with ConfigCategories::addCategoryDescription

### 4.23.2 Member Function Documentation

#### 4.23.2.1 addCategoryDescription()

```
void ConfigCategories::addCategoryDescription (
            ConfigCategoryDescription * elem )
```

Add a ConfigCategoryDescription element.

**Parameters**

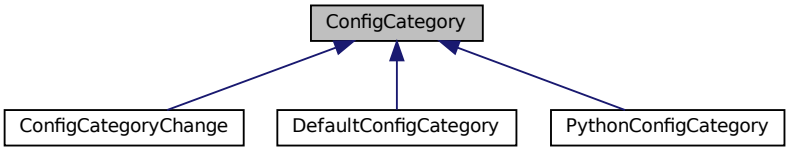
| | |
|---|---|
| *elem* | The ConfigCategoryDescription elemen to add |

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

- C/common/include/config_category.h
- C/common/config_category.cpp

## 4.24 ConfigCategory Class Reference

Inheritance diagram for ConfigCategory:




### Classes

- class CategoryItem

### Public Types

- enum **ItemType** {
  **UnknownType**, **StringItem**, **EnumerationItem**, **JsonItem**,
  **BoolItem**, **NumberItem**, **DoubleItem**, **ScriptItem**,
  **CategoryType**, **CodeItem**, **BucketItem**, **ListItem**,
  **KVListItem** }
- enum **ItemAttribute** {
  **ORDER_ATTR**, **READONLY_ATTR**, **MANDATORY_ATTR**, **FILE_ATTR**,
  **MINIMUM_ATTR**, **MAXIMUM_ATTR**, **LENGTH_ATTR**, **VALIDITY_ATTR**,
  **GROUP_ATTR**, **DISPLAY_NAME_ATTR**, **DEPRECATED_ATTR**, **RULE_ATTR**,
  **BUCKET_PROPERTIES_ATTR**, **LIST_SIZE_ATTR**, **ITEM_TYPE_ATTR**, **LIST_NAME_ATTR**,
  **KVLIST_KEY_NAME_ATTR**, **KVLIST_KEY_DESCRIPTION_ATTR**, **JSON_SCHEMA_ATTR** }
- using **iterator** = std::vector< CategoryItem * >::iterator
- using **const_iterator** = std::vector< CategoryItem * >::const_iterator

### Public Member Functions

- **ConfigCategory** (const std::string &name, const std::string &json)
- ConfigCategory (const ConfigCategory &orig)

  *Copy constructor for a configuration category.*
- ConfigCategory (const ConfigCategory *orig)

  *Copy constructor for a configuration category when copying from a pointer.*
- ~ConfigCategory ()

  *Configuration category destructor.*

- void addItem (const std::string &name, const std::string description, const std::string &type, const std::string def, const std::string &value)

  *Add an item to a configuration category.*
- void **addItem** (const std::string &name, const std::string description, const std::string def, const std::string &value, const std::vector< std::string > options)
- void removeItems ()

  *Delete all the items from the configuration category.*
- void removeItemsType (ItemType type)

  *Delete all the items from the configuration category having a specific type.*
- void keepItemsType (ItemType type)

  *Delete all the items from the configuration category not having a specific type.*
- bool extractSubcategory (ConfigCategory &subCategories)

  *Extracts, process and adds subcategory information from a given category to the current instance.*
- void setDescription (const std::string &description)

  *Set the description for the configuration category.*
- std::string **getName** () const
- std::string **getDescription** () const
- std::string **getDisplayName** () const
- void **setDisplayName** (const std::string &displayName)
- unsigned int **getCount** () const
- bool itemExists (const std::string &name) const

  *Check for the existence of an item within the configuration category.*
- bool setItemDisplayName (const std::string &name, const std::string &displayName)

  *Set the display name of an item.*
- std::string **getValue** (const std::string &name) const
- std::string getValue (const std::string &name, const std::string &defaultValue) const

  *Return the value of the configuration category item with a default.*
- bool getBoolValue (const std::string &name, bool defaultValue=false) const

  *Return a boolean value from a configuration category item.*
- int getIntegerValue (const std::string &name, int defaultValue=0) const

  *Return an integer value from a configuration category item.*
- long getLongValue (const std::string &name, long defaultValue=0) const

  *Return a long value from a configuration category item.*
- double getDoubleValue (const std::string &name, double defaultValue=0) const

  *Return a double value from a configuration category item.*
- std::vector< std::string > getValueList (const std::string &name) const

  *Return the value of the configuration category item list, this is a convience function used when simple lists are defined and allows for central processing of the list values.*
- std::map< std::string, std::string > getValueKVList (const std::string &name) const

  *Return the value of the configuration category item kvlist, this is a convience function used when key/value lists are defined and allows for central processing of the list values.*
- std::string getType (const std::string &name) const

  *Return the type of the configuration category item.*
- std::string **getDescription** (const std::string &name) const
- std::string getDefault (const std::string &name) const

  *Return the default value of the configuration category item.*
- bool setDefault (const std::string &name, const std::string &value)

  *Update the default value of the configuration category item.*
- bool setValue (const std::string &name, const std::string &value)

  *Update the value of the configuration category item.*
- std::string **getDisplayName** (const std::string &name) const
- std::string **getmParentName** () const

- std::vector< std::string > getOptions (const std::string &name) const

  *Return the options of the configuration category item.*
- std::string getLength (const std::string &name) const

  *Return the length value of the configuration category item.*
- std::string getMinimum (const std::string &name) const

  *Return the minimum value of the configuration category item.*
- std::string getMaximum (const std::string &name) const

  *Return the maximum of the configuration category item.*
- bool isString (const std::string &name) const

  *Return if the configuration item is a string item.*
- bool isEnumeration (const std::string &name) const

  *Return if the configuration item is an enumeration item.*
- bool isJSON (const std::string &name) const

  *Return if the configuration item is a JSON item.*
- bool isBool (const std::string &name) const

  *Return if the configuration item is a Bool item.*
- bool isNumber (const std::string &name) const

  *Return if the configuration item is a Numeric item.*
- bool isDouble (const std::string &name) const

  *Return if the configuration item is a Double item.*
- bool isList (const std::string &name) const

  *Return if the configuration item is a list item.*
- bool isKVList (const std::string &name) const

  *Return if the configuration item is a kvlist item.*
- bool isDeprecated (const std::string &name) const

  *Return if the configuration item is deprecated a item.*
- std::string toJSON (const bool full=false) const

  *Return JSON string of all category components.*
- std::string itemsToJSON (const bool full=false) const

  *Return JSON string of category items only.*
- ConfigCategory & operator= (ConfigCategory const &rhs)

  *Operator= for ConfigCategory.*
- ConfigCategory & operator+= (ConfigCategory const &rhs)

  *Operator+= for ConfigCategory.*
- void setItemsValueFromDefault ()

  *Set the m_value from m_default for each item.*
- void checkDefaultValuesOnly () const

  *Check whether at least one item in the category object has both 'value' and 'default' set.*
- std::string itemToJSON (const std::string &itemName) const

  *Return JSON string of a category item.*
- std::string to_string (const rapidjson::Value &v) const

  *Convert a RapidJSON value to a string.*
- std::vector< std::string > getPermissions (const std::string &name) const

  *Return the permissions of the configuration category item.*
- bool hasPermission (const std::string &name, const std::string &username) const

  *Return true if the user has permission to update the named item.*
- std::string getItemAttribute (const std::string &itemName, ItemAttribute itemAttribute) const

  *Return the requested attribute of a configuration category item.*
- bool setItemAttribute (const std::string &itemName, ItemAttribute itemAttribute, const std::string &value)

  *Set the requested attribute of a configuration category item.*
- std::vector< std::pair< std::string, std::string > > * parseBucketItemValue (const std::string &)

*Parse BucketItem value in JSON dict format and return the key value pairs within that.*

- const_iterator **begin** () const
- const_iterator **end** () const
- const_iterator **cbegin** () const
- const_iterator **cend** () const

## Protected Attributes

- std::vector< CategoryItem * > **m_items**
- std::string **m_name**
- std::string **m_parent_name**
- std::string **m_description**
- std::string **m_displayName**

## 4.24.1 Constructor & Destructor Documentation

### 4.24.1.1 ConfigCategory() [1/2]

```
ConfigCategory::ConfigCategory (
            const ConfigCategory & orig )
```

Copy constructor for a configuration category.

**Parameters**

| | |
|---|---|
| *rhs* | The configuration category to copy |

### 4.24.1.2 ConfigCategory() [2/2]

```
ConfigCategory::ConfigCategory (
            const ConfigCategory * orig )
```

Copy constructor for a configuration category when copying from a pointer.

**Parameters**

| | |
|---|---|
| *rhs* | The configuration category to copy |

## 4.24.2 Member Function Documentation

#### 4.24.2.1 checkDefaultValuesOnly()

```
void ConfigCategory::checkDefaultValuesOnly ( ) const
```

Check whether at least one item in the category object has both 'value' and 'default' set.

**Exceptions**

| | |
|---|---|
| *ConfigValueFoundWithDefault* | |

#### 4.24.2.2 extractSubcategory()

```
bool ConfigCategory::extractSubcategory (
            ConfigCategory & subCategories )
```

Extracts, process and adds subcategory information from a given category to the current instance.

**Parameters**

| | |
|---|---|
| *subCategories* | Configuration category from which the subcategories information should be extracted |

- 

#### 4.24.2.3 getBoolValue()

```
bool ConfigCategory::getBoolValue (
            const std::string & name,
            bool defaultValue = false ) const
```

Return a boolean value from a configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item |
| *defaultValue* | The value to return if item is not found or invalid |

**Returns**

bool The boolean value

#### 4.24.2.4 getDefault()

```
string ConfigCategory::getDefault (
            const std::string & name ) const
```

Return the default value of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.5 getItemAttribute()

```
string ConfigCategory::getItemAttribute (
            const std::string & itemName,
            ItemAttribute itemAttribute ) const
```

Return the requested attribute of a configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |
| *itemAttribute* | The item attribute (such as "file", "order", "readonly" |

**Returns**

The configuration item attribute as string

**Exceptions**

| | |
|---|---|
| *ConfigItemNotFound* | if the item does not exist in the category ConfigItemAttributeNotFound if the requested attribute does not exist for the found item. |

#### 4.24.2.6 getLength()

```
string ConfigCategory::getLength (
```

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

Return the length value of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.7 getMaximum()

```
string ConfigCategory::getMaximum (
            const std::string & name ) const
```

Return the maximum of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.8 getMinimum()

```
string ConfigCategory::getMinimum (
            const std::string & name ) const
```

Return the minimum value of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.9 getOptions()

```
vector< string > ConfigCategory::getOptions (
            const std::string & name ) const
```

Return the options of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.10 getPermissions()

```
vector< string > ConfigCategory::getPermissions (
            const std::string & name ) const
```

Return the permissions of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

vector<string> The configuration item permissions

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.11 getType()

```
string ConfigCategory::getType (
            const std::string & name ) const
```

Return the type of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.12 getValue()

```
string ConfigCategory::getValue (
            const std::string & name,
            const std::string & defaultValue ) const
```

Return the value of the configuration category item with a default.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |
| *defaultValue* | The default value to return if the item does not exist |

**Returns**

string The configuration item value or the default

#### 4.24.2.13 getValueKVList()

```
map< string, string > ConfigCategory::getValueKVList (
            const std::string & name ) const
```

Return the value of the configuration category item kvlist, this is a convience function used when key/value lists are defined and allows for central processing of the list values.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.14 getValueList()

```
vector< string > ConfigCategory::getValueList (
            const std::string & name ) const
```

Return the value of the configuration category item list, this is a convience function used when simple lists are defined and allows for central processing of the list values.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |

**Returns**

string The configuration item name

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.15 hasPermission()

```
bool ConfigCategory::hasPermission (
```

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

Return true if the user has permission to update the named item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |
| *rolename* | The name of the user role to test |

**Returns**

bool True if the named user can update the configuration item

**Exceptions**

| | |
|---|---|
| *exception* | if the item does not exist in the category |

#### 4.24.2.16 isBool()

```
bool ConfigCategory::isBool (
            const std::string & name ) const
```

Return if the configuration item is a Bool item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a Bool type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.17 isDeprecated()

```
bool ConfigCategory::isDeprecated (
            const std::string & name ) const
```

Return if the configuration item is deprecated a item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a deprecated type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.18 isDouble()

```
bool ConfigCategory::isDouble (
            const std::string & name ) const
```

Return if the configuration item is a Double item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a Double type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.19 isEnumeration()

```
bool ConfigCategory::isEnumeration (
            const std::string & name ) const
```

Return if the configuration item is an enumeration item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a string type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.20 isJSON()

```
bool ConfigCategory::isJSON (
            const std::string & name ) const
```

Return if the configuration item is a JSON item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a JSON type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.21 isKVList()

```
bool ConfigCategory::isKVList (
            const std::string & name ) const
```

Return if the configuration item is a kvlist item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a Numeric type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.22 isList()

```
bool ConfigCategory::isList (
            const std::string & name ) const
```

Return if the configuration item is a list item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a Numeric type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.23 isNumber()

```
bool ConfigCategory::isNumber (
            const std::string & name ) const
```

Return if the configuration item is a Numeric item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a Numeric type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.24 isString()

```
bool ConfigCategory::isString (
            const std::string & name ) const
```

Return if the configuration item is a string item.

**Parameters**

| | |
|---|---|
| *name* | The name of the item to test |

**Returns**

bool True if the item is a string type

**Exceptions**

| | |
|---|---|
| *exception* | If the item was not found in the configuration category |

#### 4.24.2.25 itemExists()

```
bool ConfigCategory::itemExists (
            const std::string & name ) const
```

Check for the existence of an item within the configuration category.

**Parameters**

| | |
|---|---|
| *name* | Item name to check within the category |

#### 4.24.2.26 itemsToJSON()

```
string ConfigCategory::itemsToJSON (
            const bool full = false ) const
```

Return JSON string of category items only.

**Parameters**

| | |
|---|---|
| *full* | false is the deafult, true evaluates all the members of the CategoryItems |

#### 4.24.2.27 itemToJSON()

```
string ConfigCategory::itemToJSON (
            const std::string & itemName ) const
```

Return JSON string of a category item.

**Parameters**

| | |
|---|---|
| *itemName* | The given item within current category |

**Returns**

The JSON string version of itemName If not found {} is returned

#### 4.24.2.28 keepItemsType()

```
void ConfigCategory::keepItemsType (
            ConfigCategory::ItemType type )
```

Delete all the items from the configuration category not having a specific type.

**Parameters**

| | |
|---|---|
| *type* | Type to maintain |

•

#### 4.24.2.29 parseBucketItemValue()

```
vector< pair< string, string > > * ConfigCategory::parseBucketItemValue (
            const std::string &  )
```

Parse BucketItem value in JSON dict format and return the key value pairs within that.

**Parameters**

| | |
|---|---|
| *json* | JSON string representing the BucketItem value |

**Returns**

Vector with pairs of found key/value string pairs in BucketItem value

#### 4.24.2.30 removeItems()

```
void ConfigCategory::removeItems ( )
```

Delete all the items from the configuration category.

#### 4.24.2.31 removeItemsType()

```
void ConfigCategory::removeItemsType (
            ConfigCategory::ItemType type )
```

Delete all the items from the configuration category having a specific type.

**Parameters**

| | |
|---|---|
| *type* | Type to delete |

- 

#### 4.24.2.32 setDefault()

```
bool ConfigCategory::setDefault (
            const std::string & name,
            const std::string & value )
```

Update the default value of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to update |
| *value* | New value of the configuration item |

**Returns**

bool Whether update succeeded

#### 4.24.2.33 setDescription()

```
void ConfigCategory::setDescription (
            const std::string & description )
```

Set the description for the configuration category.

**Parameters**

| | |
|---|---|
| *description* | The configuration category description |

#### 4.24.2.34 setItemAttribute()

```
bool ConfigCategory::setItemAttribute (
            const std::string & itemName,
            ItemAttribute itemAttribute,
            const std::string & value )
```

Set the requested attribute of a configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to return |
| *itemAttribute* | The item attribute (such as "file", "order", "readonly" |
| *value* | The value to set |

**Returns**

The configuration item attribute as string

**Exceptions**

| | |
|---|---|
| *ConfigItemNotFound* | if the item does not exist in the category ConfigItemAttributeNotFound if the requested attribute does not exist for the found item. |

#### 4.24.2.35 setItemDisplayName()

```
bool ConfigCategory::setItemDisplayName (
            const std::string & name,
            const std::string & displayName )
```

Set the display name of an item.

**Parameters**

| | |
|---|---|
| *name* | The item name in the category |
| *displayName* | The display name to set |

**Returns**

true if the item was found

#### 4.24.2.36 setValue()

```
bool ConfigCategory::setValue (
            const std::string & name,
            const std::string & value )
```

Update the value of the configuration category item.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration item to update |
| *value* | New value of the configuration item |

**Returns**

bool Whether update succeeded

#### 4.24.2.37 to_string()

```
std::string ConfigCategory::to_string (
            const rapidjson::Value & v ) const
```

Convert a RapidJSON value to a string.

**Parameters**

| | |
|---|---|
| *v* | The RapidJSON value |

#### 4.24.2.38 toJSON()

```
string ConfigCategory::toJSON (
            const bool full = false ) const
```

Return JSON string of all category components.

**Parameters**

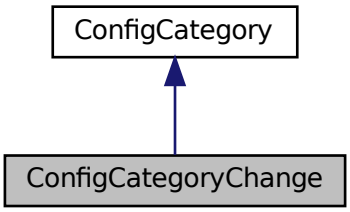
| | |
|---|---|
| *full* | false is the deafult, true evaluates all the members of the CategoryItems |

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

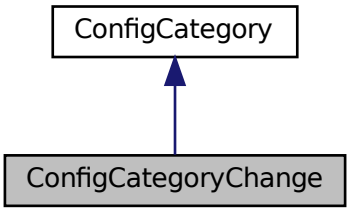
- C/common/include/config_category.h
- C/common/config_category.cpp

## 4.25 ConfigCategoryChange Class Reference

Inheritance diagram for ConfigCategoryChange:




Collaboration diagram for ConfigCategoryChange:




### Public Member Functions

- ConfigCategoryChange (const std::string &json)

  *Configuration Category constructor.*

### Additional Inherited Members

### 4.25.1 Constructor & Destructor Documentation

#### 4.25.1.1 ConfigCategoryChange()

```
ConfigCategoryChange::ConfigCategoryChange (
            const std::string & json )
```

Configuration Category constructor.

**Parameters**

| | |
|---|---|
| *name* | The name of the configuration category |
| *json* | JSON content of the configuration category |

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

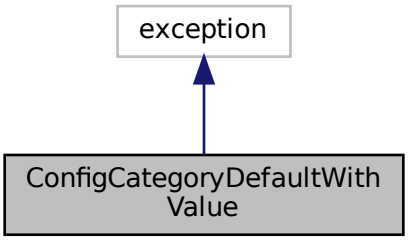
- C/common/include/config_category.h
- C/common/config_category.cpp

## 4.26 ConfigCategoryDefaultWithValue Class Reference

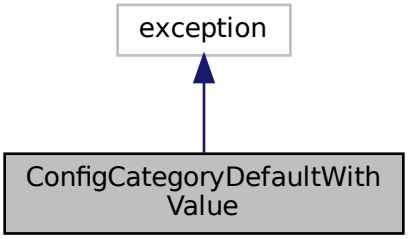
ConfigCategoryDefaultWithValue.

```
#include <configuration_manager.h>
```

Inheritance diagram for ConfigCategoryDefaultWithValue:



Collaboration diagram for ConfigCategoryDefaultWithValue:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.26.1 Detailed Description

ConfigCategoryDefaultWithValue.

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

- C/services/core/include/configuration_manager.h

## 4.27 ConfigCategoryDescription Class Reference

### Public Member Functions

- **ConfigCategoryDescription** (const std::string &name, const std::string &description)
- **ConfigCategoryDescription** (const std::string &name, const std::string &displayName, const std::string &description)
- std::string **getName** () const
- std::string **getDisplayName** () const
- std::string **getDescription** () const
- std::string toJSON () const

  *Return the JSON string of a ConfigCategoryDescription element.*

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

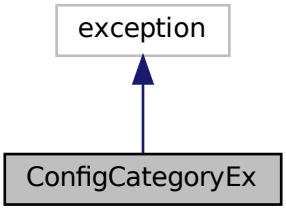
- C/common/include/config_category.h
- C/common/config_category.cpp

## 4.28 ConfigCategoryEx Class Reference

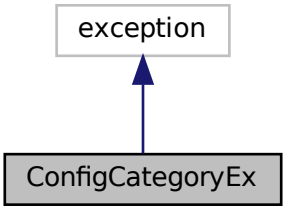
ConfigCategoryEx.

```
#include <configuration_manager.h>
```

Inheritance diagram for ConfigCategoryEx:




Collaboration diagram for ConfigCategoryEx:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.28.1 Detailed Description

ConfigCategoryEx.

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

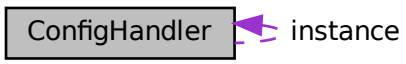
- C/services/core/include/configuration_manager.h

## 4.29 ConfigHandler Class Reference

Handler class within a service to manage configuration changes.

```
#include <config_handler.h>
```

Collaboration diagram for ConfigHandler:




### Public Member Functions

- void configChange (const std::string &category, const std::string &config)

  *Handle a callback from the core to propagate a configuration category change and propagate that to all the local ServiceHandlers that have registered for it.*
- void configChildCreate (const std::string &parent_category, const std::string &child_category, const std::string &config)

  *Handle a callback from the core to handle the creation of a child category.*
- void configChildDelete (const std::string &parent_category, const std::string &child_category)

  *Handle a callback from the core to handle the deletion of a child category.*
- void registerCategory (ServiceHandler *handler, const std::string &category)

  *Register a service handler for a given configuration category.*
- void registerCategoryChild (ServiceHandler *handler, const std::string &category)

  *Register a service handler for a given configuration category when a children category is changed.*
- void unregisterCategory (ServiceHandler *handler, const std::string &category)

  *Unregister a configuration category from the ConfigHandler for a particular registered ServiceHandler class.*

### Static Public Member Functions

- static ConfigHandler * getInstance (ManagementClient *)

  *ConfigHandler Singleton implementation.*

### Static Public Attributes

- static ConfigHandler * **instance** = 0

### 4.29.1 Detailed Description

Handler class within a service to manage configuration changes.

## 4.29.2 Member Function Documentation

### 4.29.2.1 configChange()

```
void ConfigHandler::configChange (
            const std::string & category,
            const std::string & config )
```

Handle a callback from the core to propagate a configuration category change and propagate that to all the local ServiceHandlers that have registered for it.

**Parameters**

| | |
|---|---|
| *category* | The name of the category that has changed |
| *config* | The configuration category itself |

### 4.29.2.2 configChildCreate()

```
void ConfigHandler::configChildCreate (
            const std::string & parent_category,
            const std::string & child_category,
            const std::string & config )
```

Handle a callback from the core to handle the creation of a child category.

**Parameters**

| | |
|---|---|
| *parent_category* | The parent category of the child |
| *child_category* | The name of the category that has created |
| *config* | Configuration of the child category |

### 4.29.2.3 configChildDelete()

```
void ConfigHandler::configChildDelete (
            const std::string & parent_category,
            const std::string & child_category )
```

Handle a callback from the core to handle the deletion of a child category.

**Parameters**

| | |
|---|---|
| *parent_category* | The parent category of the child |
| *child_category* | The name of the category that has created |

#### 4.29.2.4 registerCategory()

```
void ConfigHandler::registerCategory (
            ServiceHandler * handler,
            const std::string & category )
```

Register a service handler for a given configuration category.

**Parameters**

| | |
|---|---|
| *handler* | The service handler to call |
| *category* | The configuration category to register |

#### 4.29.2.5 registerCategoryChild()

```
void ConfigHandler::registerCategoryChild (
            ServiceHandler * handler,
            const std::string & category )
```

Register a service handler for a given configuration category when a children category is changed.

**Parameters**

| | |
|---|---|
| *handler* | The service handler to call |
| *category* | The configuration category to register |

#### 4.29.2.6 unregisterCategory()

```
void ConfigHandler::unregisterCategory (
            ServiceHandler * handler,
            const std::string & category )
```

Unregister a configuration category from the ConfigHandler for a particular registered ServiceHandler class.

**Parameters**

| | |
|---|---|
| *handler* | The configuration handler we would call |
| *category* | The category to remove. |

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

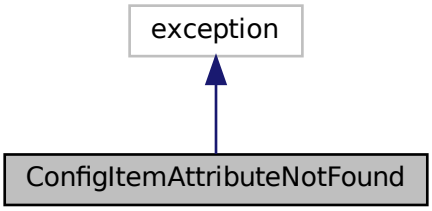
- C/services/common/include/config_handler.h
- C/services/common/config_handler.cpp

## 4.30 ConfigItemAttributeNotFound Class Reference

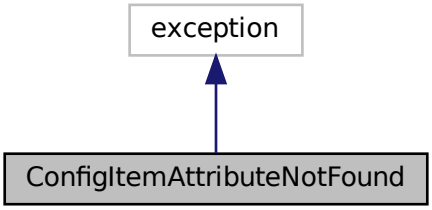
This exception must be raised when a requested item attribute does not exist.

```
#include <config_category.h>
```

Inheritance diagram for ConfigItemAttributeNotFound:




Collaboration diagram for ConfigItemAttributeNotFound:




### Public Member Functions

- virtual const char * **what** () const throw ()

### 4.30.1 Detailed Description

This exception must be raised when a requested item attribute does not exist.

Supported item attributes: "order", "readonly", "file".

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

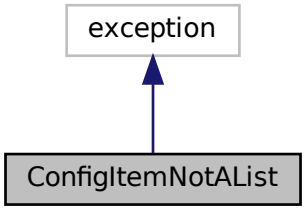
- C/common/include/config_category.h

## 4.31 ConfigItemNotAList Class Reference

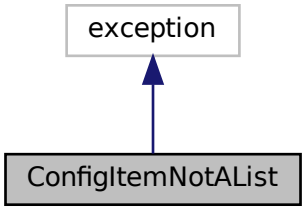
An attempt has been made to access a configuration item as a list when the item is not of type list.

```
#include <config_category.h>
```

Inheritance diagram for ConfigItemNotAList:




Collaboration diagram for ConfigItemNotAList:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.31.1 Detailed Description

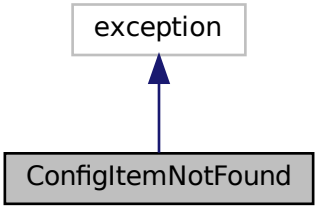
An attempt has been made to access a configuration item as a list when the item is not of type list.

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

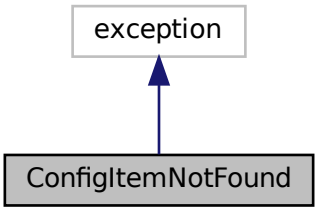
- C/common/include/config_category.h

## 4.32 ConfigItemNotFound Class Reference

Inheritance diagram for ConfigItemNotFound:




Collaboration diagram for ConfigItemNotFound:




### Public Member Functions

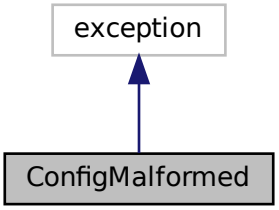
- virtual const char ∗ **what** () const throw ()

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

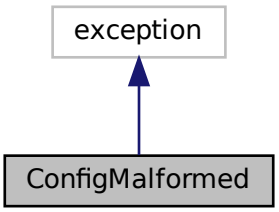
- C/common/include/config_category.h

## 4.33 ConfigMalformed Class Reference

Inheritance diagram for ConfigMalformed:




Collaboration diagram for ConfigMalformed:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

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

- C/common/include/config_category.h

## 4.34 ConfigurationManager Class Reference

### Public Member Functions

- ConfigCategories getAllCategoryNames () const

  *Return all Fledge categories from storage layer.*
- ConfigCategory getCategoryAllItems (const std::string &categoryName) const

*Return all the items of a specific category from the storage layer.*

- ConfigCategory createCategory (const std::string &categoryName, const std::string &categoryDescription, const std::string &categoryItems, bool keepOriginalItems=false) const

  *Create or update a new category.*
- std::string getCategoryItem (const std::string &categoryName, const std::string &itemName) const

  *Get a given item within a given category.*
- bool setCategoryItemValue (const std::string &categoryName, const std::string &itemName, const std::string &newValue) const

  *Set the "value" entry of a given item within a given category.*
- std::string addChildCategory (const std::string &parentCategoryName, const std::string &childCategories) const

  *Add child categories under a given (parent) category.*
- ConfigCategories getChildCategories (const std::string &parentCategoryName) const

  *Get all the child categories of a given category name.*
- std::string deleteChildCategory (const std::string &parentCategoryName, const std::string &childCategory) const

  *Remove the link between a child category and its parent.*
- std::string deleteCategoryItemValue (const std::string &categoryName, const std::string &itemName) const

  *Unset the category item value.*
- ConfigCategories deleteCategory (const std::string &categoryName) const

  *Delete a category from database.*
- std::string getCategoryItemValue (const std::string &categoryName, const std::string &itemName) const

  *Get the value of a given item within a given category.*

## Static Public Member Functions

- static ConfigurationManager * getInstance (const std::string &, short unsigned int)

  *Return the singleton instance of the configuration manager.*

## 4.34.1 Member Function Documentation

### 4.34.1.1 addChildCategory()

```
string ConfigurationManager::addChildCategory (
            const std::string & parentCategoryName,
            const std::string & childCategories ) const
```

Add child categories under a given (parent) category.

**Parameters**

| | |
|---|---|
| *parentCategoryName* | The parent category name |
| *childCategories* | The child categories list (JSON array) |

**Returns**

The JSON string with all (old and new) child categories of the parent category name

**Exceptions**

| | |
|---|---|
| *ChildCategoriesEx* | exception |
| *ExistingChildCategories* | exception @thow NoSuchCategory exception |

Do the insert: we don't check for failed result as we checked parent/child presence above

#### 4.34.1.2 createCategory()

```
ConfigCategory ConfigurationManager::createCategory (
            const std::string & categoryName,
            const std::string & categoryDescription,
            const std::string & categoryItems,
            bool keepOriginalItems = false ) const
```

Create or update a new category.

**Parameters**

| | |
|---|---|
| *categoryName* | The category name |
| *categoryDescription* | The category description |
| *categoryItems* | The category items |
| *keepOriginalItems* | Keep stored iterms or replace them |

**Returns**

The ConfigCategory object with "value" and "default" of the new category added or the merged configuration of the updated confguration.

**Exceptions**

| | |
|---|---|
| *CategoryDetailsEx* | exception |
| *ConfigCategoryEx* | exception |
| *ConfigCategoryDefaultWithValue* | exception |

Merge input data with stored data: stored configuration items are merged or replaced accordingly to keepOriginal↩ Items parameter value.

Items "value" are preserved for items being updated, only "default" values are overwritten.

#### 4.34.1.3 deleteCategory()

```
ConfigCategories ConfigurationManager::deleteCategory (
            const std::string & categoryName ) const
```

Delete a category from database.

Also remove the link between a child category and its parent.

**Parameters**

| | |
|---|---|
| *categoryName* | The category being deleted |

**Returns**

The remaining config categories as object

**Exceptions**

| | |
|---|---|
| *NoSuchCategory* | exception |
| *ConfigCategoryEx* | exception |

#### 4.34.1.4 deleteCategoryItemValue()

```
string ConfigurationManager::deleteCategoryItemValue (
            const std::string & categoryName,
            const std::string & itemName ) const
```

Unset the category item value.

**Parameters**

| | |
|---|---|
| *categoryName* | The category name |
| *itemName* | The item name |

**Returns**

JSON string of category item

**Exceptions**

| | |
|---|---|
| *ConfigCategoryEx* | exception |
| *NoSuchCategoryItem* | exception |

#### 4.34.1.5 deleteChildCategory()

```
string ConfigurationManager::deleteChildCategory (
            const std::string & parentCategoryName,
            const std::string & childCategory ) const
```

Remove the link between a child category and its parent.

The child becomes a root category when the link is broken. Note the child category still exists after this call is made.

**Parameters**

| | |
|---|---|
| *parentCategoryName* | The parennt category |
| *childCategory* | The child category to remove |

**Returns**

JSON array string with remaining child categories

**Exceptions**

| | |
|---|---|
| *ChildCategoriesEx* | exception |

#### 4.34.1.6 getAllCategoryNames()

```
ConfigCategories ConfigurationManager::getAllCategoryNames ( ) const
```

Return all Fledge categories from storage layer.

**Returns**

ConfigCategories class object with key and description for all found categories.

**Exceptions**

| | |
|---|---|
| *CategoryDetailsEx* | exception |

#### 4.34.1.7 getCategoryAllItems()

```
ConfigCategory ConfigurationManager::getCategoryAllItems (
            const std::string & categoryName ) const
```

Return all the items of a specific category from the storage layer.

**Parameters**

| | |
|---|---|
| *categoryName* | The specified category name |

**Returns**

ConfigCategory calss object with all category items

**Exceptions**

| | |
|---|---|
| *NoSuchCategory* | exception |
| *ConfigCategoryEx* | exception |
| *CategoryDetailsEx* | exception |

#### 4.34.1.8 getCategoryItem()

```
string ConfigurationManager::getCategoryItem (
            const std::string & categoryName,
            const std::string & itemName ) const
```

Get a given item within a given category.

**Parameters**

| | |
|---|---|
| *categoryName* | The given category |
| *itemName* | The given item |

**Returns**

JSON string with item details

#### 4.34.1.9 getCategoryItemValue()

```
string ConfigurationManager::getCategoryItemValue (
            const std::string & categoryName,
            const std::string & itemName ) const
```

Get the value of a given item within a given category.

**Parameters**

| | |
|---|---|
| *categoryName* | The given category |
| *itemName* | The given item |

**Returns**

string with item value

**Exceptions**

| | |
|---|---|
| *NoSuchCategoryItemValue* | exception |

#### 4.34.1.10 getChildCategories()

```
ConfigCategories ConfigurationManager::getChildCategories (
            const std::string & parentCategoryName ) const
```

Get all the child categories of a given category name.

**Parameters**

| | |
|---|---|
| *parentCategoryName* | The given category name |

**Returns**

A ConfigCategories object with child categories (name and description)

**Exceptions**

| | |
|---|---|
| *ChildCategoriesEx* | exception |

For each element fetch then category description and add the entry to ConfigCategories result

#### 4.34.1.11 getInstance()

```
ConfigurationManager * ConfigurationManager::getInstance (
            const std::string & ,
            short unsigned int  )  [static]
```

Return the singleton instance of the configuration manager.

**Parameters**

| | |
|---|---|
| *host* | Storage layer TCP address |
| *port* | Storage layer TCP port |

**Returns**

The configuration manager class instance

#### 4.34.1.12 setCategoryItemValue()

```
bool ConfigurationManager::setCategoryItemValue (
            const std::string & categoryName,
            const std::string & itemName,
            const std::string & newValue ) const
```

Set the "value" entry of a given item within a given category.

**Parameters**

| | |
|---|---|
| *categoryName* | The given category |
| *itemName* | The given item |
| *newValue* | The "value" entry to set |

**Returns**

True on success. False on DB update error or storage layer exception

**Exceptions**

| | |
|---|---|
| *NoSuchCategoryItem* | exception if categoryName/itemName doesn't exist |

Check whether newValue is the same as currentValue NOTE: Does it work if newValue represents JSON object istead of a simple value?

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

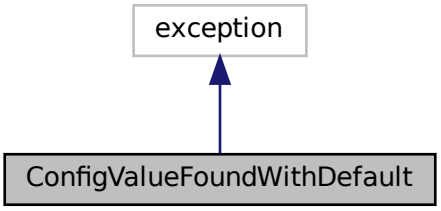
- C/services/core/include/configuration_manager.h
- C/services/core/configuration_manager.cpp

## 4.35 ConfigValueFoundWithDefault Class Reference

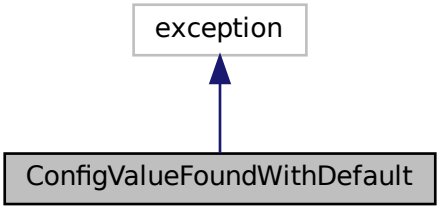
This exception must be raised when at least one of the JSON items of a new being created category have both "value" and "default" fields.

```
#include <config_category.h>
```

Inheritance diagram for ConfigValueFoundWithDefault:



Collaboration diagram for ConfigValueFoundWithDefault:




### Public Member Functions

- **ConfigValueFoundWithDefault** (const std::string &item)
- virtual const char ∗ **what** () const throw ()

### 4.35.1 Detailed Description

This exception must be raised when at least one of the JSON items of a new being created category have both "value" and "default" fields.

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

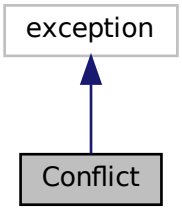
- C/common/include/config_category.h

## 4.36 Conflict Class Reference

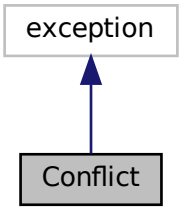
Conflict exception.

```
#include <http_sender.h>
```

Inheritance diagram for Conflict:



Collaboration diagram for Conflict:




### Public Member Functions

- **Conflict** (const std::string &serverReply)
- virtual const char ∗ **what** () const throw ()

### 4.36.1 Detailed Description

Conflict exception.

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

- C/plugins/common/include/http_sender.h

## 4.37 Connection Class Reference

### Public Member Functions

- Connection ()

  *Create a database connection.*
- ∼Connection ()

  *Destructor for the database connection.*
- bool **retrieve** (const std::string &schema, const std::string &table, const std::string &condition, std::string &resultSet)
- bool **retrieveReadings** (const std::string &condition, std::string &resultSet)
- int insert (const std::string &table, const std::string &data)

  *Insert data into a table.*
- int **update** (const std::string &table, const std::string &data)
- int **deleteRows** (const std::string &table, const std::string &condition)
- int appendReadings (const char ∗readings)

  *Append a set of readings to the readings table.*
- bool fetchReadings (unsigned long id, unsigned int blksize, std::string &resultSet)

  *Fetch a block of readings from the reading table.*
- unsigned int purgeReadings (unsigned long age, unsigned int flags, unsigned long sent, std::string &results)

*Purge readings from the reading table.*

- unsigned int purgeReadingsByRows (unsigned long rowcount, unsigned int flags, unsigned long sent, std↩ ::string &results)

  *Purge readings from the reading table leaving a number of rows equal to the parameter rows.*
- unsigned long purgeOperation (const char ∗sql, const char ∗logSection, const char ∗phase, bool retrieve)

  *Execute a SQL command for the purge task.*
- long **tableSize** (const std::string &table)
- void setTrace (bool flag)

  *Enable or disable the tracing of SQL statements.*
- int **create_table_snapshot** (const std::string &table, const std::string &id)
- int **load_table_snapshot** (const std::string &table, const std::string &id)
- int **delete_table_snapshot** (const std::string &table, const std::string &id)
- bool **get_table_snapshots** (const std::string &table, std::string &resultSet)
- bool **aggregateQuery** (const rapidjson::Value &payload, std::string &resultSet)
- int create_schema (const std::string &payload)

  *Create schema of tables.*
- bool findSchemaFromDB (const std::string &service, const std::string &name, std::string &resultSet)

  *Find existing payload schema from the DB fledge.service_schema table.*
- unsigned int **purgeReadingsAsset** (const std::string &asset)
- void **setMaxReadingRows** (long rows)
- bool parseDatabaseStorageSchema (int &version, const std::string &res, std::unordered_map< std::string, std::unordered_set< columnRec, columnRecHasher, columnRecComparator > > &tableColumnMap, std↩ ::unordered_map< std::string, std::vector< std::string > > &tableIndexMap, bool &schemaCreationRequest)

  *This function parses the fledge.service_schema table payload retrieved in and outputs a set of data structures containg the information about the tables and their columns and indexes.*
- Connection (ConnectionManager ∗manager)

  *Create a SQLite3 database connection.*
- bool createSchema (const std::string &schema)

  *Create schema and populate with tables and indexes as defined in the JSON schema definition.*
- bool **retrieve** (const std::string &schema, const std::string &table, const std::string &condition, std::string &resultSet)
- int insert (const std::string &schema, const std::string &table, const std::string &data)

  *Insert data into a table.*
- int **update** (const std::string &schema, const std::string &table, const std::string &data)
- int **deleteRows** (const std::string &schema, const std::string &table, const std::string &condition)
- int **create_table_snapshot** (const std::string &table, const std::string &id)
- int **load_table_snapshot** (const std::string &table, const std::string &id)
- int **delete_table_snapshot** (const std::string &table, const std::string &id)
- bool **get_table_snapshots** (const std::string &table, std::string &resultSet)
- int **appendReadings** (const char ∗readings)
- int readingStream (ReadingStream ∗∗readings, bool commit)

  *Append a stream of readings to SQLite db.*
- bool **fetchReadings** (unsigned long id, unsigned int blksize, std::string &resultSet)
- bool **retrieveReadings** (const std::string &condition, std::string &resultSet)
- unsigned int **purgeReadings** (unsigned long age, unsigned int flags, unsigned long sent, std::string &results)
- unsigned int **purgeReadingsByRows** (unsigned long rowcount, unsigned int flags, unsigned long sent, std::string &results)
- long **tableSize** (const std::string &table)
- void **setTrace** (bool)
- bool **formatDate** (char ∗formatted_date, size_t formatted_date_size, const char ∗date)
- bool **aggregateQuery** (const rapidjson::Value &payload, std::string &resultSet)
- bool **getNow** (std::string &Now)
- sqlite3 ∗ **getDbHandle** ()

- void setUsedDbId (int dbId)

  *Append a set of readings to the readings table.*
- void shutdownAppendReadings ()

  *Wait until all the threads executing the appendReadings are shutted down.*
- unsigned int **purgeReadingsAsset** (const std::string &asset)
- bool vacuum ()

  *Execute a SQLite VACUUM command on the database.*
- bool **supportsReadings** ()
- bool **createSchema** (const std::string &schema)
- bool **retrieve** (const std::string &schema, const std::string &table, const std::string &condition, std::string &resultSet)
- int **insert** (const std::string &schema, const std::string &table, const std::string &data)
- int **update** (const std::string &schema, const std::string &table, const std::string &data)
- int **deleteRows** (const std::string &schema, const std::string &table, const std::string &condition)
- int **create_table_snapshot** (const std::string &table, const std::string &id)
- int **load_table_snapshot** (const std::string &table, const std::string &id)
- int **delete_table_snapshot** (const std::string &table, const std::string &id)
- bool **get_table_snapshots** (const std::string &table, std::string &resultSet)
- int **appendReadings** (const char ∗readings)
- int **readingStream** (ReadingStream ∗∗readings, bool commit)
- bool **fetchReadings** (unsigned long id, unsigned int blksize, std::string &resultSet)
- bool **retrieveReadings** (const std::string &condition, std::string &resultSet)
- unsigned int **purgeReadings** (unsigned long age, unsigned int flags, unsigned long sent, std::string &results)
- unsigned int **purgeReadingsByRows** (unsigned long rowcount, unsigned int flags, unsigned long sent, std::string &results)
- long **tableSize** (const std::string &table)
- void **setTrace** (bool)
- bool **formatDate** (char ∗formatted_date, size_t formatted_date_size, const char ∗date)
- bool **aggregateQuery** (const rapidjson::Value &payload, std::string &resultSet)
- bool **getNow** (std::string &Now)
- unsigned int **purgeReadingsAsset** (const std::string &asset)
- bool **vacuum** ()
- void **setPurgeBlockSize** (unsigned long purgeBlockSize)

## Static Public Member Functions

- static bool formatDate (char ∗formatted_date, size_t formatted_date_size, const char ∗date)

  *Format a date to a fixed format with milliseconds, microseconds and timezone expressed, examples :*

## 4.37.1 Constructor & Destructor Documentation

### 4.37.1.1 Connection() [1/2]

```
Connection::Connection ( )
```

Create a database connection.

Create a SQLite3 database connection. Make a connection to the database and check backend connection was successfully made Note: we assume the database already exists, so the flag SQLITE_OPEN_CREATE is not added in sqlite3_open_v2 call

#### 4.37.1.2 ~Connection()

```
Connection::~Connection ( )
```

Destructor for the database connection.

Close the connection to Postgres

Close the connection to SQLite3 db

#### 4.37.1.3 Connection() [2/2]

```
Connection::Connection (
            ConnectionManager * manager )
```

Create a SQLite3 database connection.

Make a connection to the database and check backend connection was successfully made Note: we assume the database already exists, so the flag SQLITE_OPEN_CREATE is not added in sqlite3_open_v2 call

### 4.37.2 Member Function Documentation

#### 4.37.2.1 create_schema()

```
int Connection::create_schema (
            const std::string & payload )
```

Create schema of tables.

**Parameters**

| | |
|---|---|
| *payload* | The payload containing information about schema of tables to create |

**Returns**

true if the tables can be crated successfully

#### 4.37.2.2 createSchema()

```
bool Connection::createSchema (
            const std::string & schema )
```

Create schema and populate with tables and indexes as defined in the JSON schema definition.

**Parameters**

| | |
|---|---|
| *schema* | The schema defintion as a JSON document containing information about schema of tables to create |

**Returns**

true if the schema was created

#### 4.37.2.3 fetchReadings()

```
bool Connection::fetchReadings (
            unsigned long id,
            unsigned int blksize,
            std::string & resultSet )
```

Fetch a block of readings from the reading table.

Fetch a block of readings from the reading table It might not work with SQLite 3.

Fetch, used by the north side, returns timestamp in UTC.

NOTE : it expects to handle a date having a fixed format with milliseconds, microseconds and timezone expressed, like for example :

2019-01-11 15:45:01.123456+01:00

#### 4.37.2.4 findSchemaFromDB()

```
bool Connection::findSchemaFromDB (
            const std::string & service,
            const std::string & schema,
            std::string & resultSet )
```

Find existing payload schema from the DB fledge.service_schema table.

**Parameters**

| | |
|---|---|
| *service* | The string containing service name |
| *name* | The string containing schema name |

**Returns**

resultSet string containing the output of the sql query executed

#### 4.37.2.5 formatDate()

```
bool Connection::formatDate (
            char * formatted_date,
            size_t buffer_size,
            const char * date )  [static]
```

Format a date to a fixed format with milliseconds, microseconds and timezone expressed, examples :

case - formatted |2019-01-01 10:01:01.000000+00:00| date |2019-01-01 10:01:01| case - formatted |2019-02-01 10:02:01.000000+00:00| date |2019-02-01 10:02:01.0| case - formatted |2019-02-02 10:02:02.841000+00:00| date |2019-02-02 10:02:02.841| case - formatted |2019-02-03 10:02:03.123456+00:00| date |2019-02-03 10:02:03.↩ 123456| case - formatted |2019-03-01 10:03:01.100000+00:00| date |2019-03-01 10:03:01.1+00:00| case - formatted |2019-03-02 10:03:02.123000+00:00| date |2019-03-02 10:03:02.123+00:00| case - formatted |2019-03-03 10:03:03.123456+00:00| date |2019-03-03 10:03:03.123456+00:00| case - formatted |2019-03-04 10:03:04.↩ 123456+01:00| date |2019-03-04 10:03:04.123456+01:00| case - formatted |2019-03-05 10:03:05.123456-01:00| date |2019-03-05 10:03:05.123456-01:00| case - formatted |2019-03-04 10:03:04.123456+02:30| date |2019-03-04 10:03:04.123456+02:30| case - formatted |2019-03-05 10:03:05.123456-02:30| date |2019-03-05 10:03:05.123456-02:30|

**Parameters**

| | |
|---|---|
| *out* | false if the date is invalid |

#### 4.37.2.6 parseDatabaseStorageSchema()

```
bool Connection::parseDatabaseStorageSchema (
            int & version,
            const std::string & res,
            std::unordered_map< std::string, std::unordered_set< columnRec, columnRecHasher,
columnRecComparator > > & tableColumnMap,
            std::unordered_map< std::string, std::vector< std::string > > & tableIndexMap,
            bool & schemaCreationRequest )
```

This function parses the fledge.service_schema table payload retrieved in and outputs a set of data structures containg the information about the tables and their columns and indexes.

**Parameters**

| | | |
|---|---|---|
| out | *version* | version retrieved form payload |
| in | *res* | output containing payload information |
| out | *tableColumnMap* | map[tablename —> set of columns] |
| out | *tableIndexMap* | map[tablename —> indexes] where each index is a comma separated string of columns |
| | *[ouy]* | schemaCreationRequest which is like this is first schema creation request or schema already exist in the DB |

**Returns**

true if parsing is successful else false

#### 4.37.2.7 purgeReadingsByRows()

```
unsigned int Connection::purgeReadingsByRows (
            unsigned long rowcount,
            unsigned int flags,
            unsigned long sent,
            std::string & results )
```

Purge readings from the reading table leaving a number of rows equal to the parameter rows.

Purge readings from the reading table.

#### 4.37.2.8 readingStream()

```
int Connection::readingStream (
            ReadingStream ** readings,
            bool commit )
```

Append a stream of readings to SQLite db.

**Parameters**

| | |
|---|---|
| *readings* | readings to store into the SQLite db |
| *commit* | if true a database commit is executed and a new transaction will be opened at the next execution |

TODO: the current code should be adapted to use the multi databases/tables implementation

**Parameters**

| | |
|---|---|
| *readings* | readings to store into the SQLite db |
| *commit* | if true a database commit is executed and a new transaction will be opened at the next execution |

#### 4.37.2.9 setTrace()

```
void Connection::setTrace (
            bool flag )  [inline]
```

Enable or disable the tracing of SQL statements.

**Parameters**

| | |
|---|---|
| *flag* | Desired state of the SQL trace flag |

#### 4.37.2.10 vacuum()

```
bool Connection::vacuum ( )
```

Execute a SQLite VACUUM command on the database.

Add a vacuum funtion, this is not needed for SQLite In Memory, but is here to satisfy the interface requirement.

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

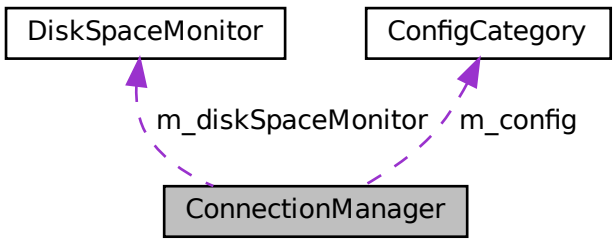
- C/plugins/storage/postgres/include/connection.h
- C/plugins/storage/postgres/connection.cpp
- C/plugins/storage/sqlite/common/readings.cpp

## 4.38 ConnectionManager Class Reference

Singleton class to manage Postgres connection pool.

```
#include <connection_manager.h>
```

Collaboration diagram for ConnectionManager:



## Public Member Functions

- void growPool (unsigned int)

  *Grow the connection pool by the number of connections specified.*
- unsigned int shrinkPool (unsigned int)

  *Attempt to shrink the number of connections in the idle pool.*
- Connection ∗ allocate ()

  *Allocate a connection from the idle pool.*
- void release (Connection ∗)

  *Release a connection back to the idle pool for reallocation.*
- void shutdown ()

  *Called at shutdown.*
- void setError (const char ∗, const char ∗, bool)

  *Set the last error information for a plugin.*
- PLUGIN_ERROR ∗ **getError** ()
- void **setMaxReadingRows** (long rows)
- void **growPool** (unsigned int)
- unsigned int **shrinkPool** (unsigned int)
- Connection ∗ **allocate** ()
- bool attachNewDb (std::string &path, std::string &alias)

  *Attach a database to all the connections, idle and inuse.*
- bool attachRequestNewDb (int newDbId, sqlite3 ∗dbHandle)

  *Adds to all the connections a request to attach a database.*
- bool detachNewDb (std::string &alias)

  *Detach a database from all the connections.*
- void **release** (Connection ∗)
- void **shutdown** ()
- void **setError** (const char ∗, const char ∗, bool)
- PLUGIN_ERROR ∗ **getError** ()
- void background ()

  *Background thread used to execute periodic tasks and oversee the database activity.*
- void **setVacuumInterval** (long hours)
- bool allowMoreDatabases ()

  *Determine if we can allow another database to be created and attached to all the connections.*
- void **setConfiguration** (ConfigCategory ∗category)
- std::string getDBConfiguration ()

  *Return the pragma configuration for the database.*
- void **growPool** (unsigned int)
- unsigned int **shrinkPool** (unsigned int)
- Connection ∗ **allocate** ()
- void **release** (Connection ∗)
- void **shutdown** ()
- void **setError** (const char ∗, const char ∗, bool)
- PLUGIN_ERROR ∗ **getError** ()
- void **background** ()
- void **setVacuumInterval** (long hours)
- void **setPersist** (bool persist, const std::string &filename="")
- bool **persist** ()
- std::string **filename** ()
- void setPurgeBlockSize (unsigned long purgeBlockSize)

  *Set the purge block size in each of the connections.*

## Static Public Member Functions

- static ConnectionManager ∗ getInstance ()

  *Return the singleton instance of the connection manager.*
- static ConnectionManager ∗ **getInstance** ()
- static ConnectionManager ∗ **getInstance** ()

## Protected Attributes

- bool **m_trace**
- bool **m_shutdown**
- std::thread ∗ **m_background**
- long **m_vacuumInterval**
- unsigned int **m_descriptorLimit**
- unsigned int **m_attachedDatabases**
- DiskSpaceMonitor ∗ **m_diskSpaceMonitor**
- ConfigCategory ∗ **m_config**
- bool **m_persist**
- std::string **m_filename**
- unsigned long **m_purgeBlockSize**

## 4.38.1 Detailed Description

Singleton class to manage Postgres connection pool.

Singleton class to manage SQLite3 connection pool.

## 4.38.2 Member Function Documentation

### 4.38.2.1 allocate()

```
Connection * ConnectionManager::allocate ( )
```

Allocate a connection from the idle pool.

If no connection is available add a new connection

### 4.38.2.2 allowMoreDatabases()

```
bool ConnectionManager::allowMoreDatabases ( )
```

Determine if we can allow another database to be created and attached to all the connections.

**Returns**

True if we can create another database.

### 4.38.2.3 attachNewDb()

```
bool ConnectionManager::attachNewDb (
            std::string & path,
            std::string & alias )
```

Attach a database to all the connections, idle and inuse.

**Parameters**

| | |
|---|---|
| *path* | - path of the database to attach |
| *alias* | - alias to be assigned to the attached database |

#### 4.38.2.4 attachRequestNewDb()

```
bool ConnectionManager::attachRequestNewDb (
            int newDbId,
            sqlite3 * dbHandle )
```

Adds to all the connections a request to attach a database.

**Parameters**

| | |
|---|---|
| *newDbId* | - database id to attach |
| *dbHandle* | - dbhandle for which the attach request should NOT be added |

- 

#### 4.38.2.5 background()

```
void ConnectionManager::background ( )
```

Background thread used to execute periodic tasks and oversee the database activity.

We will run the SQLite vacuum command periodically to allow space to be reclaimed

We will runt he SQLite vacuum command periodically to allow space to be reclaimed

#### 4.38.2.6 detachNewDb()

```
bool ConnectionManager::detachNewDb (
            std::string & alias )
```

Detach a database from all the connections.

#### 4.38.2.7 getInstance()

```
ConnectionManager * ConnectionManager::getInstance ( )  [static]
```

Return the singleton instance of the connection manager.

if none was created then create it.

#### 4.38.2.8 growPool()

```
void ConnectionManager::growPool (
            unsigned int delta )
```

Grow the connection pool by the number of connections specified.

**Parameters**

| | |
|---|---|
| *delta* | The number of connections to add to the pool |

#### 4.38.2.9 release()

```
void ConnectionManager::release (
            Connection * conn )
```

Release a connection back to the idle pool for reallocation.

**Parameters**

| | |
|---|---|
| *conn* | The connection to release. |

#### 4.38.2.10 setError()

```
void ConnectionManager::setError (
            const char * source,
            const char * description,
            bool retryable )
```

Set the last error information for a plugin.

**Parameters**

| | |
|---|---|
| *source* | The source of the error |
| *description* | The error description |
| *retryable* | Flag to determien if the error condition is transient |

#### 4.38.2.11 setPurgeBlockSize()

```
void ConnectionManager::setPurgeBlockSize (
            unsigned long purgeBlockSize )
```

Set the purge block size in each of the connections.

**Parameters**

| | |
|---|---|
| *purgeBlockSize* | The requested purgeBlockSize |

#### 4.38.2.12 shrinkPool()

```
unsigned int ConnectionManager::shrinkPool (
            unsigned int delta )
```

Attempt to shrink the number of connections in the idle pool.

**Parameters**

| | |
|---|---|
| *delta* | Number of connections to attempt to remove |

**Returns**

The number of connections removed.

#### 4.38.2.13 shutdown()

```
void ConnectionManager::shutdown ( )
```

Called at shutdown.

Shrink the idle pool, this will have the side effect of closing the connections to the database.

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

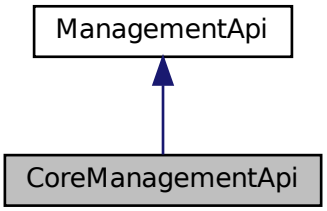
- C/plugins/storage/postgres/include/connection_manager.h
- C/plugins/storage/postgres/connection_manager.cpp

## 4.39 CoreManagementApi Class Reference

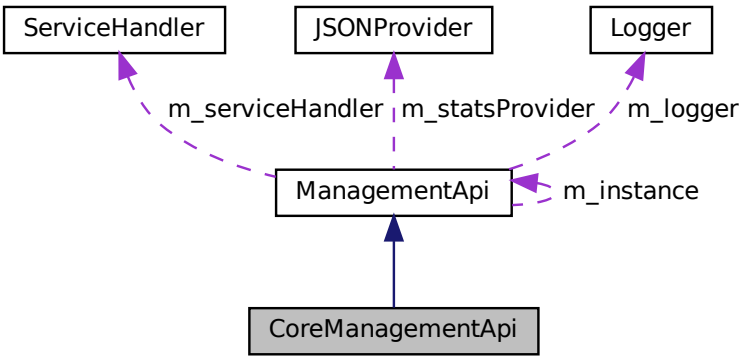
Management API server for a C++ microservice.

```
#include <core_management_api.h>
```

Inheritance diagram for CoreManagementApi:




Collaboration diagram for CoreManagementApi:




## Public Member Functions

- CoreManagementApi (const std::string &name, const unsigned short port)

  *Construct a microservices management API manager class.*
- void registerMicroService (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< Http↩ Server::Request > request)

  *Received a service registration request.*
- void unRegisterMicroService (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< Http↩ Server::Request > request)

  *Received a service unregister request.*
- void getAllCategories (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer↩ ::Request > request)

  *Received a GET /fledge/service/category.*
- void getCategory (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::↩ Request > request)

  *Received a GET /fledge/service/category/{categoryName}.*

- void getCategoryItem (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer↩ ::Request > request)

  *Received a GET /fledge/service/category/{categoryName}/{itemName].*
- void deleteCategoryItemValue (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::Request > request)

  *Received a DELETE /fledge/service/category/{categoryName}/{configItem}/value.*
- void setCategoryItemValue (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< Http↩ Server::Request > request)

  *Received PUT /fledge/service/category/{categoryName}/{configItem} Payload is {"value" : "some_data"} Send to client the JSON string of category item properties.*
- void deleteCategory (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::↩ Request > request)

  *Delete a config category Received DELETE /fledge/service/category/{categoryName} Send to client the JSON string of all remaining categories.*
- void deleteChildCategory (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< Http↩ Server::Request > request)

  *Delete child categories of a config category Received DELETE /fledge/service/category/{categoryName}/children/{child↩ Category} Send to client the JSON string of all remaining categories.*
- void createCategory (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::↩ Request > request)

  *Create a new configuration category Received POST /fledge/service/category.*
- void addChildCategory (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer↩ ::Request > request)

  *Add child categories to a given category name Received POST /fledge/service/category/{categoryName}/children.*
- void defaultResource (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer↩ ::Request > request)

  *Handle a bad URL endpoint call.*

## Static Public Member Functions

- static CoreManagementApi * getInstance ()

  *Return the singleton instance of the core management interface.*

## Additional Inherited Members

### 4.39.1 Detailed Description

Management API server for a C++ microservice.

### 4.39.2 Member Function Documentation

#### 4.39.2.1 addChildCategory()

```
void CoreManagementApi::addChildCategory (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Add child categories to a given category name Received POST /fledge/service/category/{categoryName}/children.

Send to client the JSON string with child categories

#### 4.39.2.2 createCategory()

```
void CoreManagementApi::createCategory (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Create a new configuration category Received POST /fledge/service/category.

Send to client the JSON string of new category's items

#### 4.39.2.3 getInstance()

```
CoreManagementApi * CoreManagementApi::getInstance ( )  [static]
```

Return the singleton instance of the core management interface.

Note if one has not been explicitly created then this will return 0.

#### 4.39.2.4 registerMicroService()

```
void CoreManagementApi::registerMicroService (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Received a service registration request.

Storage layer is registered Setup ConfigurationManager instance and URL entry points

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

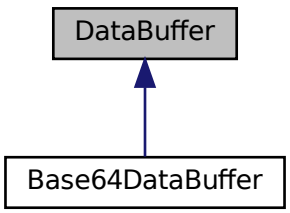
- C/services/core/include/core_management_api.h
- C/services/core/core_management_api.cpp

## 4.40 DataBuffer Class Reference

Buffer type for storage of arbitrary buffers of data within a datapoint.

```
#include <databuffer.h>
```

Inheritance diagram for DataBuffer:



## Public Member Functions

- DataBuffer (size_t itemSize, size_t len)

  *Buffer constructor.*
- DataBuffer (const DataBuffer &rhs)

  *DataBuffer copy constructor.*
- DataBuffer & **operator=** (const DataBuffer &rhs)
- ~DataBuffer ()

  *DataBuffer destructor.*
- void populate (void *src, int len)

  *Populate the contents of a DataBuffer.*
- size_t getItemSize ()

  *Return the size of each item in the buffer.*
- size_t getItemCount ()

  *Return the number of items in the buffer.*
- void * getData ()

  *Return a pointer to the raw data in the data buffer.*

## Protected Attributes

- size_t **m_itemSize**
- size_t **m_len**
- void * **m_data**

### 4.40.1 Detailed Description

Buffer type for storage of arbitrary buffers of data within a datapoint.

A DataBuffer is essentially a 1 dimensional array of a memory primitive of itemSize.

### 4.40.2 Constructor & Destructor Documentation

#### 4.40.2.1 DataBuffer() [1/2]

```
DataBuffer::DataBuffer (
            size_t itemSize,
            size_t len )
```

Buffer constructor.

**Parameters**

| | |
|---|---|
| *itemSize* | The size of each item in the buffer |
| *len* | The length of the buffer, i.e. how many items can it hold |

#### 4.40.2.2 DataBuffer() [2/2]

```
DataBuffer::DataBuffer (
            const DataBuffer & rhs )
```

DataBuffer copy constructor.

**Parameters**

| | |
|---|---|
| *rhs* | DataBuffer to copy |

### 4.40.3 Member Function Documentation

#### 4.40.3.1 populate()

```
void DataBuffer::populate (
            void * src,
            int len )
```

Populate the contents of a DataBuffer.

**Parameters**

| | |
|---|---|
| *src* | Source of the data |
| *len* | Number of bytes in the source to copy |

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

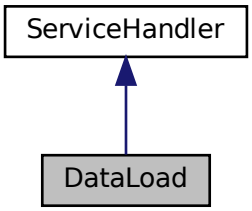
- C/common/include/databuffer.h
- C/common/databuffer.cpp

## 4.41 DataLoad Class Reference

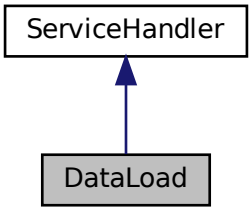
A class used in the North service to load data from the buffer.

```
#include <data_load.h>
```

Inheritance diagram for DataLoad:




Collaboration diagram for DataLoad:




## Public Member Functions

- DataLoad (const std::string &name, long streamId, StorageClient *storage)

  *DataLoad Constructor.*
- virtual ~DataLoad ()

  *DataLoad destructor.*
- void loadThread ()

  *The background thread that loads data from the database.*
- bool setDataSource (const std::string &source)

  *Set the source of data for the service.*
- void triggerRead (unsigned int blockSize)

  *Trigger the loading thread to read a block of data.*
- void updateLastSentId (unsigned long id)

  *Update the last sent ID for our stream.*
- void flushLastSentId ()

  *Flush the last sent Id to the storeage layer.*
- ReadingSet * fetchReadings (bool wait)

  *Fetch Readings.*
- void shutdown ()

*External call to shutdown the north service.*

- void restart ()

  *External call to restart the north service.*
- bool **isRunning** ()
- void configChange (const std::string &category, const std::string &newConfig)

  *Configuration change for one of the filters or to the pipeline.*
- void **configChildCreate** (const std::string &, const std::string &, const std::string &)
- void **configChildDelete** (const std::string &, const std::string &)
- unsigned long **getLastFetched** ()
- void **setBlockSize** (unsigned long blockSize)
- void **setStreamUpdate** (unsigned long streamUpdate)
- void **setPerfMonitor** (PerformanceMonitor *perfMonitor)
- const std::string & **getName** ()
- StorageClient * **getStorage** ()
- void **setPrefetchLimit** (unsigned int limit)
- bool **attachDebugger** ()
- void **detachDebugger** ()
- void **setDebuggerBuffer** (unsigned int size)
- std::string **getDebuggerBuffer** ()
- void **isolate** (bool isolate)
- bool **isolated** ()
- void **replayDebugger** ()
- void **suspendIngest** (bool suspend)
- bool **isSuspended** ()
- void **stepDebugger** (unsigned int steps)
- bool **willStep** ()

## Static Public Member Functions

- static void passToOnwardFilter (OUTPUT_HANDLE *outHandle, READINGSET *readings)

  *Pass the current readings set to the next filter in the pipeline.*
- static void pipelineEnd (OUTPUT_HANDLE *outHandle, READINGSET *readings)

  *Use the current readings (they have been filtered by all filters)*

### 4.41.1 Detailed Description

A class used in the North service to load data from the buffer.

This class is responsible for loading the reading from the storage service and buffering them ready for the egress thread to process them.

### 4.41.2 Constructor & Destructor Documentation

#### 4.41.2.1 DataLoad()

```
DataLoad::DataLoad (
            const std::string & name,
            long streamId,
            StorageClient * storage )
```

DataLoad Constructor.

Create and start the loading thread

#### 4.41.2.2 ~DataLoad()

```
DataLoad::~DataLoad ( )  [virtual]
```

DataLoad destructor.

Shutdown and wait for the loading thread

### 4.41.3 Member Function Documentation

#### 4.41.3.1 configChange()

```
void DataLoad::configChange (
            const std::string & category,
            const std::string & newConfig )  [virtual]
```

Configuration change for one of the filters or to the pipeline.

**Parameters**

| | |
|---|---|
| *category* | The name of the configuration category |
| *newConfig* | The new category contents |

The category that has changed is the one for the north service itself. The only items that concerns us here is the filter item that defines the filter pipeline and the data source. If the item is the filter pipeline we extract that item and check to see if it defines a pipeline that is different to the one we currently have.

If it is the filter pipeline we destroy the current pipeline and create a new one.

Implements ServiceHandler.

#### 4.41.3.2 fetchReadings()

```
ReadingSet * DataLoad::fetchReadings (
            bool wait )
```

Fetch Readings.

**Parameters**

| | |
|---|---|
| *wait* | Boolean to determine if the call should block the calling thread |

**Returns**

ReadingSet∗ Return a block of readings from the buffer

#### 4.41.3.3 passToOnwardFilter()

```
void DataLoad::passToOnwardFilter (
            OUTPUT_HANDLE * outHandle,
            READINGSET * readingSet )  [static]
```

Pass the current readings set to the next filter in the pipeline.

Note: This routine must be passed to all filters "plugin_init" except the last one

Static method

**Parameters**

| | |
|---|---|
| *outHandle* | Pointer to next filter |
| *readings* | Current readings set |

#### 4.41.3.4 pipelineEnd()

```
void DataLoad::pipelineEnd (
            OUTPUT_HANDLE * outHandle,
            READINGSET * readingSet )  [static]
```

Use the current readings (they have been filtered by all filters)

The assumption is that one of two things has happened.

1. The filtering has all been done in place. In which case the m_data vector is in the ReadingSet passed in here.
2. The filtering has created new ReadingSet in which case the reading vector must be copied into m_data from the ReadingSet.

Note: This routine must be passed to last filter "plugin_init" only

Static method

**Parameters**

| | |
|---|---|
| *outHandle* | Pointer to DataLoad class |
| *readingSet* | Filtered reading set being added to Ingest::m_data |

#### 4.41.3.5 setDataSource()

```
bool DataLoad::setDataSource (
            const std::string & source )
```

Set the source of data for the service.

**Parameters**

| | |
|---|---|
| *source* | The data source |

#### 4.41.3.6 triggerRead()

```
void DataLoad::triggerRead (
            unsigned int blockSize )
```

Trigger the loading thread to read a block of data.

This is called by any thread to request that data be added to the buffer ready for collection.

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

- C/services/north/include/data_load.h
- C/services/north/data_load.cpp

## 4.42 Datapoint Class Reference

Name and value pair used to represent a data value within an asset reading.

```
#include <datapoint.h>
```

## Public Member Functions

- Datapoint (const std::string &name, DatapointValue &value)

  *Construct with a data point value.*
- std::string toJSONProperty ()

  *Return asset reading data point as a JSON property that can be included within a JSON document.*
- const std::string getName () const

  *Return the Datapoint name.*
- void setName (std::string name)

  *Rename the datapoint.*
- const DatapointValue getData () const

  *Return Datapoint value.*
- DatapointValue & getData ()

  *Return reference to Datapoint value.*
- std::vector< Datapoint * > * parseJson (const std::string &json)

  *Parse a json string and generates a corresponding datapoint vector*
- std::vector< Datapoint * > * recursiveJson (const rapidjson::Value &document)

  *Recursive method to convert a JSON string to a datapoint.*

### 4.42.1 Detailed Description

Name and value pair used to represent a data value within an asset reading.

### 4.42.2 Member Function Documentation

#### 4.42.2.1 parseJson()

```
std::vector< Datapoint * > * Datapoint::parseJson (
            const std::string & json )
```

Parse a json string and generates a corresponding datapoint vector

Parsing a Json string.

**Parameters**

| | |
|---|---|
| *json* | : string json |

**Returns**

vector of datapoints

#### 4.42.2.2 recursiveJson()

```
std::vector< Datapoint * > * Datapoint::recursiveJson (
            const rapidjson::Value & document )
```

Recursive method to convert a JSON string to a datapoint.

**Parameters**

| | |
|---|---|
| *document* | : object rapidjon |

**Returns**

vector of datapoints

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

- C/common/include/datapoint.h
- C/common/datapoint.cpp

## 4.43 DatapointValue Class Reference

Class to hold an actual reading value.

```
#include <datapoint.h>
```

### Public Types

- enum **DatapointTag** {
  **T_STRING**, **T_INTEGER**, **T_FLOAT**, **T_FLOAT_ARRAY**,
  **T_DP_DICT**, **T_DP_LIST**, **T_IMAGE**, **T_DATABUFFER**,
  **T_2D_FLOAT_ARRAY** }
- typedef enum DatapointValue::DatapointTag **dataTagType**

### Public Member Functions

- DatapointValue (const std::string &value)

  *Construct with a string.*
- DatapointValue (const long value)

  *Construct with an integer value.*
- DatapointValue (const double value)

  *Construct with a floating point value.*
- DatapointValue (const std::vector< double > &values)

  *Construct with an array of floating point values.*
- DatapointValue (std::vector< Datapoint * > *&values, bool isDict)

  *Construct with an array of Datapoints.*
- DatapointValue (const DPImage &value)

  *Construct with an Image.*

- DatapointValue (const DataBuffer &value)

  *Construct with a DataBuffer.*
- DatapointValue (DPImage *value)

  *Construct with an Image Pointer, the image becomes owned by the datapointValue.*
- DatapointValue (DataBuffer *value)

  *Construct with a DataBuffer.*
- DatapointValue (const std::vector< std::vector< double > * > &values)

  *Construct with a 2 dimentional array of floating point values.*
- DatapointValue (const DatapointValue &obj)

  *Copy constructor.*
- DatapointValue & operator= (const DatapointValue &rhs)

  *Assignment Operator.*
- ~DatapointValue ()

  *Destructor.*
- void setValue (std::string value)

  *Set the value of a datapoint, this may also cause the type to be changed.*
- void setValue (long value)

  *Set the value of a datapoint, this may also cause the type to be changed.*
- void setValue (double value)

  *Set the value of a datapoint, this may also cause the type to be changed.*
- void setValue (const DPImage &value)

  *Set the value of a datapoint to be an image.*
- std::string toString () const

  *Return the value as a string.*
- std::string toStringValue () const

  *Return string value without trailing/leading quotes.*
- long toInt () const

  *Return long value.*
- double toDouble () const

  *Return double value.*
- dataTagType getType () const

  *Return the Tag type.*
- std::string **getTypeStr** () const
- std::vector< Datapoint * > *& getDpVec ()

  *Return array of datapoints.*
- std::vector< double > *& getDpArr ()

  *Return array of float.*
- std::vector< std::vector< double > * > *& getDp2DArr ()

  *Return 2D array of float.*
- DPImage * getImage ()

  *Return the Image.*
- DataBuffer * getDataBuffer ()

  *Return the DataBuffer.*

### 4.43.1 Detailed Description

Class to hold an actual reading value.

The class is simply a tagged union that also contains methods to return the value as a string for encoding in a JSON document.

## 4.43.2 Constructor & Destructor Documentation

### 4.43.2.1 ~DatapointValue()

```
DatapointValue::~DatapointValue ( )
```

Destructor.

DatapointValue class destructor.

## 4.43.3 Member Function Documentation

### 4.43.3.1 setValue() [1/4]

```
void DatapointValue::setValue (
            const DPImage & value )  [inline]
```

Set the value of a datapoint to be an image.

**Parameters**

| | |
|---|---|
| *value* | The image to set in the data point |

### 4.43.3.2 setValue() [2/4]

```
void DatapointValue::setValue (
            double value )  [inline]
```

Set the value of a datapoint, this may also cause the type to be changed.

**Parameters**

| | |
|---|---|
| *value* | A floating point value to set |

### 4.43.3.3 setValue() [3/4]

```
void DatapointValue::setValue (
            long value )  [inline]
```

Set the value of a datapoint, this may also cause the type to be changed.

**Parameters**

| | |
|---|---|
| *value* | An integer value to set |

#### 4.43.3.4 setValue() [4/4]

```
void DatapointValue::setValue (
            std::string value )  [inline]
```

Set the value of a datapoint, this may also cause the type to be changed.

**Parameters**

| | |
|---|---|
| *value* | An string value to set |

#### 4.43.3.5 toString()

```
std::string DatapointValue::toString ( ) const
```

Return the value as a string.

**Returns**

String representing the DatapointValue object

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

- C/common/include/datapoint.h
- C/common/datapoint.cpp

## 4.44 DataSender Class Reference

### Public Member Functions

- DataSender (NorthPlugin *plugin, DataLoad *loader, NorthService *north)

  *Constructor for the data sending class.*
- ~DataSender ()

  *Destructor for the data sender class.*
- void sendThread ()

  *The sending thread entry point.*
- void **updatePlugin** (NorthPlugin *plugin)

- void pause ()

  *Cause the data sender process to pause sending data until a corresponding release call is made.*
- void release ()

  *Release the paused data sender thread.*
- void **setPerfMonitor** (PerformanceMonitor *perfMonitor)
- bool **isRunning** ()
- void flushStatistics ()

  *Flush statistics to storage service.*
- bool isDryRun ()

  *Check status of dryrun flag.*
- void configChange ()

  *Notify the data sender that there has been a configuration change.*

## 4.44.1 Member Function Documentation

### 4.44.1.1 configChange()

```
void DataSender::configChange ( )
```

Notify the data sender that there has been a configuration change.

This is used to wake up early from the wait that is performed when we have failures to send. Changing the configuration may resolve the issue causing the send failure.

### 4.44.1.2 isDryRun()

```
bool DataSender::isDryRun ( )
```

Check status of dryrun flag.

**Returns**

True dryrun flag is true

### 4.44.1.3 pause()

```
void DataSender::pause ( )
```

Cause the data sender process to pause sending data until a corresponding release call is made.

This call does not block until release is called, but does block until the current send completes.

Called by external classes that want to prevent interaction with the north plugin.

#### 4.44.1.4 release()

```
void DataSender::release ( )
```

Release the paused data sender thread.

Called by external classes that want to release interaction with thew north plugin.

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

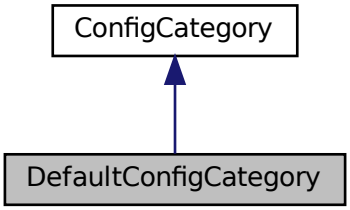
- C/services/north/include/data_sender.h
- C/services/north/data_send.cpp

## 4.45 DefaultConfigCategory Class Reference

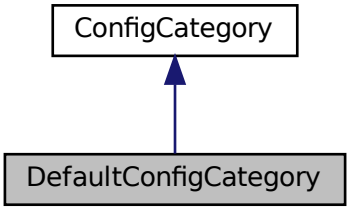
DefaultConfigCategory.

```
#include <config_category.h>
```

Inheritance diagram for DefaultConfigCategory:




Collaboration diagram for DefaultConfigCategory:



### Public Member Functions

- **DefaultConfigCategory** (const std::string &name, const std::string &json)
- **DefaultConfigCategory** (const ConfigCategory &orig)
- ~DefaultConfigCategory ()

  *Destructor for the default configuration category.*
- std::string toJSON () const

  *Return JSON string of all category components of a DefaultConfigCategory class.*
- std::string itemsToJSON () const

  *Return DefaultConfigCategory "default" items only.*

### Additional Inherited Members

### 4.45.1 Detailed Description

DefaultConfigCategory.

json input parameter must contain only "default" items. itemsToJSON() reports only "defaults"

This class must be used when creating/updating a category via ManagementClient::addCategoryDefault(Default↩ ConfigCategory categoryDefault)

### 4.45.2 Constructor & Destructor Documentation

#### 4.45.2.1 ~DefaultConfigCategory()

```
DefaultConfigCategory::~DefaultConfigCategory ( )
```

Destructor for the default configuration category.

Simply call the base class destructor.

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

- C/common/include/config_category.h
- C/common/config_category.cpp

## 4.46 DiskSpaceMonitor Class Reference

A class to monitor the free disk space used to store the various storage databases.

```
#include <disk_monitor.h>
```

## Public Member Functions

- DiskSpaceMonitor (const std::string &db1, const std::string &db2)

  *Construct a disk space monitor class.*
- void periodic (int interval)

  *Called periodically to monitor the disk usage.*

## 4.46.1 Detailed Description

A class to monitor the free disk space used to store the various storage databases.

## 4.46.2 Constructor & Destructor Documentation

### 4.46.2.1 DiskSpaceMonitor()

```
DiskSpaceMonitor::DiskSpaceMonitor (
            const std::string & db1,
            const std::string & db2 )
```

Construct a disk space monitor class.

It monitors the free space on two path, since the storage service may use different locations for reading and configuration storage. If they are both the same file system then the monitor is only done once.

If the free space falls below 5% a fatal error is written to the error log If it falls below 10% a warning is written advising that disk space should be released It attempts to predict when storage will become exhausted. If it is less then 14 days it will report this to the error log once a day. If it is less than 72 hours it will report this once per hour.

All reporting is to the system log

NB In an ideal world we would make the thresholds and reporting interval configurable, however we are running within the limited environment of a storage plugin and do not have access to the manaagement client or configuration subsystem.

## 4.46.3 Member Function Documentation

### 4.46.3.1 periodic()

```
void DiskSpaceMonitor::periodic (
            int interval )
```

Called periodically to monitor the disk usage.

**Parameters**

| | |
|---|---|
| *interval* | The number of seconds between calls |

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

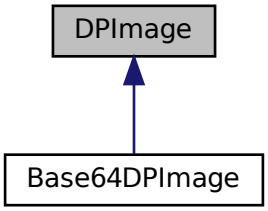
- C/plugins/storage/common/include/disk_monitor.h
- C/plugins/storage/common/disk_monitor.cpp

## 4.47 DPImage Class Reference

Simple Image class that will be used within data points to store image data.

```
#include <dpimage.h>
```

Inheritance diagram for DPImage:




### Public Member Functions

- DPImage (int width, int height, int depth, void ∗data)

  *DPImage constructor.*
- DPImage (const DPImage &rhs)

  *Copy constructor.*
- DPImage & operator= (const DPImage &rhs)

  *Assignment operator.*
- ∼DPImage ()

  *Destructor for the image.*
- int getHeight ()

  *Return the height of the image.*
- int getWidth ()

  *Return the width of the image.*
- int getDepth ()

  *Return the depth of the image in bits.*
- void ∗ getData ()

  *Return a pointer to the raw data of the image.*

## Protected Attributes

- int **m_width**
- int **m_height**
- int **m_depth**
- void * **m_pixels**
- int **m_byteSize**

### 4.47.1 Detailed Description

Simple Image class that will be used within data points to store image data.

This class merely acts to encapsulate the data of a simple image in memory, complex functionality will be supported elsewhere. Images within the class are stored as a simple, single area of memory the size of which is defined by the width, hieght and depth of the image.

### 4.47.2 Constructor & Destructor Documentation

#### 4.47.2.1 DPImage() [1/2]

```
DPImage::DPImage (
            int width,
            int height,
            int depth,
            void * data )
```

DPImage constructor.

**Parameters**

| | |
|---|---|
| *width* | The image width |
| *height* | The image height |
| *depth* | The image depth |
| *data* | The actual image data |

#### 4.47.2.2 DPImage() [2/2]

```
DPImage::DPImage (
            const DPImage & rhs )
```

Copy constructor.

**Parameters**

| | |
|---|---|
| *DPImage* | The image to copy |

### 4.47.3 Member Function Documentation

#### 4.47.3.1 operator=()

```
DPImage & DPImage::operator= (
            const DPImage & rhs )
```

Assignment operator.

**Parameters**

| | |
|---|---|
| *rhs* | Righthand side of equals operator |

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

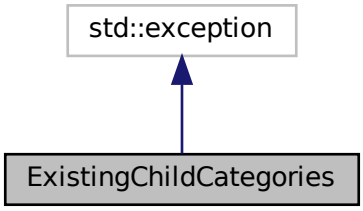
- C/common/include/dpimage.h
- C/common/image.cpp

## 4.48 ExistingChildCategories Class Reference

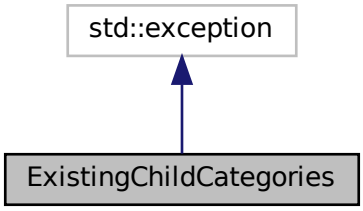
ExistingChildCategories.

```
#include <configuration_manager.h>
```

Inheritance diagram for ExistingChildCategories:



Collaboration diagram for ExistingChildCategories:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.48.1 Detailed Description

ExistingChildCategories.

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

- C/services/core/include/configuration_manager.h

## 4.49 Expression Class Reference

Class that defines data to be inserted or updated in a column within the table.

```
#include <expression.h>
```

### Public Member Functions

- **Expression** (const std::string &column, const std::string &op, int value)
- **Expression** (const std::string &column, const std::string &op, double value)
- const std::string **toJSON** () const

### 4.49.1 Detailed Description

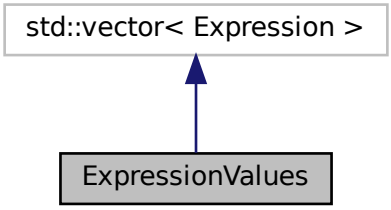
Class that defines data to be inserted or updated in a column within the table.

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

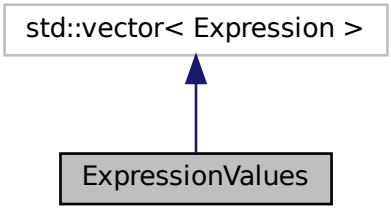
- C/common/include/expression.h

## 4.50 ExpressionValues Class Reference

Inheritance diagram for ExpressionValues:




Collaboration diagram for ExpressionValues:




### Public Member Functions

- const std::string **toJSON** () const

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

- C/common/include/expression.h

## 4.51 FormData::FieldValue Class Reference

### Public Attributes

- const uint8_t ∗ **start**
- size_t **size**
- std::string **filename**

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

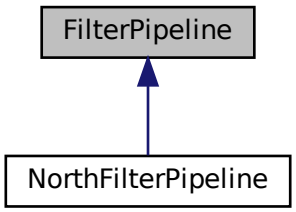
- C/common/include/form_data.h

## 4.52 FilterPipeline Class Reference

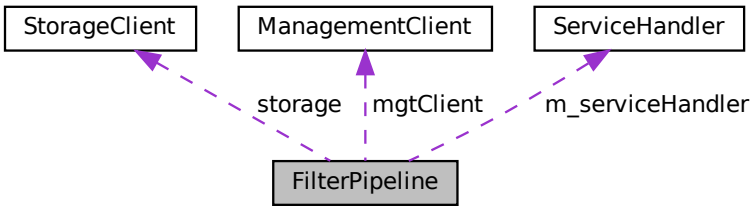
The FilterPipeline class is used to represent a pipeline of filters applicable to a task/service.

```
#include <filter_pipeline.h>
```

Inheritance diagram for FilterPipeline:




Collaboration diagram for FilterPipeline:




### Public Member Functions

- FilterPipeline (ManagementClient *mgtClient, StorageClient &storage, std::string serviceName)

  *FilterPipeline class constructor.*
- ~FilterPipeline ()

  *FilterPipeline destructor.*
- PipelineElement * **getFirstFilterPlugin** ()
- unsigned int **getFilterCount** ()
- void configChange (const std::string &, const std::string &)

  *Configuration change for one of the filters.*
- void cleanupFilters (const std::string &categoryName)

  *Cleanup all the loaded filters.*
- bool loadFilters (const std::string &categoryName)

  *Load all filter plugins in the pipeline.*

- bool setupFiltersPipeline (void ∗passToOnwardFilter, void ∗useFilteredData, void ∗ingest)

  *Set the filter pipeline.*
- bool **isReady** ()
- bool **hasChanged** (const std::string pipeline) const
- bool **isShuttingDown** ()
- void **setShuttingDown** ()
- void execute ()

  *Called when we pass the data into the pipeline.*
- void awaitCompletion ()

  *Wait for all active branches of the pipeline to complete.*
- void startBranch ()

  *A new branch has started in the pipeline.*
- void completeBranch ()

  *A branch in the pipeline has completed.*
- bool attachDebugger ()

  *Attach the debugger to the pipeline elements.*
- void detachDebugger ()

  *Detach the debugger from the pipeline elements.*
- void setDebuggerBuffer (unsigned int size)

  *Set the debugger buffer size to the pipeline elements.*
- std::string getDebuggerBuffer ()

  *Get the debugger buffer contents for all the pipeline elements.*
- std::string **getDebuggerBuffer** (const std::string &name)
- void replayDebugger ()

  *Replay the data in the first saved buffer to the filter pipeline.*

## Protected Attributes

- ManagementClient ∗ **mgtClient**
- StorageClient & **storage**
- std::string **serviceName**
- std::vector< PipelineElement ∗ > **m_filters**
- std::map< std::string, PipelineElement ∗ > **m_filterCategories**
- std::string **m_pipeline**
- bool **m_ready**
- bool **m_shutdown**
- ServiceHandler ∗ **m_serviceHandler**
- int **m_activeBranches**
- std::mutex **m_actives**
- std::condition_variable **m_branchActivations**

### 4.52.1 Detailed Description

The FilterPipeline class is used to represent a pipeline of filters applicable to a task/service.

Methods are provided to load filters, setup filtering pipeline and for pipeline/filters cleanup.

### 4.52.2 Constructor & Destructor Documentation

#### 4.52.2.1 FilterPipeline()

```
FilterPipeline::FilterPipeline (
            ManagementClient * mgtClient,
            StorageClient & storage,
            std::string serviceName )
```

FilterPipeline class constructor.

This class abstracts the filter pipeline interface

**Parameters**

| | |
|---|---|
| *mgtClient* | Management client handle |
| *storage* | Storage client handle |
| *serviceName* | Name of the service to which this pipeline applies |

### 4.52.3 Member Function Documentation

#### 4.52.3.1 attachDebugger()

```
bool FilterPipeline::attachDebugger ( )
```

Attach the debugger to the pipeline elements.

**Returns**

bool True if the pipeline was attached

#### 4.52.3.2 cleanupFilters()

```
void FilterPipeline::cleanupFilters (
            const std::string & categoryName )
```

Cleanup all the loaded filters.

Call "plugin_shutdown" method and free the FilterPlugin object

**Parameters**

| | |
|---|---|
| *categoryName* | Configuration category name |

#### 4.52.3.3 configChange()

```
void FilterPipeline::configChange (
            const std::string & ,
            const std::string &  )
```

Configuration change for one of the filters.

Lookup the category name and find the plugin to call. Call the reconfigure method of that plugin with the new configuration.

**Parameters**

| | |
|---|---|
| *category* | The name of the configuration category |
| *newConfig* | The new category contents |

#### 4.52.3.4 execute()

```
void FilterPipeline::execute ( )
```

Called when we pass the data into the pipeline.

Set the number of active branches to 1

#### 4.52.3.5 getDebuggerBuffer()

```
string FilterPipeline::getDebuggerBuffer ( )
```

Get the debugger buffer contents for all the pipeline elements.

**Returns**

string JSON document with all the buffer contents

#### 4.52.3.6 loadFilters()

```
bool FilterPipeline::loadFilters (
            const std::string & categoryName )
```

Load all filter plugins in the pipeline.

**Parameters**

| | |
|---|---|
| *categoryName* | Configuration category name |

**Returns**

True if filters are loaded (or no filters at all) False otherwise

#### 4.52.3.7 setDebuggerBuffer()

```
void FilterPipeline::setDebuggerBuffer (
            unsigned int size )
```

Set the debugger buffer size to the pipeline elements.

**Parameters**

| | |
|---|---|
| *size* | The request number of readings to buffer |

#### 4.52.3.8 setupFiltersPipeline()

```
bool FilterPipeline::setupFiltersPipeline (
            void * passToOnwardFilter,
            void * useFilteredData,
            void * ingest )
```

Set the filter pipeline.

This method calls the method "plugin_init" for all loadad filters. Up-to-date filter configurations and Ingest filtering methods are passed to "plugin_init"

**Parameters**

| | |
|---|---|
| *passToOnwardFilter* | Ptr to function that passes data to next filter |
| *useFilteredData* | Ptr to function that gets final filtered data |
| *ingest* | The ingest class handle |

**Returns**

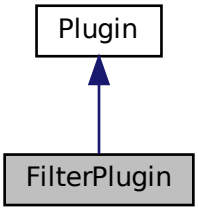
True on success, False otherwise. @thown Any caught exception

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

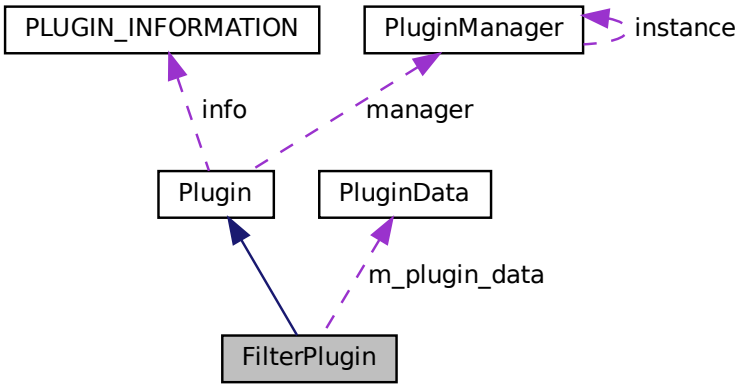
- C/common/include/filter_pipeline.h
- C/common/filter_pipeline.cpp

## 4.53 FilterPlugin Class Reference

Inheritance diagram for FilterPlugin:




Collaboration diagram for FilterPlugin:




### Public Member Functions

- FilterPlugin (const std::string &name, PLUGIN_HANDLE handle)

  *FilterPlugin class constructor.*
- ~FilterPlugin ()

  *FilterPlugin destructor.*
- const std::string **getName** () const
- PLUGIN_HANDLE init (const ConfigCategory &config, OUTPUT_HANDLE *outHandle, OUTPUT_STREAM outputFunc)

  *Call the loaded plugin "plugin_init" method.*
- void shutdown ()

  *Call the loaded plugin "plugin_shutdown" method.*
- void ingest (READINGSET *)

*Call the loaded plugin "plugin_ingest" method.*

- bool **persistData** ()
- void startData (const std::string &pluginData)

  *Call plugin_start passing plugin data.*
- std::string shutdownSaveData ()

  *Call the loaded plugin "plugin_shutdown" method returning plugind data (as string)*
- void start ()

  *Call plugin_start.*
- void reconfigure (const std::string &)

  *Call plugin_reconfigure method.*

## Public Attributes

- PluginData * **m_plugin_data**

## Additional Inherited Members

## 4.53.1 Constructor & Destructor Documentation

### 4.53.1.1 FilterPlugin()

```
FilterPlugin::FilterPlugin (
            const std::string & name,
            PLUGIN_HANDLE handle )
```

FilterPlugin class constructor.

This class wraps the filter plugin C interface and creates set of function pointers that resolve to the loaded plugin and enclose in the class.

**Parameters**

| | |
|---|---|
| *name* | The filter name |
| *handle* | The loaded plugin handle |

Set the function pointers to Filter Plugin C API

## 4.53.2 Member Function Documentation

### 4.53.2.1 ingest()

```
void FilterPlugin::ingest (
            READINGSET * readings )
```

Call the loaded plugin "plugin_ingest" method.

This call ingest the readings through the filters chain

**Parameters**

| | |
|---|---|
| *readings* | The reading set to ingest |

#### 4.53.2.2 init()

```
PLUGIN_HANDLE FilterPlugin::init (
            const ConfigCategory & config,
            OUTPUT_HANDLE * outHandle,
            OUTPUT_STREAM outputFunc )
```

Call the loaded plugin "plugin_init" method.

**Parameters**

| | |
|---|---|
| *config* | The filter configuration |
| *outHandle* | The output_handled passed with filtered data to OUTPUT_STREAM function |
| *outputFunc* | The output_stream function pointer the filter uses to pass data out |

**Returns**

The PLUGIN_HANDLE object

#### 4.53.2.3 reconfigure()

```
void FilterPlugin::reconfigure (
            const std::string &  )
```

Call plugin_reconfigure method.

**Parameters**

| | |
|---|---|
| *configuration* | The new filter configuration |

#### 4.53.2.4 shutdownSaveData()

```
string FilterPlugin::shutdownSaveData ( )
```

Call the loaded plugin "plugin_shutdown" method returning plugind data (as string)

**Returns**

Plugin data as JSON string (to be saved into strage layer)

#### 4.53.2.5 startData()

```
void FilterPlugin::startData (
            const std::string & pluginData )
```

Call plugin_start passing plugin data.

**Parameters**

| | |
|---|---|
| *storedData* | Plugin data to pass (from storage layer) |

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

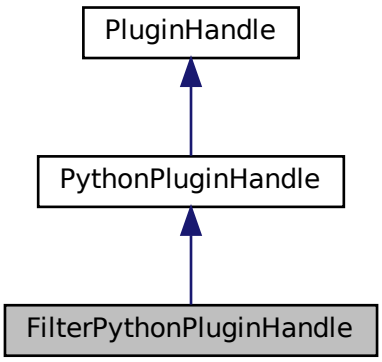
- C/common/include/filter_plugin.h
- C/common/filter_plugin.cpp

## 4.54 FilterPythonPluginHandle Class Reference

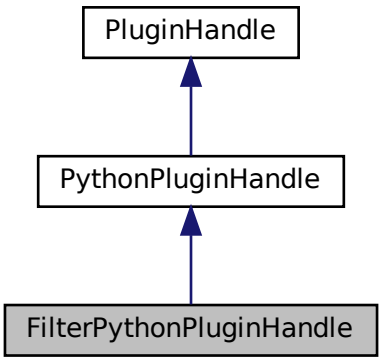
The PythonPluginHandle class is used to represent an interface to a plugin that is available as a python script.

```
#include <filter_python_plugin_handle.h>
```

Inheritance diagram for FilterPythonPluginHandle:



Collaboration diagram for FilterPythonPluginHandle:




## Public Member Functions

- FilterPythonPluginHandle (const char ∗name, const char ∗path)

  *Constructor for FilterPythonPluginHandle.*

## Additional Inherited Members

### 4.54.1 Detailed Description

The PythonPluginHandle class is used to represent an interface to a plugin that is available as a python script.

### 4.54.2 Constructor & Destructor Documentation

#### 4.54.2.1 FilterPythonPluginHandle()

```
FilterPythonPluginHandle::FilterPythonPluginHandle (
            const char * pluginName,
            const char * pluginPathName )
```

Constructor for FilterPythonPluginHandle.

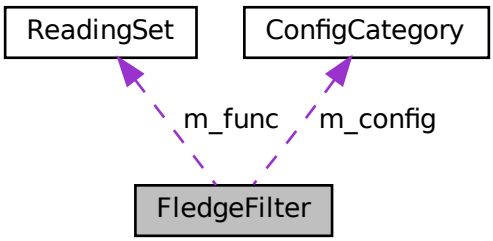
- Load python interface library and initialize the interface

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

- C/services/common/include/filter_python_plugin_handle.h
- C/services/common/filter_python_plugin_handle.cpp

## 4.55 FledgeFilter Class Reference

Collaboration diagram for FledgeFilter:




### Public Member Functions

- FledgeFilter (const std::string &filterName, ConfigCategory &filterConfig, OUTPUT_HANDLE *outHandle, OUTPUT_STREAM output)

  *FledgeFilter constructor.*
- const std::string & **getName** () const
- bool **isEnabled** () const
- ConfigCategory & **getConfig** ()
- void **disableFilter** ()
- void setConfig (const std::string &newConfig)

  *Set a new configurartion for this plugin.*

### Public Attributes

- OUTPUT_HANDLE * **m_data**
- OUTPUT_STREAM **m_func**

### Protected Attributes

- std::string **m_name**
- ConfigCategory **m_config**
- bool **m_enabled**

### 4.55.1 Constructor & Destructor Documentation

#### 4.55.1.1 FledgeFilter()

```
FledgeFilter::FledgeFilter (
            const std::string & filterName,
            ConfigCategory & filterConfig,
            OUTPUT_HANDLE * outHandle,
            OUTPUT_STREAM output )
```

FledgeFilter constructor.

This class or a derived one has to be used as return object from Fledge filters C interface "plugin_init"A

**Parameters**

| | |
|---|---|
| *filterName* | The filter plugin name |
| *filterConfig* | The filter plugin configuration |
| *outHandle* | A handle passed to the filter output stream function |
| *output* | The The output stream function pointer |

### 4.55.2 Member Function Documentation

#### 4.55.2.1 setConfig()

```
void FledgeFilter::setConfig (
            const std::string & newConfig )
```

Set a new configurartion for this plugin.

**Parameters**

| | |
|---|---|
| *newConfig* | The new configuration |

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

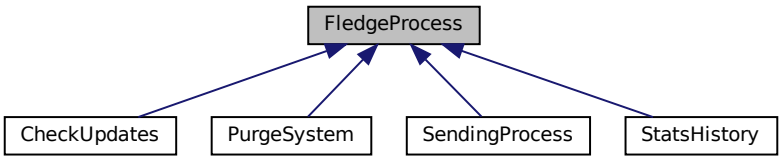
- C/plugins/filter/common/include/filter.h
- C/plugins/filter/common/filter.cpp

## 4.56 FledgeProcess Class Reference

Fledge process base class.

```
#include <process.h>
```

Inheritance diagram for FledgeProcess:



## Public Member Functions

- **FledgeProcess** (int argc, char **argv)
- StorageClient * getStorageClient () const

  *Return storage client.*
- ManagementClient * getManagementClient () const

  *Return management client.*
- Logger * getLogger () const

  *Return Logger.*
- std::string **getName** () const
- time_t **getStartTime** () const

## Protected Member Functions

- std::string getArgValue (const std::string &name) const

  *Get command line argument value like "--xyx=ABC" Argument name to pass is "--xyz=".*

## Protected Attributes

- bool **m_dryRun**

### 4.56.1 Detailed Description

Fledge process base class.

### 4.56.2 Member Function Documentation

#### 4.56.2.1 getArgValue()

```
string FledgeProcess::getArgValue (
            const std::string & name ) const  [protected]
```

Get command line argument value like "--xyx=ABC" Argument name to pass is "--xyz=".

**Parameters**

| | |
|---|---|
| *name* | The argument name (–xyz=) |

**Returns**

The argument value if found or an emopty string

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

- C/common/include/process.h
- C/common/process.cpp

## 4.57 FormData Class Reference

This class represents a parsed HTTP form data uploaded to SimpleWeb::Server<SimpleWeb::HTTP.

```
#include <form_data.h>
```

### Classes

- class FieldValue

### Public Member Functions

- **FormData** (std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Request > request)
- void getUploadedData (const std::string &field, FieldValue &data)

  *Fetch content data uploaded without file, example curl -v -v -v –output - -X POST -F 'attributes={"name": "B1", "type": "model"}' 127.0.0.1:43605/fledge/south/uploadA.*
- void getUploadedFile (const std::string &field, FieldValue &data)

  *Fetch content data uploaded as file, example curl -v -v -v –output - -X POST -F "bucket=@/some_path/file.bin" 127.↩ 0.0.1:43605/fledge/south/uploadA.*
- bool saveFile (FieldValue &b, const std::string &fileName)

  *Save the uploaded file.*

### 4.57.1 Detailed Description

This class represents a parsed HTTP form data uploaded to SimpleWeb::Server<SimpleWeb::HTTP.

FormData::FieldValue holds the field value as buffer start, size and filename if data comes form a file upload

FormData holds the input buffer, size and boundary multipart data

Public methods fetch value of a given field name and save file to filesystem

### 4.57.2 Member Function Documentation

#### 4.57.2.1 getUploadedData()

```
void FormData::getUploadedData (
            const std::string & field,
            FieldValue & data )
```

Fetch content data uploaded without file, example curl -v -v -v –output - -X POST -F 'attributes={"name": "B1", "type": "model"}' 127.0.0.1:43605/fledge/south/uploadA.

**Parameters**

| | |
|---|---|
| *field* | The field name to fetch |
| *data* | The value reference to fill on success |

#### 4.57.2.2 getUploadedFile()

```
void FormData::getUploadedFile (
            const std::string & field,
            FieldValue & data )
```

Fetch content data uploaded as file, example curl -v -v -v –output - -X POST -F "bucket=@/some_path/file.bin" 127.0.0.1:43605/fledge/south/uploadA.

**Parameters**

| | |
|---|---|
| *field* | The field name (filename type) to fetch |
| *data* | The value reference to fill on success |

#### 4.57.2.3 saveFile()

```
bool FormData::saveFile (
            FieldValue & b,
            const std::string & fileName )
```

Save the uploaded file.

**Parameters**

| | |
|---|---|
| *v* | The Field value data |

**Returns**

Returns true if the file was succesfully saved

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

- C/common/include/form_data.h
- C/common/form_data.cpp

## 4.58 std::hash< AssetTrackingTuple * > Struct Reference

### Public Member Functions

- size_t **operator()** (AssetTrackingTuple *t) const

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

- C/common/include/asset_tracking.h

## 4.59 std::hash< AssetTrackingTuple > Struct Reference

### Public Member Functions

- size_t **operator()** (const AssetTrackingTuple &t) const

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

- C/common/include/asset_tracking.h

## 4.60 std::hash< StorageAssetTrackingTuple * > Struct Reference

### Public Member Functions

- size_t **operator()** (StorageAssetTrackingTuple *t) const

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

- C/common/include/asset_tracking.h

## 4.61 std::hash< StorageAssetTrackingTuple > Struct Reference

### Public Member Functions

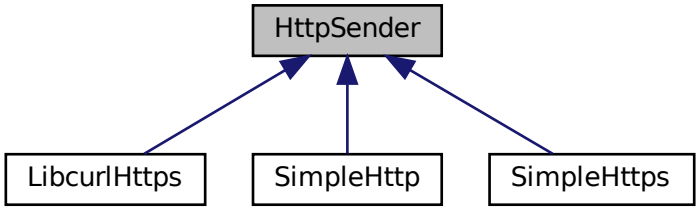
- size_t **operator()** (const StorageAssetTrackingTuple &t) const

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

- C/common/include/asset_tracking.h

## 4.62 HttpSender Class Reference

Inheritance diagram for HttpSender:



## Public Member Functions

- HttpSender ()

  *Constructor:*
- virtual ~HttpSender ()

  *Destructor.*
- virtual void setProxy (const std::string &proxy)=0

  *Set a proxy server.*
- virtual int sendRequest (const std::string &method=std::string(HTTP_SENDER_DEFAULT_METHOD), const std::string &path=std::string(HTTP_SENDER_DEFAULT_PATH), const std::vector< std::pair< std::string, std::string >> &headers={}, const std::string &payload=std::string())=0

  *HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.*
- virtual std::string **getHostPort** ()=0
- virtual std::string **getHTTPResponse** ()=0
- virtual unsigned int **getMaxRetries** ()=0
- virtual void **setAuthMethod** (std::string &authMethod)=0
- virtual void **setAuthBasicCredentials** (std::string &authBasicCredentials)=0
- virtual void **setMaxRetries** (unsigned int retries)=0
- virtual void **setOCSNamespace** (std::string &OCSNamespace)=0
- virtual void **setOCSTenantId** (std::string &OCSTenantId)=0
- virtual void **setOCSClientId** (std::string &OCSClientId)=0
- virtual void **setOCSClientSecret** (std::string &OCSClientSecret)=0
- virtual void **setOCSToken** (std::string &OCSToken)=0

## Static Public Member Functions

- static std::string getOMFTracePath ()

  *Constructs the file path for the OMF log.*
- static bool createDebugTraceDirectory ()

  *Creates the '/logs/debug-trace' subdirectory in the Fledge data directory.*

## 4.62.1 Constructor & Destructor Documentation

### 4.62.1.1 HttpSender()

```
HttpSender::HttpSender ( )
```

Constructor:

Constructor.

## 4.62.2 Member Function Documentation

#### 4.62.2.1 createDebugTraceDirectory()

```
bool HttpSender::createDebugTraceDirectory ( ) [static]
```

Creates the '/logs/debug-trace' subdirectory in the Fledge data directory.

This function ensures that both the 'logs' directory and the 'debug-trace' directory are created if they do not exist.

#### 4.62.2.2 getOMFTracePath()

```
std::string HttpSender::getOMFTracePath ( ) [static]
```

Constructs the file path for the OMF log.

**Returns**

A string representing the path to the OMF log file.

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

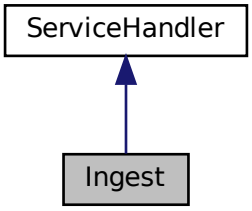
- C/plugins/common/include/http_sender.h
- C/plugins/common/http_sender.cpp

## 4.63 Ingest Class Reference

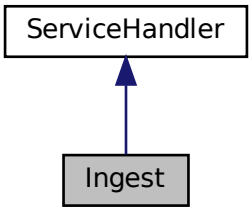
The ingest class is used to ingest asset readings.

```
#include <ingest.h>
```

Inheritance diagram for Ingest:



Collaboration diagram for Ingest:




## Public Member Functions

- Ingest (StorageClient &storage, const std::string &serviceName, const std::string &pluginName, ManagementClient *mgmtClient)

  *Construct an Ingest class to handle the readings queue.*
- ~Ingest ()

  *Destructor for the Ingest class.*
- void ingest (const Reading &reading)

  *Add a reading to the reading queue.*
- void ingest (const std::vector< Reading * > *vec)

  *Add a set of readings to the reading queue.*
- void start (long timeout, unsigned int threshold)

  *Start the ingest threads.*
- bool running ()

  *Check if the ingest process is still running.*
- bool isStopping ()

  *Check if a shutdown is requested.*
- bool **isRunning** ()
- void processQueue ()

  *Process the queue of readings.*
- void waitForQueue ()

  *Wait for a period of time to allow the queue to build.*
- size_t queueLength ()

  *Return the number of queued readings in the south service.*
- void updateStats (void)

  *Update statistics for this south service.*
- int createStatsDbEntry (const std::string &assetName)

  *Create a row for given assetName in statistics DB table, if not present already The key checked/created in the table is "<assetName>".*
- bool loadFilters (const std::string &categoryName)

  *Load filter plugins.*
- void **setTimeout** (const long timeout)
- void **setThreshold** (const unsigned int threshold)
- void configChange (const std::string &, const std::string &)

  *Configuration change for one of the filters or to the pipeline.*

- void **configChildCreate** (const std::string &, const std::string &, const std::string &)
- void **configChildDelete** (const std::string &, const std::string &)
- void **shutdown** ()
- void **restart** ()
- void unDeprecateAssetTrackingRecord (AssetTrackingTuple *currentTuple, const std::string &assetName, const std::string &event)

  *Load an up-to-date AssetTracking record for the given parameters and un-deprecate AssetTracking record it has been found as deprecated Existing cache element is updated.*
- void unDeprecateStorageAssetTrackingRecord (StorageAssetTrackingTuple *currentTuple, const std::string &assetName, const std::string &, const unsigned int &)

  *Load an up-to-date StorageAssetTracking record for the given parameters and un-deprecate StorageAssetTracking record it has been found as deprecated Existing cache element is updated.*
- void setStatistics (const std::string &option)

  *Set the statistics option.*
- std::string **getStringFromSet** (const std::set< std::string > &dpSet)
- void **setFlowControl** (unsigned int lowWater, unsigned int highWater)
- void setResourceLimit (ServiceBufferingType serviceBufferingType, unsigned long serviceBufferSize, DiscardPolicy discardPolicy)

  *Set the resource limit for the service.*
- void flowControl ()

  *Implement flow control backoff for the async ingest mechanism.*
- void **setPerfMon** (PerformanceMonitor *mon)
- void configureRateMonitor (long interval, long factor)

  *Configure the ingest rate class with the collection interval and the sigma factor allowed before reporting.*
- bool **attachDebugger** ()
- void **detachDebugger** ()
- void **setDebuggerBuffer** (unsigned int size)
- std::string **getDebuggerBuffer** ()
- void **isolate** (bool isolate)
- bool **isolated** ()
- void **replayDebugger** ()

## Static Public Member Functions

- static void passToOnwardFilter (OUTPUT_HANDLE *outHandle, READINGSET *readings)

  *Pass the current readings set to the next filter in the pipeline.*
- static void useFilteredData (OUTPUT_HANDLE *outHandle, READINGSET *readings)

  *Use the current input readings (they have been filtered by all filters)*

### 4.63.1 Detailed Description

The ingest class is used to ingest asset readings.

It maintains a queue of readings to be sent to storage, these are sent using a background thread that regularly wakes up and sends the queued readings.

### 4.63.2 Constructor & Destructor Documentation

#### 4.63.2.1 Ingest()

```
Ingest::Ingest (
            StorageClient & storage,
            const std::string & serviceName,
            const std::string & pluginName,
            ManagementClient * mgmtClient )
```

Construct an Ingest class to handle the readings queue.

A seperate thread is used to send the readings to the storage layer based on time. This thread in created in the constructor and will terminate when the destructor is called.

**Parameters**

| | |
|---|---|
| *storage* | The storage client to use |

#### 4.63.2.2 ~Ingest()

```
Ingest::~Ingest ( )
```

Destructor for the Ingest class.

Set's the running flag to false. This will cause the processing thread to drain the queue and then exit. Once this thread has exited the destructor will return.

### 4.63.3 Member Function Documentation

#### 4.63.3.1 configChange()

```
void Ingest::configChange (
            const std::string & ,
            const std::string &  )  [virtual]
```

Configuration change for one of the filters or to the pipeline.

**Parameters**

| | |
|---|---|
| *category* | The name of the configuration category |
| *newConfig* | The new category contents |

The category that has changed is the one for the south service itself. The only item that concerns us here is the filter item that defines the filter pipeline. We extract that item and check to see if it defines a pipeline that is different to the one we currently have.

If it is we destroy the current pipeline and create a new one.

Implements ServiceHandler.

#### 4.63.3.2 configureRateMonitor()

```
void Ingest::configureRateMonitor (
            long interval,
            long factor )
```

Configure the ingest rate class with the collection interval and the sigma factor allowed before reporting.

**Parameters**

| | |
|---|---|
| *interval* | Number of minutes to average ingest stats over |
| *factor* | Number of standard deviations to allow before reporting |

#### 4.63.3.3 createStatsDbEntry()

```
int Ingest::createStatsDbEntry (
            const std::string & assetName )
```

Create a row for given assetName in statistics DB table, if not present already The key checked/created in the table is "<assetName>".

**Parameters**

| | |
|---|---|
| *assetName* | Asset name for the plugin that is sending readings |

**Returns**

int Return -1 on error, 0 if not required or 1 if the entry exists

#### 4.63.3.4 flowControl()

```
void Ingest::flowControl ( )
```

Implement flow control backoff for the async ingest mechanism.

The flow control is "soft" in that it will only wait for a maximum amount of time before continuing regardless of the queue length.

The mechanism is to have a high water and low water mark. When the queue get longer than the high water mark we wait until the queue drains below the low water mark before proceeding.

The wait is done with a backoff algorithm that start at AFC_SLEEP_INCREMENT and doubles each time we have not dropped below the low water mark. It will sleep for a maximum of AFC_SLEEP_MAX before testing again.

#### 4.63.3.5 ingest() [1/2]

```
void Ingest::ingest (
            const Reading & reading )
```

Add a reading to the reading queue.

**Parameters**

| | |
|---|---|
| *reading* | The single reading to ingest |

#### 4.63.3.6 ingest() [2/2]

```
void Ingest::ingest (
            const std::vector< Reading * > * vec )
```

Add a set of readings to the reading queue.

**Parameters**

| | |
|---|---|
| *vec* | A vector of readings to ingest |

#### 4.63.3.7 loadFilters()

```
bool Ingest::loadFilters (
            const std::string & categoryName )
```

Load filter plugins.

Filters found in configuration are loaded and add to the Ingest class instance

**Parameters**

| | |
|---|---|
| *categoryName* | Configuration category name |
| *ingest* | The Ingest class reference Filters are added to m_filters member False for errors. |

**Returns**

True if filters were loaded and initialised or there are no filters False with load/init errors

#### 4.63.3.8 passToOnwardFilter()

```
void Ingest::passToOnwardFilter (
            OUTPUT_HANDLE * outHandle,
            READINGSET * readingSet )  [static]
```

Pass the current readings set to the next filter in the pipeline.

Note: This routine must be passed to all filters "plugin_init" except the last one

Static method

**Parameters**

| | |
|---|---|
| *outHandle* | Pointer to next filter |
| *readings* | Current readings set |

#### 4.63.3.9 processQueue()

```
void Ingest::processQueue ( )
```

Process the queue of readings.

Send them to the storage layer as a block. If the append call fails requeue the readings for the next transmission.

In order not to lock the queue for an excessie time a new queue is created and the old one moved to a local variable. This minimise the time we hold the queue mutex to the time it takes to swap two variables. 'm_data' vector is ready to be sent to storage service.

Note: m_data might contain:

- Readings set by the configured service "plugin" OR
- filtered readings by filter plugins in 'readingSet' object: 1- values only 2- some readings removed 3- New set of readings

#### 4.63.3.10 running()

```
bool Ingest::running ( )
```

Check if the ingest process is still running.

This becomes false when the service is shutdown and is used to allow the queue to drain and then the processing routine to terminate.

#### 4.63.3.11 setResourceLimit()

```
void Ingest::setResourceLimit (
            ServiceBufferingType serviceBufferingType,
            unsigned long serviceBufferSize,
            DiscardPolicy discardPolicy )
```

Set the resource limit for the service.

**Parameters**

| | |
|---|---|
| *serviceBufferingType* | Buffering type |
| *serviceBufferSize* | Buffer size |
| *discardPolicy* | Discard policy |

#### 4.63.3.12 setStatistics()

```
void Ingest::setStatistics (
            const std::string & option )
```

Set the statistics option.

The statistics collection regime may be one of "per asset", "per service" or "per asset & service".

**Parameters**

| | |
|---|---|
| *option* | The desired statistics collection regime |

#### 4.63.3.13 start()

```
void Ingest::start (
            long timeout,
            unsigned int threshold )
```

Start the ingest threads.

**Parameters**

| | |
|---|---|
| *timeout* | Maximum time before sending a queue of readings in milliseconds |
| *threshold* | Length of queue before sending readings |

#### 4.63.3.14 unDeprecateAssetTrackingRecord()

```
void Ingest::unDeprecateAssetTrackingRecord (
            AssetTrackingTuple * currentTuple,
            const std::string & assetName,
            const std::string & event )
```

Load an up-to-date AssetTracking record for the given parameters and un-deprecate AssetTracking record it has been found as deprecated Existing cache element is updated.

**Parameters**

| | |
|---|---|
| *currentTuple* | Current AssetTracking record for given assetName |
| *assetName* | AssetName to fetch from AssetTracking |
| *event* | The event type to fetch |

#### 4.63.3.15 unDeprecateStorageAssetTrackingRecord()

```
void Ingest::unDeprecateStorageAssetTrackingRecord (
            StorageAssetTrackingTuple * currentTuple,
            const std::string & assetName,
            const std::string & ,
            const unsigned int &  )
```

Load an up-to-date StorageAssetTracking record for the given parameters and un-deprecate StorageAssetTracking record it has been found as deprecated Existing cache element is updated.

**Parameters**

| | |
|---|---|
| *currentTuple* | Current StorageAssetTracking record for given assetName |
| *assetName* | AssetName to fetch from AssetTracking |
| *datapoints* | The datapoints comma separated list |
| *count* | The number of datapoints per asset |

#### 4.63.3.16 updateStats()

```
void Ingest::updateStats (
            void  )
```

Update statistics for this south service.

Successfully processed readings are reflected against plugin asset name and READINGS keys. Discarded readings stats are updated against DISCARDED key.

#### 4.63.3.17 useFilteredData()

```
void Ingest::useFilteredData (
            OUTPUT_HANDLE * outHandle,
            READINGSET * readingSet )  [static]
```

Use the current input readings (they have been filtered by all filters)

The assumption is that one of two things has happened.

1. The filtering has all been done in place. In which case the m_data vector is in the ReadingSet passed in here.
2. The filtering has created new ReadingSet in which case the reading vector must be copied into m_data from the ReadingSet.

Note: This routine must be passed to last filter "plugin_init" only

Static method

**Parameters**

| | |
|---|---|
| *outHandle* | Pointer to Ingest class instance |
| *readingSet* | Filtered reading set being added to Ingest::m_data |

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

- C/services/south/include/ingest.h
- C/services/south/ingest.cpp

## 4.64 IngestRate Class Reference

A class used to track and report on the ingest rates of a data stream.

```
#include <ingest_rate.h>
```

### Public Member Functions

- IngestRate (ManagementClient *mgmtClient, const std::string &service)

  *Constructor for ingest rate class.*
- ~IngestRate ()

  *Destructor for the ingest rate class.*
- void ingest (unsigned int numberReadings)

  *Called each time we ingest any readings.*
- void periodic ()

  *Called periodically by the stats update thread.*
- void updateConfig (int interval, int factor)

  *Update the configuration of the ingest rate mechanism.*
- void relearn ()

  *The configuration has changed so we need to reset our state and go back into the mode of determining what a good mean and standard deviation is for the select interval.*

### 4.64.1 Detailed Description

A class used to track and report on the ingest rates of a data stream.

It collects the number of readings ingested in a configurable period, the period being defined in minutes. It then calculates the mean of the collection rates and the standard deviation. If the current collection rate differs from the averaged collection rate by more than a configured number of standard deviations an alert is raised. A "priming" mechanism is used to require two consecutive deviations from the norm to be required before an alert is trigger. This reduces the occurance of false positives caused by transient spikes in colection. These spikes may occur when heavy operations are performed on the Fledge instance.

When the rate returns to within the number of defined standard deviations of the average then the alert is cleared.

Before alerting is enabled a number of the defined time periods, IGRSAMPLES, must have passed, during this time an average is calculated and used for the intial comparision.

If the user reconfigures the collection rate of the plugin then the accumulated average is discarded and the process starts again by collecting a new average

## 4.64.2 Constructor & Destructor Documentation

### 4.64.2.1 IngestRate()

```
IngestRate::IngestRate (
            ManagementClient * mgmtClient,
            const std::string & service )
```

Constructor for ingest rate class.

**Parameters**

| | |
|---|---|
| *mgmtClient* | The management Client interface |

## 4.64.3 Member Function Documentation

### 4.64.3.1 ingest()

```
void IngestRate::ingest (
            unsigned int numberReadings )
```

Called each time we ingest any readings.

**Parameters**

| | |
|---|---|
| *numberReadings* | The number of readings ingested |

### 4.64.3.2 periodic()

```
void IngestRate::periodic ( )
```

Called periodically by the stats update thread.

Called every FLUSH_STATS_INTERVAL seconds

### 4.64.3.3 updateConfig()

```
void IngestRate::updateConfig (
            int interval,
            int factor )
```

Update the configuration of the ingest rate mechanism.

**Parameters**

| | |
|---|---|
| *interval* | Number of minutes to average over |
| *factor* | Number of standard deviations to tolarate |

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

- C/services/south/include/ingest_rate.h
- C/services/south/ingestRate.cpp

## 4.65 InsertValue Class Reference

Class that defines data to be inserted or updated in a column within the table.

```
#include <insert.h>
```

### Public Member Functions

- **InsertValue** (const std::string &column, const std::string &value)
- **InsertValue** (const std::string &column, const int value)
- **InsertValue** (const std::string &column, const long value)
- **InsertValue** (const std::string &column, const double value)
- **InsertValue** (const std::string &column, const rapidjson::Value &value)
- **InsertValue** (const std::string &column)
- **InsertValue** (const InsertValue &rhs)
- const std::string **toJSON** () const

### 4.65.1 Detailed Description

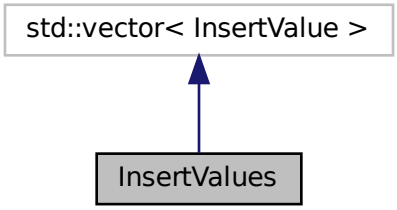
Class that defines data to be inserted or updated in a column within the table.

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

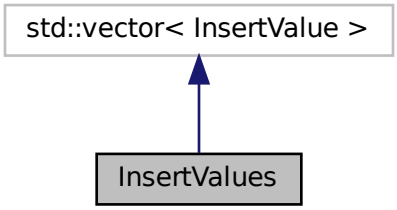
- C/common/include/insert.h

## 4.66 InsertValues Class Reference

Inheritance diagram for InsertValues:




Collaboration diagram for InsertValues:




### Public Member Functions

- const std::string **toJSON** () const

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

- C/common/include/insert.h

## 4.67 Join Class Reference

Join clause representation.

```
#include <join.h>
```

### Public Member Functions

- **Join** (const std::string &table, const std::string &on, Query *query)
- **Join** (const std::string &table, const std::string &column, const std::string &on, Query *query)
- ~Join ()

  *Destructor fo rthe join clause.*
- const std::string toJSON () const

  *Convert a join clause to its JSON representation.*

## 4.67.1 Detailed Description

Join clause representation.

## 4.67.2 Member Function Documentation

#### 4.67.2.1 toJSON()

```
const string Join::toJSON ( ) const
```

Convert a join clause to its JSON representation.

**Returns**

string The JSON form of the join

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

- C/common/include/join.h
- C/common/join.cpp

# 4.68 JSONPath Class Reference

A simple implementation of a JSON Path search mechanism to use alongside RapidJSON.

```
#include <JSONPath.h>
```

### Public Member Functions

- **JSONPath** (const std::string &path)
- ~JSONPath ()

  *Destructor for the JSONPath.*
- rapidjson::Value * findNode (rapidjson::Value &root)

  *Find the matching node in the JSON document.*

### 4.68.1 Detailed Description

A simple implementation of a JSON Path search mechanism to use alongside RapidJSON.

### 4.68.2 Constructor & Destructor Documentation

#### 4.68.2.1 ~JSONPath()

```
JSONPath::~JSONPath ( )
```

Destructor for the JSONPath.

Reclaim the vector of components.

### 4.68.3 Member Function Documentation

#### 4.68.3.1 findNode()

```
Value * JSONPath::findNode (
            rapidjson::Value & root )
```

Find the matching node in the JSON document.

param node The node to search from

**Returns**

the matching node. Throws an exception if there was no match

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

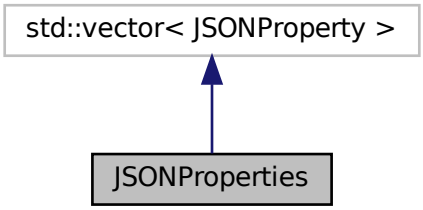
- C/common/include/JSONPath.h
- C/common/JSONPath.cpp

## 4.69 JSONProperties Class Reference

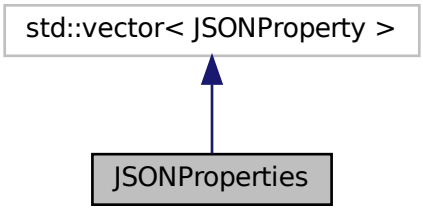
Class that defines JSON properties for update.

```
#include <json_properties.h>
```

Inheritance diagram for JSONProperties:




Collaboration diagram for JSONProperties:




### Public Member Functions

- const std::string **toJSON** () const

### 4.69.1 Detailed Description

Class that defines JSON properties for update.

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

- C/common/include/json_properties.h

## 4.70 JSONProperty Class Reference

### Public Member Functions

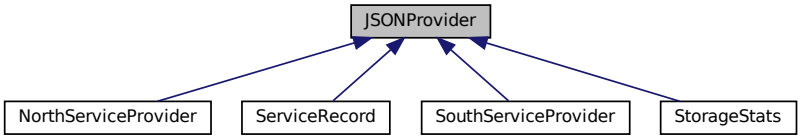
- **JSONProperty** (const std::string &column, std::vector< std::string > path, const std::string &value)
- const std::string **toJSON** () const

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

- C/common/include/json_properties.h

## 4.71 JSONProvider Class Reference

Inheritance diagram for JSONProvider:




### Public Member Functions

- virtual void **asJSON** (std::string &) const =0

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

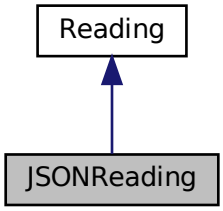
- C/common/include/json_provider.h

## 4.72 JSONReading Class Reference

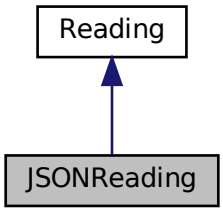
JSONReading class.

```
#include <reading_set.h>
```

Inheritance diagram for JSONReading:




Collaboration diagram for JSONReading:




## Public Member Functions

- JSONReading (const rapidjson::Value &json)

  *Construct a reading from a JSON document.*
- unsigned long **getId** () const

## Additional Inherited Members

### 4.72.1 Detailed Description

JSONReading class.

A specialised reading class that allows creation from a JSON document

### 4.72.2 Constructor & Destructor Documentation

#### 4.72.2.1 JSONReading()

```
JSONReading::JSONReading (
            const rapidjson::Value & json )
```

Construct a reading from a JSON document.

The data can be in the "value" property as single numeric value or in the JSON "reading" with different values and types

**Parameters**

| | |
|---|---|
| *json* | The JSON document that contains the reading |

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

- C/common/include/reading_set.h
- C/common/reading_set.cpp

## 4.73 ACL::KeyValueItem Class Reference

### Public Member Functions

- **KeyValueItem** (const std::string &k, const std::string &v)

### Public Attributes

- std::string **key**
- std::string **value**

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

- C/common/include/acl.h

## 4.74 LALookup Class Reference

Linked Asset Information class.

```
#include <linkedlookup.h>
```

## Public Member Functions

- bool **assetState** (const std::string &tagName)
- bool **linkState** (const std::string &tagName)
- bool **containerState** (const std::string &tagName)
- bool **afLinkState** ()
- void setBaseType (const std::string &baseType)

  *Set the base type by passing the string of the base type.*
- OMFBaseType **getBaseType** ()
- std::string getBaseTypeString ()

  *Get a string representation of the base type that was sent.*
- void **assetSent** (const std::string &tagName)
- void **linkSent** (const std::string &tagName)
- void **afLinkSent** ()
- void containerSent (const std::string &tagName, const std::string &baseType)

  *The container has been sent with the specific base type.*
- void containerSent (const std::string &tagName, OMFBaseType baseType)

  *The container has been sent with the specific base type.*

### 4.74.1 Detailed Description

Linked Asset Information class.

This is the data stored for each asset and asset datapoint pair that is being sent to PI using the linked container mechanism. We use the class so we can combine all the information we need in a single lookup table, this not only saves space but allows to build and retain the table before we start building the payloads. This hopefully will help prevent to much memory fragmentation, which was an issue with the old, separate lookup mechanism we had.

### 4.74.2 Member Function Documentation

#### 4.74.2.1 containerSent() [1/2]

```
void LALookup::containerSent (
            const std::string & tagName,
            const std::string & baseType )
```

The container has been sent with the specific base type.

**Parameters**

| | |
|---|---|
| *tagName* | The name of the tag we are using |
| *baseType* | The baseType we resolve to |

#### 4.74.2.2 containerSent() [2/2]

```
void LALookup::containerSent (
            const std::string & tagName,
            OMFBaseType baseType )
```

The container has been sent with the specific base type.

**Parameters**

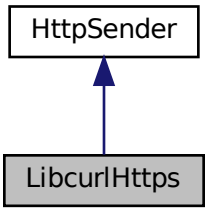
| | |
|---|---|
| *tagName* | The name of the tag we are using |
| *baseType* | The baseType we resolve to |

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

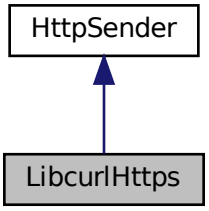
- C/plugins/north/OMF/include/linkedlookup.h
- C/plugins/north/OMF/linkdata.cpp

## 4.75 LibcurlHttps Class Reference

Inheritance diagram for LibcurlHttps:




Collaboration diagram for LibcurlHttps:



## Public Member Functions

- LibcurlHttps (const std::string &host_port, unsigned int connect_timeout=0, unsigned int request_timeout=0, unsigned int retry_sleep_Time=1, unsigned int max_retry=4)

  *Constructor: pass host:port, connect & request timeouts, retry_sleep_Time, max_retry.*
- ~LibcurlHttps ()

  *Destructor.*
- void setProxy (const std::string &proxy)

  *Set the proxy host and port.*
- int sendRequest (const std::string &method=std::string(HTTP_SENDER_DEFAULT_METHOD), const std↩ ::string &path=std::string(HTTP_SENDER_DEFAULT_PATH), const std::vector< std::pair< std::string, std↩ ::string >> &headers={}, const std::string &payload=std::string())

  *HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.*
- void **setAuthMethod** (std::string &authMethod)
- void **setAuthBasicCredentials** (std::string &authBasicCredentials)
- void **setMaxRetries** (unsigned int retries)
- void **setOCSNamespace** (std::string &OCSNamespace)
- void **setOCSTenantId** (std::string &OCSTenantId)
- void **setOCSClientId** (std::string &OCSClientId)
- void **setOCSClientSecret** (std::string &OCSClientSecret)
- void **setOCSToken** (std::string &OCSToken)
- std::string **getHostPort** ()
- std::string **getHTTPResponse** ()
- unsigned int **getMaxRetries** ()

## Additional Inherited Members

## 4.75.1 Member Function Documentation

### 4.75.1.1 sendRequest()

```
int LibcurlHttps::sendRequest (
            const std::string & method = std::string(HTTP_SENDER_DEFAULT_METHOD),
            const std::string & path = std::string(HTTP_SENDER_DEFAULT_PATH),
            const std::vector< std::pair< std::string, std::string >> & headers = {},
            const std::string & payload = std::string() )  [virtual]
```

HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.

Send data, it retries the operation m_max_retry times waiting m_retry_sleep_time∗2 at each attempt.

**Parameters**

| | |
|---|---|
| *method* | The HTTP method (GET, POST, ...) |
| *path* | The URL path |
| *headers* | The optional headers to send |
| *payload* | The optional data payload (for POST, PUT) |

**Returns**

The HTTP code for the cases : 1xx Informational / 2xx Success / 3xx Redirection

**Exceptions**

| | |
|---|---|
| *BadRequest* | for HTTP 400 error std::exception as generic exception for all the cases >= 401 Client errors / 5xx Server errors |

Implements HttpSender.

#### 4.75.1.2 setProxy()

```
void LibcurlHttps::setProxy (
            const std::string & proxy )  [virtual]
```

Set the proxy host and port.

Add a proxy server.

**Parameters**

| | |
|---|---|
| *proxy* | The host and port of the proxy |

Implements HttpSender.

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

- C/plugins/common/include/libcurl_https.h
- C/plugins/common/libcurl_https.cpp

## 4.76 Logger Class Reference

Fledge Logger class used to log to syslog.

```
#include <logger.h>
```

### Public Types

- enum **LogLevel** {
  **ERROR**, **WARNING**, **INFO**, **DEBUG**,
  **FATAL** }
- typedef void(∗ **LogInterceptor**) (LogLevel, const std::string &, void ∗)

## Public Member Functions

- Logger (const std::string &application)

  *Constructor for the Logger class.*
- ~Logger ()

  *Destructor for the logger class.*
- void debug (const std::string &msg,...)

  *Log a message at the level debug.*
- void printLongString (const std::string &, LogLevel=LogLevel::DEBUG)

  *Log a long string across multiple syslog entries.*
- void info (const std::string &msg,...)

  *Log a message at the level info.*
- void warn (const std::string &msg,...)

  *Log a message at the level warn.*
- void error (const std::string &msg,...)

  *Log a message at the level error.*
- void fatal (const std::string &msg,...)

  *Log a message at the level fatal.*
- void setMinLevel (const std::string &level)

  *Set the minimum level of logging to write to syslog.*
- std::string & **getMinLevel** ()
- bool registerInterceptor (LogLevel level, LogInterceptor callback, void *userData)

  *Register a callback function to be called when a log message is written that matches the specification given.*
- bool unregisterInterceptor (LogLevel level, LogInterceptor callback)

  *Remove the registration of a previously registered callback.*

## Static Public Member Functions

- static Logger * getLogger ()

  *Return the singleton instance of the logger class.*

## 4.76.1 Detailed Description

Fledge Logger class used to log to syslog.

At startup this class should be constructed using the standard constructor. To log a message call debug, info, warn etc. using the instance of the class.

To obtain that singleton instance call the static method getLogger.

It is generally unsafe to delete the logger class as it may be called asynchronouly from multiple threads and single handlers. The destructor has hence been made private to prevent the destruction of the class.

## 4.76.2 Constructor & Destructor Documentation

### 4.76.2.1 Logger()

```
Logger::Logger (
            const std::string & application )
```

Constructor for the Logger class.

**Parameters**

| | |
|---|---|
| *application* | The application name |

## 4.76.3 Member Function Documentation

### 4.76.3.1 debug()

```
void Logger::debug (
            const std::string & msg,
             ...  )
```

Log a message at the level debug.

**Parameters**

| | |
|---|---|
| *msg* | A printf format string |
| *...* | The variable arguments required by the printf format |

### 4.76.3.2 error()

```
void Logger::error (
            const std::string & msg,
             ...  )
```

Log a message at the level error.

**Parameters**

| | |
|---|---|
| *msg* | A printf format string |
| *...* | The variable arguments required by the printf format |

### 4.76.3.3 fatal()

```
void Logger::fatal (
            const std::string & msg,
             ...  )
```

Log a message at the level fatal.

**Parameters**

| | |
|---|---|
| *msg* | A printf format string |
| *...* | The variable arguments required by the printf format |

#### 4.76.3.4 info()

```
void Logger::info (
            const std::string & msg,
             ...  )
```

Log a message at the level info.

**Parameters**

| | |
|---|---|
| *msg* | A printf format string |
| *...* | The variable arguments required by the printf format |

#### 4.76.3.5 printLongString()

```
void Logger::printLongString (
            const std::string & ,
            LogLevel  = LogLevel::DEBUG )
```

Log a long string across multiple syslog entries.

**Parameters**

| | |
|---|---|
| *s* | The string to log |
| *level* | level to log the string at |

#### 4.76.3.6 registerInterceptor()

```
bool Logger::registerInterceptor (
            LogLevel level,
            LogInterceptor callback,
            void * userData )
```

Register a callback function to be called when a log message is written that matches the specification given.

Note: The callback functions are called on a separate thread. This worker thread is only created when the first callback is registered.

**Parameters**

| | |
|---|---|
| *level* | The level that must be matched |
| *callback* | The funtion to be called |
| *userData* | User date to pass to the callback function |

**Returns**

bool Return true if the callback was registered

#### 4.76.3.7 setMinLevel()

```
void Logger::setMinLevel (
            const std::string & level )
```

Set the minimum level of logging to write to syslog.

**Parameters**

| | |
|---|---|
| *level* | The minimum, inclusive, level of logging to write |

#### 4.76.3.8 unregisterInterceptor()

```
bool Logger::unregisterInterceptor (
            LogLevel level,
            LogInterceptor callback )
```

Remove the registration of a previously registered callback.

**Parameters**

| | |
|---|---|
| *level* | The matching log level for the callback |
| *callback* | The callback to unregister |

**Returns**

bool True if the callback was unregistered.

#### 4.76.3.9 warn()

```
void Logger::warn (
            const std::string & msg,
             ...  )
```

Log a message at the level warn.

**Parameters**

| | |
|---|---|
| *msg* | A printf format string |
| ... | The variable arguments required by the printf format |

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

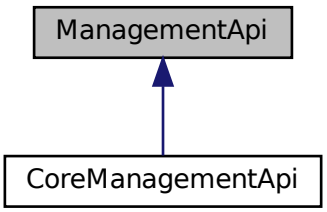
- C/common/include/logger.h
- C/common/logger.cpp

## 4.77 ManagementApi Class Reference

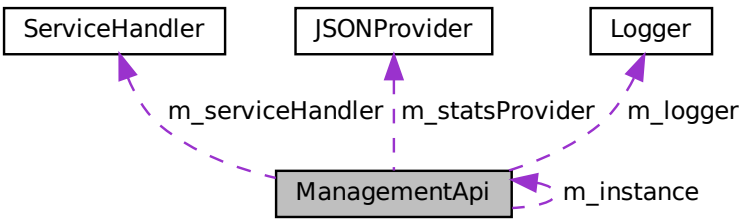
Management API server for a C++ microservice.

```
#include <management_api.h>
```

Inheritance diagram for ManagementApi:




Collaboration diagram for ManagementApi:



## Public Member Functions

- ManagementApi (const std::string &name, const unsigned short port)

  *Construct a microservices management API manager class.*
- ~ManagementApi ()

  *Management API destructor.*
- void **start** ()
- void **startServer** ()
- void **stop** ()
- void **stopServer** ()
- void registerStats (JSONProvider *statsProvider)

  *Register a statistics provider.*
- void registerProvider (JSONProvider *provider)

  *Register a generic provider.*
- void **registerService** (ServiceHandler *serviceHandler)
- unsigned short **getListenerPort** ()
- void ping (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::Request > request)

  *Received a ping request, construct a reply and return to caller.*
- void shutdown (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::Request > request)

  *Received a shutdown request, construct a reply and return to caller.*
- void configChange (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::↩ Request > request)

  *Received a config change request, construct a reply and return to caller.*
- void configChildCreate (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer↩ ::Request > request)

  *Received a children creation request, construct a reply and return to caller.*
- void configChildDelete (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer↩ ::Request > request)

  *Received a children deletion request, construct a reply and return to caller.*
- void securityChange (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::↩ Request > request)

  *Received a security change request, construct a reply and return to caller.*

## Static Public Member Functions

- static ManagementApi * getInstance ()

  *Return the signleton instance of the management interface.*

## Protected Attributes

- std::string **m_name**
- Logger * **m_logger**
- time_t **m_startTime**
- HttpServer * **m_server**
- JSONProvider * **m_statsProvider**
- ServiceHandler * **m_serviceHandler**
- std::thread * **m_thread**

## Static Protected Attributes

- static ManagementApi * **m_instance** = 0

## 4.77.1 Detailed Description

Management API server for a C++ microservice.

## 4.77.2 Constructor & Destructor Documentation

### 4.77.2.1 ManagementApi()

```
ManagementApi::ManagementApi (
            const std::string & name,
            const unsigned short port )
```

Construct a microservices management API manager class.

**Parameters**

| | |
|---|---|
| *name* | The service name |
| *port* | The management API port |

## 4.77.3 Member Function Documentation

### 4.77.3.1 configChange()

```
void ManagementApi::configChange (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Received a config change request, construct a reply and return to caller.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.77.3.2 configChildCreate()

```
void ManagementApi::configChildCreate (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Received a children creation request, construct a reply and return to caller.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.77.3.3 configChildDelete()

```
void ManagementApi::configChildDelete (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Received a children deletion request, construct a reply and return to caller.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.77.3.4 getInstance()

```
ManagementApi * ManagementApi::getInstance ( )  [static]
```

Return the signleton instance of the management interface.

Note if one has not been explicitly created then this will return 0.

#### 4.77.3.5 ping()

```
void ManagementApi::ping (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Received a ping request, construct a reply and return to caller.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.77.3.6 registerProvider()

```
void ManagementApi::registerProvider (
            JSONProvider * provider )
```

Register a generic provider.

There can be multiple providers for a single service

**Parameters**

| | |
|---|---|
| *provider* | The JSON status provider to add |

#### 4.77.3.7 registerStats()

```
void ManagementApi::registerStats (
            JSONProvider * statsProvider )
```

Register a statistics provider.

**Parameters**

| | |
|---|---|
| *statsProvider* | The statistics provider for the service |

#### 4.77.3.8 securityChange()

```
void ManagementApi::securityChange (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request )
```

Received a security change request, construct a reply and return to caller.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.77.3.9 shutdown()

```
void ManagementApi::shutdown (
```

```
std::shared_ptr< HttpServer::Response > response,
std::shared_ptr< HttpServer::Request > request )
```

Received a shutdown request, construct a reply and return to caller.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

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

- C/services/common/include/management_api.h
- C/services/common/management_api.cpp

## 4.78 ManagementClient Class Reference

The management client class used by services and tasks to communicate with the management API of the Fledge core microservice.

```
#include <management_client.h>
```

### Public Member Functions

- ManagementClient (const std::string &hostname, const unsigned short port)

  *Management Client constructor.*
- ~ManagementClient ()

  *Destructor for management client.*
- bool registerService (const ServiceRecord &service)

  *Register this service with the Fledge core.*
- bool unregisterService ()

  *Unregister this service with the Fledge core.*
- bool restartService ()

  *Restart this service by sendign a request to the Fledge core.*
- bool getService (ServiceRecord &service)

  *Get the specified service.*
- bool getServices (std::vector< ServiceRecord * > &services)

  *Return all services registered with the Fledge core.*
- bool **getServices** (std::vector< ServiceRecord * > &services, const std::string &type)
- bool registerCategory (const std::string &categoryName)

  *Register interest in a configuration category.*
- bool registerCategoryChild (const std::string &categoryName)

  *Register interest in a configuration category.*
- bool unregisterCategory (const std::string &categoryName)

  *Unregister interest in a configuration category.*
- ConfigCategories getCategories ()

  *Get the set of all configuration categories from the core micro service.*
- ConfigCategory getCategory (const std::string &categoryName)

*Return the content of the named category by calling the management API of the Fledge core.*

- std::string setCategoryItemValue (const std::string &categoryName, const std::string &itemName, const std::string &itemValue)

  *Set a category configuration item value.*
- std::string addChildCategories (const std::string &parentCategory, const std::vector< std::string > &children)

  *Add child categories to a (parent) category.*
- std::vector< AssetTrackingTuple * > & getAssetTrackingTuples (const std::string serviceName="")

  *Get the asset tracking tuples for a service or all services.*
- std::vector< StorageAssetTrackingTuple * > & getStorageAssetTrackingTuples (const std::string serviceName)

  *Get the storage asset tracking tuples for a service or all services.*
- StorageAssetTrackingTuple * getStorageAssetTrackingTuple (const std::string &serviceName, const std::string &assetName, const std::string &event, const std::string &dp, const unsigned int &c)

  *Get the asset tracking tuple for a service and asset name.*
- bool addAssetTrackingTuple (const std::string &service, const std::string &plugin, const std::string &asset, const std::string &event)

  *Add a new asset tracking tuple.*
- bool addStorageAssetTrackingTuple (const std::string &service, const std::string &plugin, const std::string &asset, const std::string &event, const bool &deprecated=false, const std::string &datapoints="", const int &count=0)

  *Add a new asset tracking tuple.*
- ConfigCategories getChildCategories (const std::string &categoryName)

  *Return child categories of a given category.*
- HttpClient * getHttpClient ()

  *Creates a HttpClient object for each thread it stores/retrieves the reference to the HttpClient and the associated thread id in a map.*
- bool addAuditEntry (const std::string &serviceName, const std::string &severity, const std::string &details)

  *Add an Audit Entry.*
- std::string & **getRegistrationBearerToken** ()
- void **setNewBearerToken** (const std::string &bearerToken)
- bool verifyBearerToken (BearerToken &token)

  *Checks and validate the JWT bearer token string.*
- bool verifyAccessBearerToken (BearerToken &bToken)

  *Checks and validate the JWT bearer token object as reference.*
- bool **verifyAccessBearerToken** (std::shared_ptr< HttpServer::Request > request)
- bool refreshBearerToken (const std::string &currentToken, std::string &newToken)

  *Refresh the JWT bearer token string.*
- std::string & **getBearerToken** ()
- bool addProxy (const std::string &serviceName, const std::string &operation, const std::string &publicEnpoint, const std::string &privateEndpoint)

  *Request that the core proxy a URL to the service.*
- bool **addProxy** (const std::string &serviceName, const std::map< std::string, std::vector< std::pair< std::string, std::string > > > &endpoints)
- bool deleteProxy (const std::string &serviceName)

  *Delete the current proxy endpoitn for the named service.*
- const std::string **getUrlbase** ()
- ACL getACL (const std::string &aclName)

  *Return the content of the named ACL by calling the management API of the Fledge core.*
- AssetTrackingTuple * getAssetTrackingTuple (const std::string &serviceName, const std::string &assetName, const std::string &event)

  *Get the asset tracking tuple for a service and asset name.*
- int validateDatapoints (std::string dp1, std::string dp2)

*Compare the datapoints to be equal or not, they can be "" enclosed.*

- AssetTrackingTable * getDeprecatedAssetTrackingTuples ()

  *Get the asset tracking tuples for all the deprecated assets.*
- std::string getAlertByKey (const std::string &key)

  *Get an alert by specific key.*
- bool raiseAlert (const std::string &key, const std::string &message, const std::string &urgency="normal")

  *Raise an alert.*
- bool clearAlert (const std::string &key)

  *Clear an alert.*
- template<class T >
  bool addCategory (const T &t, bool keepOriginalItems=false)

## 4.78.1 Detailed Description

The management client class used by services and tasks to communicate with the management API of the Fledge core microservice.

The class encapsulates the management REST API and provides methods for accessing each of those APIs.

## 4.78.2 Constructor & Destructor Documentation

### 4.78.2.1 ManagementClient()

```
ManagementClient::ManagementClient (
            const std::string & hostname,
            const unsigned short port )
```

Management Client constructor.

Creates a class used to send management API requests from a micro service to the Fledge core service.

The parameters required here are passed to new services and tasks using the –address= and –port= arguments when the service is started.

**Parameters**

| | |
|---|---|
| *hostname* | The hostname of the Fledge core micro service |
| *port* | The port of the management service API listener in the Fledge core |

## 4.78.3 Member Function Documentation

#### 4.78.3.1 addAssetTrackingTuple()

```
bool ManagementClient::addAssetTrackingTuple (
            const std::string & service,
            const std::string & plugin,
            const std::string & asset,
            const std::string & event )
```

Add a new asset tracking tuple.

**Parameters**

| | |
|---|---|
| *service* | Service name |
| *plugin* | Plugin name |
| *asset* | Asset name |
| *event* | Event type |

**Returns**

whether operation was successful

#### 4.78.3.2 addAuditEntry()

```
bool ManagementClient::addAuditEntry (
            const std::string & code,
            const std::string & severity,
            const std::string & message )
```

Add an Audit Entry.

Called when an auditable event occurs to regsiter that event.

Fledge API call example :

curl -X POST -d '{"source":"LMTR", "severity":"WARNING", "details":{"message":"Engine oil pressure low"}}' http://localhost:8081/fledge/audit

**Parameters**

| | |
|---|---|
| *code* | The log code for the entry |
| *severity* | The severity level |
| *message* | The JSON message to log |

#### 4.78.3.3 addCategory()

```
template<class T >
bool ManagementClient::addCategory (
```

```
            const T & t,
            bool keepOriginalItems = false )  [inline]
```

Note: At the time being the keep_original_items is added into payload and configuration manager in the Fledge handles it.

In the near future keep_original_items will be passed as URL modifier, i.e: 'URL?keep_original_items=true'

#### 4.78.3.4 addChildCategories()

```
string ManagementClient::addChildCategories (
            const std::string & parentCategory,
            const std::vector< std::string > & children )
```

Add child categories to a (parent) category.

**Parameters**

| | |
|---|---|
| *parentCategory* | The given category name |
| *children* | Categories to add under parent |

**Returns**

JSON string with current child categories

**Exceptions**

| | |
|---|---|
| *std::exception* | |

#### 4.78.3.5 addProxy()

```
bool ManagementClient::addProxy (
            const std::string & serviceName,
            const std::string & operation,
            const std::string & publicEndpoint,
            const std::string & privateEndpoint )
```

Request that the core proxy a URL to the service.

URL's in the public Fledge API will be forwarded to the service API of the named service.

**Parameters**

| | |
|---|---|
| *serviceName* | The name of the service to send the request to |
| *operation* | The type of operations; post, put, get or delete |
| *publicEndpoint* | The URL inthe Fledge public API to be proxied |
| *privateEnpoint* | The URL in the service API of the named service to which the reuests will be proxied. |

**Returns**

bool True if the proxy request was accepted

#### 4.78.3.6 addStorageAssetTrackingTuple()

```
bool ManagementClient::addStorageAssetTrackingTuple (
            const std::string & service,
            const std::string & plugin,
            const std::string & asset,
            const std::string & event,
            const bool & deprecated = false,
            const std::string & datapoints = "",
            const int & count = 0 )
```

Add a new asset tracking tuple.

**Parameters**

| | |
|---|---|
| *service* | Service name |
| *plugin* | Plugin name |
| *asset* | Asset name |
| *event* | Event type |
| *deprecated* | Deprecated or not |
| *datapoints* | Datapoints type |
| *count* | Count Type |

**Returns**

whether operation was successful

#### 4.78.3.7 clearAlert()

```
bool ManagementClient::clearAlert (
            const std::string & key )
```

Clear an alert.

**Parameters**

| | |
|---|---|
| *key* | Alert key |

**Returns**

whether operation was successful

#### 4.78.3.8 deleteProxy()

```
bool ManagementClient::deleteProxy (
            const std::string & serviceName )
```

Delete the current proxy endpoitn for the named service.

Normally called prior to the service shutting down.

**Parameters**

| | |
|---|---|
| *serviceName* | THe name of the service to sto the proxying for |

**Returns**

bool True if the request succeeded

#### 4.78.3.9 getACL()

```
ACL ManagementClient::getACL (
            const std::string & aclName )
```

Return the content of the named ACL by calling the management API of the Fledge core.

**Parameters**

| | |
|---|---|
| *aclName* | The name of the ACL to return |

**Returns**

ACL The ACL class

**Exceptions**

| | |
|---|---|
| *exception* | If the ACL does not exist or the JSON result can not be parsed |

#### 4.78.3.10 getAlertByKey()

```
std::string ManagementClient::getAlertByKey (
            const std::string & key )
```

Get an alert by specific key.

**Parameters**

| | |
|---|---|
| *key* | Key to get alert |

**Returns**

string Alert

#### 4.78.3.11 getAssetTrackingTuple()

```
AssetTrackingTuple * ManagementClient::getAssetTrackingTuple (
            const std::string & serviceName,
            const std::string & assetName,
            const std::string & event )
```

Get the asset tracking tuple for a service and asset name.

**Parameters**

| | |
|---|---|
| *serviceName* | The serviceName to restrict data fetch |
| *assetName* | The asset name that belongs to the service |
| *event* | The associated event type |

**Returns**

A vector of pointers to AssetTrackingTuple objects allocated on heap

#### 4.78.3.12 getAssetTrackingTuples()

```
std::vector< AssetTrackingTuple * > & ManagementClient::getAssetTrackingTuples (
            const std::string serviceName = "" )
```

Get the asset tracking tuples for a service or all services.

**Parameters**

| | |
|---|---|
| *serviceName* | The serviceName to restrict data fetch If empty records for all services are fetched |

**Returns**

A vector of pointers to AssetTrackingTuple objects allocated on heap

#### 4.78.3.13 getCategories()

```
ConfigCategories ManagementClient::getCategories ( )
```

Get the set of all configuration categories from the core micro service.

**Returns**

ConfigCategories The set of all confguration categories

#### 4.78.3.14 getCategory()

```
ConfigCategory ManagementClient::getCategory (
            const std::string & categoryName )
```

Return the content of the named category by calling the management API of the Fledge core.

**Parameters**

| | |
|---|---|
| *categoryName* | The name of the categpry to return |

**Returns**

ConfigCategory The configuration category

**Exceptions**

| | |
|---|---|
| *exception* | If the category does not exist or the result can not be parsed |

#### 4.78.3.15 getChildCategories()

```
ConfigCategories ManagementClient::getChildCategories (
            const std::string & categoryName )
```

Return child categories of a given category.

**Parameters**

| | |
|---|---|
| *categoryName* | The given category name |

**Returns**

JSON string with current child categories

**Exceptions**

| | |
|---|---|
| *std::exception* | |

#### 4.78.3.16 getDeprecatedAssetTrackingTuples()

```
AssetTrackingTable * ManagementClient::getDeprecatedAssetTrackingTuples ( )
```

Get the asset tracking tuples for all the deprecated assets.

**Returns**

A vector of pointers to AssetTrackingTuple objects allocated on heap

#### 4.78.3.17 getHttpClient()

```
HttpClient * ManagementClient::getHttpClient ( )
```

Creates a HttpClient object for each thread it stores/retrieves the reference to the HttpClient and the associated thread id in a map.

**Returns**

HttpClient The HTTP client connection to the core

#### 4.78.3.18 getService()

```
bool ManagementClient::getService (
            ServiceRecord & service )
```

Get the specified service.

Supplied with a service record that must either have the name or the type fields populated. The call will populate the other fields of the service record.

Note, if multiple service records match then only the first will be returned.

**Parameters**

| | |
|---|---|
| *service* | A partially filled service record that will be completed |

**Returns**

bool Return true if the service record was found

#### 4.78.3.19 getServices()

```
bool ManagementClient::getServices (
            std::vector< ServiceRecord * > & services )
```

Return all services registered with the Fledge core.

**Parameters**

| | |
|---|---|
| *services* | A vector of service records that will be populated |

**Returns**

bool True if the vecgtor was populated

#### 4.78.3.20 getStorageAssetTrackingTuple()

```
StorageAssetTrackingTuple * ManagementClient::getStorageAssetTrackingTuple (
            const std::string & serviceName,
            const std::string & assetName,
            const std::string & event,
            const std::string & dp,
            const unsigned int & c )
```

Get the asset tracking tuple for a service and asset name.

**Parameters**

| | |
|---|---|
| *serviceName* | The serviceName to restrict data fetch |
| *assetName* | The asset name that belongs to the service |
| *event* | The associated event type |
| *dp* | The datapoints Type |
| *c* | The count of datapoints |

**Returns**

A pointer to AssetTrackingTuple objects allocated on heap

#### 4.78.3.21 getStorageAssetTrackingTuples()

```
std::vector< StorageAssetTrackingTuple * > & ManagementClient::getStorageAssetTrackingTuples (
            const std::string serviceName )
```

Get the storage asset tracking tuples for a service or all services.

**Parameters**

| | |
|---|---|
| *serviceName* | The serviceName to restrict data fetch If empty records for all services are fetched |

**Returns**

A vector of pointers to AssetTrackingTuple objects allocated on heap

#### 4.78.3.22 raiseAlert()

```
bool ManagementClient::raiseAlert (
            const std::string & key,
            const std::string & message,
            const std::string & urgency = "normal" )
```

Raise an alert.

**Parameters**

| | |
|---|---|
| *key* | Alert key |
| *message* | Alert message |
| *urgency* | Alert urgency |

**Returns**

whether operation was successful

#### 4.78.3.23 refreshBearerToken()

```
bool ManagementClient::refreshBearerToken (
            const std::string & currentToken,
            std::string & newToken )
```

Refresh the JWT bearer token string.

**Parameters**

| | |
|---|---|
| *currentToken* | Current bearer token |
| *newToken* | New issued bearer token being set |

**Returns**

True on success, false otherwise

#### 4.78.3.24 registerCategory()

```
bool ManagementClient::registerCategory (
            const std::string & categoryName )
```

Register interest in a configuration category.

**Parameters**

| | |
|---|---|
| *category* | The name of the configuration category to register |

**Returns**

bool True if the configuration category has been registered

#### 4.78.3.25 registerCategoryChild()

```
bool ManagementClient::registerCategoryChild (
            const std::string & categoryName )
```

Register interest in a configuration category.

The service will be called with the updated configuration category whenever an item in the category is added, removed or changed.

**Parameters**

| | |
|---|---|
| *category* | The name of the category to register |

**Returns**

bool True if the registration was succesful

#### 4.78.3.26 registerService()

```
bool ManagementClient::registerService (
            const ServiceRecord & service )
```

Register this service with the Fledge core.

**Parameters**

| | |
|---|---|
| *service* | The service record of this service |

**Returns**

bool True if the service registration was sucessful

#### 4.78.3.27 restartService()

```
bool ManagementClient::restartService ( )
```

Restart this service by sendign a request to the Fledge core.

**Returns**

bool True if the service successfully requested restart

#### 4.78.3.28 setCategoryItemValue()

```
string ManagementClient::setCategoryItemValue (
            const std::string & categoryName,
            const std::string & itemName,
            const std::string & itemValue )
```

Set a category configuration item value.

**Parameters**

| | |
|---|---|
| *categoryName* | The given category name |
| *itemName* | The given item name |
| *itemValue* | The item value to set |

**Returns**

JSON string of the updated category item

**Exceptions**

| | |
|---|---|
| *std::exception* | |

#### 4.78.3.29 unregisterCategory()

```
bool ManagementClient::unregisterCategory (
            const std::string & categoryName )
```

Unregister interest in a configuration category.

The service will no longer be called when the configuration category is changed.

**Parameters**

| | |
|---|---|
| *category* | The name of the configuration category to unregister |

**Returns**

bool True if the configuration category is unregistered

#### 4.78.3.30 unregisterService()

```
bool ManagementClient::unregisterService ( )
```

Unregister this service with the Fledge core.

**Returns**

bool True if the service successfully unregistered

#### 4.78.3.31 validateDatapoints()

```
int ManagementClient::validateDatapoints (
            std::string dp1,
            std::string dp2 )
```

Compare the datapoints to be equal or not, they can be "" enclosed.

**Parameters**

| | |
|---|---|
| *dp1* | The datapoint to compare, enclosed in "" |
| *dp2* | The datapoint to compare |

**Returns**

int integer depicting result of comparison, 0 on equal

#### 4.78.3.32 verifyAccessBearerToken()

```
bool ManagementClient::verifyAccessBearerToken (
            BearerToken & token )
```

Checks and validate the JWT bearer token object as reference.

**Parameters**

| | |
|---|---|
| *request* | The bearer token object |

**Returns**

True on success, false otherwise

#### 4.78.3.33 verifyBearerToken()

```
bool ManagementClient::verifyBearerToken (
            BearerToken & bearerToken )
```

Checks and validate the JWT bearer token string.

Input token internal data will be set with new values or cached ones

**Parameters**

| | |
|---|---|
| *bearerToken* | The bearer token object |

**Returns**

True on success, false otherwise

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

- C/common/include/management_client.h
- C/common/management_client.cpp

## 4.79 MemConnectionManager Class Reference

Singleton class to manage SQLite3 Memory connection pool.

```
#include <connection_manager.h>
```

### Public Member Functions

- void **growPool** (unsigned int)
- unsigned int **shrinkPool** (unsigned int)
- Connection ∗ **allocate** ()
- void **release** (Connection ∗)
- void **shutdown** ()
- void **setError** (const char ∗, const char ∗, bool)
- PLUGIN_ERROR ∗ **getError** ()
- void **setPersist** (bool persist, const std::string &filename="")
- bool **persist** ()
- std::string **filename** ()
- void **setPurgeBlockSize** (unsigned long purgeBlockSize)

### Static Public Member Functions

- static MemConnectionManager ∗ **getInstance** ()

### 4.79.1 Detailed Description

Singleton class to manage SQLite3 Memory connection pool.

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

- C/plugins/storage/sqlitememory/include/connection_manager.h

## 4.80 NorthApi Class Reference

### Public Member Functions

- NorthApi (NorthService ∗)

  *North API class constructor.*
- ∼NorthApi ()

  *Destroy the API.*
- unsigned short getListenerPort ()

  *Return the port the service is listening on.*
- void startServer ()

  *Called on the API service thread.*
- void attachDebugger (Response response, Request request)

  *Invoke debugger attach on the north plugin.*
- void detachDebugger (Response response, Request request)

  *Invoke debugger detach on the north plugin.*
- void setDebuggerBuffer (Response response, Request request)

  *Invoke set debugger buffer size on the north plugin.*
- void getDebuggerBuffer (Response response, Request request)

  *Invoke get debugger buffer size on the north plugin.*
- void isolateDebugger (Response response, Request request)

  *Invoke isolate debugger handler on the north plugin.*
- void suspendDebugger (Response response, Request request)

  *Invoke suspend debugger handler on the north plugin.*
- void stepDebugger (Response response, Request request)

  *Invoke set debugger step command on the north plugin.*
- void replayDebugger (Response response, Request request)

  *Invoke debugger replay on the north plugin.*
- void stateDebugger (Response response, Request request)

  *Invoke debugger state on the north plugin.*

## 4.80.1 Constructor & Destructor Documentation

### 4.80.1.1 NorthApi()

```
NorthApi::NorthApi (
            NorthService * service )
```

North API class constructor.

**Parameters**

| | |
|---|---|
| *service* | The NorthService class this is the API for |

### 4.80.1.2 ~NorthApi()

```
NorthApi::~NorthApi ( )
```

Destroy the API.

Stop the service and wait for the thread to terminate.

## 4.80.2 Member Function Documentation

### 4.80.2.1 attachDebugger()

```
void NorthApi::attachDebugger (
            Response response,
            Request request )
```

Invoke debugger attach on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

### 4.80.2.2 detachDebugger()

```
void NorthApi::detachDebugger (
```

```
            Response response,
            Request request )
```

Invoke debugger detach on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.80.2.3 getDebuggerBuffer()

```
void NorthApi::getDebuggerBuffer (
            Response response,
            Request request )
```

Invoke get debugger buffer size on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.80.2.4 isolateDebugger()

```
void NorthApi::isolateDebugger (
            Response response,
            Request request )
```

Invoke isolate debugger handler on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.80.2.5 replayDebugger()

```
void NorthApi::replayDebugger (
            Response response,
            Request request )
```

Invoke debugger replay on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.80.2.6 setDebuggerBuffer()

```
void NorthApi::setDebuggerBuffer (
            Response response,
            Request request )
```

Invoke set debugger buffer size on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.80.2.7 startServer()

```
void NorthApi::startServer ( )
```

Called on the API service thread.

Start the listener for HTTP requests

#### 4.80.2.8 stateDebugger()

```
void NorthApi::stateDebugger (
            Response response,
            Request request )
```

Invoke debugger state on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.80.2.9 stepDebugger()

```
void NorthApi::stepDebugger (
```

```
            Response response,
            Request request )
```

Invoke set debugger step command on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.80.2.10 suspendDebugger()

```
void NorthApi::suspendDebugger (
            Response response,
            Request request )
```

Invoke suspend debugger handler on the north plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

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

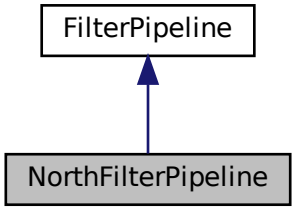
- C/services/north/include/north_api.h
- C/services/north/north_api.cpp

## 4.81 NorthFilterPipeline Class Reference

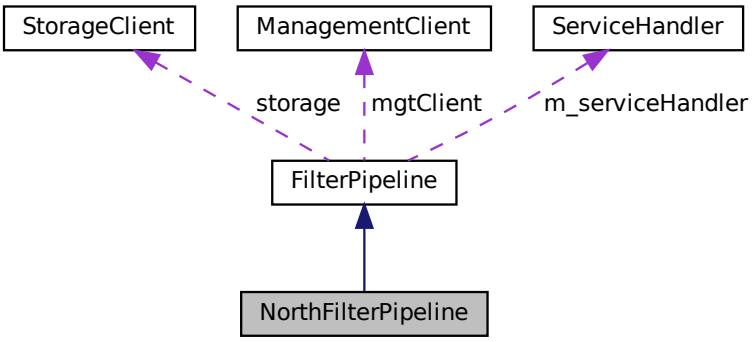
The NorthFilterPipeline class is derived from FilterPipeline class and is used to represent a pipeline of filter applicable to sending process.

```
#include <north_filter_pipeline.h>
```

Inheritance diagram for NorthFilterPipeline:



Collaboration diagram for NorthFilterPipeline:




## Public Member Functions

- NorthFilterPipeline (ManagementClient ∗mgtClient, StorageClient &storage, std::string serviceName)

  *NorthFilterPipeline class constructor.*
- bool setupFiltersPipeline (void ∗passToOnwardFilter, void ∗useFilteredData, void ∗sendingProcess)

  *Set the filter pipeline for sending process.*

## Additional Inherited Members

### 4.81.1 Detailed Description

The NorthFilterPipeline class is derived from FilterPipeline class and is used to represent a pipeline of filter applicable to sending process.

Methods are provided to load filters, setup filtering pipeline and for pipeline/filters cleanup.

### 4.81.2 Constructor & Destructor Documentation

#### 4.81.2.1 NorthFilterPipeline()

```
NorthFilterPipeline::NorthFilterPipeline (
            ManagementClient * mgtClient,
            StorageClient & storage,
            std::string serviceName )
```

NorthFilterPipeline class constructor.

This class abstracts the filter pipeline interface for sending process

**Parameters**

| | |
|---|---|
| *mgtClient* | Management client handle |
| *storage* | Storage client handle |
| *serviceName* | Name of the service to which this pipeline applies |

### 4.81.3 Member Function Documentation

#### 4.81.3.1 setupFiltersPipeline()

```
bool NorthFilterPipeline::setupFiltersPipeline (
            void * passToOnwardFilter,
            void * useFilteredData,
            void * _sendingProcess )
```

Set the filter pipeline for sending process.

This method calls the the method "plugin_init" for all loadad filters. Up to date filter configurations and Ingest filtering methods are passed to "plugin_init"

**Parameters**

| | |
|---|---|
| *passToOnwardFilter* | Ptr to function that passes data to next filter |
| *useFilteredData* | Ptr to function that gets final filtered data |
| *_sendingProcess* | The SendingProcess class handle |

**Returns**

True on success, False otherwise. @thown Any caught exception

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

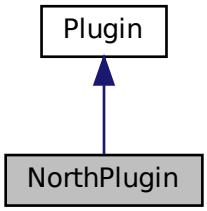
- C/tasks/north/sending_process/include/north_filter_pipeline.h
- C/tasks/north/sending_process/north_filter_pipeline.cpp

## 4.82 NorthPlugin Class Reference

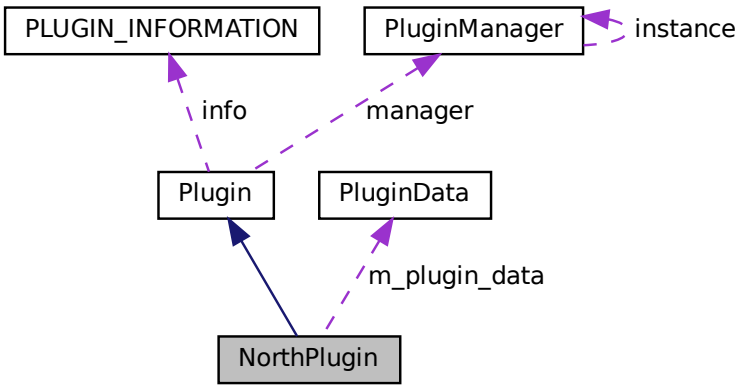
Class that represents a north plugin.

```
#include <north_plugin.h>
```

Inheritance diagram for NorthPlugin:




Collaboration diagram for NorthPlugin:




## Public Member Functions

- NorthPlugin (PLUGIN_HANDLE handle, const ConfigCategory &category)

  *Constructor for the class that wraps the north plugin.*
- uint32_t send (const std::vector< Reading * > &readings)

  *Call the send method in the plugin.*
- void reconfigure (const std::string &)

  *Call the reconfigure method in the plugin.*
- void shutdown ()

  *Call the shutdown method in the plugin.*
- bool **persistData** ()
- void start ()

  *Call the start method in the plugin with no persisted data.*
- void **startData** (const std::string &pluginData)
- std::string shutdownSaveData ()

*Call the shutdown method in the plugin and return plugin data to parsist as JSON string.*

- bool **hasControl** ()
- void pluginRegister (bool(∗write)(char ∗name, char ∗value, ControlDestination destination,...), int(∗operation)(char ∗operation, int paramCount, char ∗names[ ], char ∗parameters[ ], ControlDestination destination,...))

  *Call the plugin_register entry point of the plugin if one has been defined.*
- NorthPlugin (const PLUGIN_HANDLE handle)

  *Constructor for the class that wraps the OMF north plugin.*
- void **shutdown** ()
- std::string **shutdownSaveData** ()
- uint32_t send (const std::vector< Reading ∗ > &readings) const

  *Send vector (by reference) of readings pointer to historian server.*
- PLUGIN_HANDLE init (const ConfigCategory &config)

  *Initialise the plugin with configuration data.*
- bool **persistData** ()
- void **start** ()
- void **startData** (const std::string &pluginData)

## Public Attributes

- PluginData ∗ **m_plugin_data**

## Additional Inherited Members

## 4.82.1 Detailed Description

Class that represents a north plugin.

The purpose of this class is to hide the use of the pointers into the dynamically loaded plugin and wrap the interface into a class that can be used directly in the north subsystem.

This is achieved by having a set of private member variables which are the pointers to the functions in the plugin, and a set of public methods that will call these functions via the function pointers.

## 4.82.2 Constructor & Destructor Documentation

### 4.82.2.1 NorthPlugin() [1/2]

```
NorthPlugin::NorthPlugin (
            PLUGIN_HANDLE handle,
            const ConfigCategory & category )
```

Constructor for the class that wraps the north plugin.

Create a set of function points that resolve to the loaded plugin and enclose in the class.

### 4.82.2.2 NorthPlugin() [2/2]

```
NorthPlugin::NorthPlugin (
            const PLUGIN_HANDLE handle )
```

Constructor for the class that wraps the OMF north plugin.

Create a set of function pointers.

**Parameters**

| | |
|---|---|
| *handle* | The loaded plugin handle |

## 4.82.3 Member Function Documentation

### 4.82.3.1 init()

```
PLUGIN_HANDLE NorthPlugin::init (
            const ConfigCategory & config )
```

Initialise the plugin with configuration data.

**Parameters**

| | |
|---|---|
| *config* | The configuration data |

**Returns**

The plugin handle

### 4.82.3.2 send()

```
uint32_t NorthPlugin::send (
            const std::vector< Reading * > & readings ) const
```

Send vector (by reference) of readings pointer to historian server.

**Parameters**

| | |
|---|---|
| *readings* | The readings data |

**Returns**

The readings sent or 0 in case of any error

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

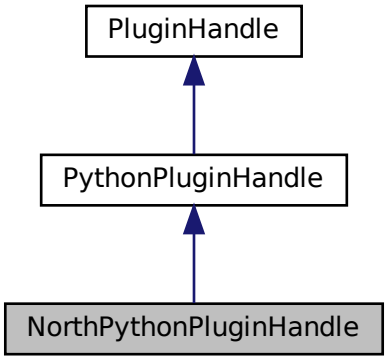
- C/services/north/include/north_plugin.h
- C/services/north/north_plugin.cpp

## 4.83 NorthPythonPluginHandle Class Reference

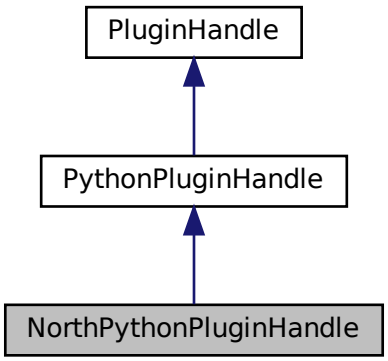
The NorthPythonPluginHandle class is used to represent an interface to a South plugin that is available as a python script.

```
#include <north_python_plugin_handle.h>
```

Inheritance diagram for NorthPythonPluginHandle:




Collaboration diagram for NorthPythonPluginHandle:




### Public Member Functions

- NorthPythonPluginHandle (const char ∗name, const char ∗path)

  *Constructor for NorthPythonPluginHandle.*

## Additional Inherited Members

### 4.83.1 Detailed Description

The NorthPythonPluginHandle class is used to represent an interface to a South plugin that is available as a python script.

### 4.83.2 Constructor & Destructor Documentation

#### 4.83.2.1 NorthPythonPluginHandle()

```
NorthPythonPluginHandle::NorthPythonPluginHandle (
            const char * pluginName,
            const char * pluginPathName )
```

Constructor for NorthPythonPluginHandle.

- Load python interface library and initialize the interface

**Parameters**

| | |
|---|---|
| *pluginName* | The Python plugin name to load |
| *pluginPathName* | The Python plugin path |

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

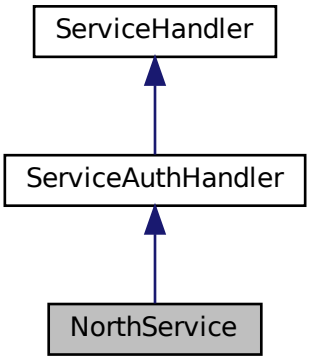
- C/services/common/include/north_python_plugin_handle.h
- C/services/common/north_python_plugin_handle.cpp

## 4.84 NorthService Class Reference

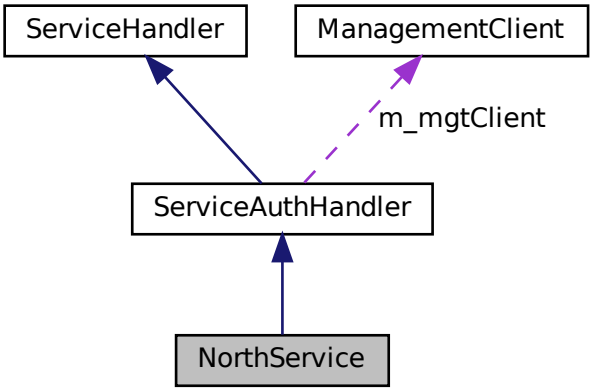
The NorthService class.

```
#include <north_service.h>
```

Inheritance diagram for NorthService:




Collaboration diagram for NorthService:




## Public Member Functions

- NorthService (const std::string &name, const std::string &token="")

  *Constructor for the north service.*
- virtual ~NorthService ()

  *Destructor for the north service.*
- void start (std::string &coreAddress, unsigned short corePort)

  *Start the north service.*
- void stop ()

  *Stop the storage service/.*

- void shutdown ()

  *Shutdown request.*
- void restart ()

  *Restart request.*
- void configChange (const std::string &, const std::string &)

  *Configuration change notification.*
- void **configChildCreate** (const std::string &, const std::string &, const std::string &)
- void **configChildDelete** (const std::string &, const std::string &)
- bool **isRunning** ()
- const std::string & **getName** ()
- const std::string & **getPluginName** ()
- void **pause** ()
- void **release** ()
- bool **write** (const std::string &name, const std::string &value, const ControlDestination)
- bool **write** (const std::string &name, const std::string &value, const ControlDestination, const std::string &arg)
- int **operation** (const std::string &name, int paramCount, char ∗names[ ], char ∗parameters[ ], const Control↩ Destination)
- int **operation** (const std::string &name, int paramCount, char ∗names[ ], char ∗parameters[ ], const Control↩ Destination, const std::string &arg)
- void **setDryRun** ()
- bool **getDryRun** ()
- void alertFailures ()

  *Raise an alert that we are having issues sending data.*
- void clearFailures ()

  *Clear the failure alert for sending data.*
- bool **attachDebugger** ()
- void **detachDebugger** ()
- void **setDebuggerBuffer** (unsigned int size)
- std::string **getDebuggerBuffer** ()
- void **suspendDebugger** (bool suspend)
- void **isolateDebugger** (bool isolate)
- void **stepDebugger** (unsigned int steps)
- void **replayDebugger** ()
- std::string debugState ()

  *Return the state of the pipeline debugger.*
- bool **debuggerAttached** ()
- bool **allowDebugger** ()

## Additional Inherited Members

### 4.84.1 Detailed Description

The NorthService class.

This class is the core of the service that provides north side services to Fledge.

### 4.84.2 Member Function Documentation

#### 4.84.2.1 alertFailures()

```
void NorthService::alertFailures ( )
```

Raise an alert that we are having issues sending data.

We also write a warning to the system log to aid with debugging

#### 4.84.2.2 clearFailures()

```
void NorthService::clearFailures ( )
```

Clear the failure alert for sending data.

We clear the alert from the status bar and write a message to the system log

#### 4.84.2.3 debugState()

```
string NorthService::debugState ( )
```

Return the state of the pipeline debugger.

**Returns**

string JSON document reporting the state of the pipeline debugger

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

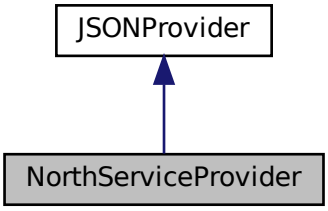
- C/services/north/include/north_service.h
- C/services/north/north.cpp

## 4.85 NorthServiceProvider Class Reference

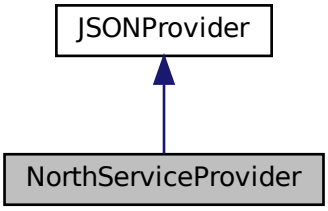
A data provider class to return data in the north service ping response.

```
#include <north_service.h>
```

Inheritance diagram for NorthServiceProvider:



Collaboration diagram for NorthServiceProvider:




### Public Member Functions

- **NorthServiceProvider** (NorthService *north)
- void **asJSON** (std::string &json) const

### 4.85.1 Detailed Description

A data provider class to return data in the north service ping response.

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

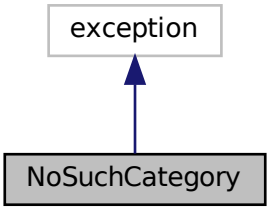
- C/services/north/include/north_service.h

## 4.86 NoSuchCategory Class Reference

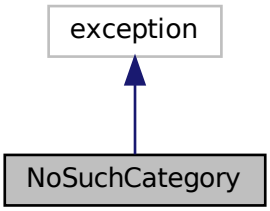
NoSuchCategory.

```
#include <configuration_manager.h>
```

Inheritance diagram for NoSuchCategory:



Collaboration diagram for NoSuchCategory:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.86.1 Detailed Description

NoSuchCategory.

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

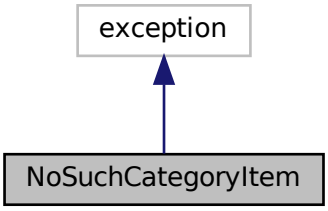
- C/services/core/include/configuration_manager.h

## 4.87 NoSuchCategoryItem Class Reference

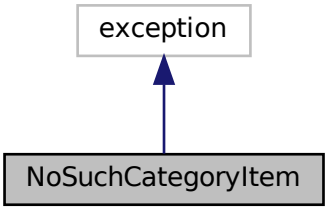
NoSuchItem.

```
#include <configuration_manager.h>
```

Inheritance diagram for NoSuchCategoryItem:



Collaboration diagram for NoSuchCategoryItem:




### Public Member Functions

- **NoSuchCategoryItem** (const std::string &message)
- virtual const char ∗ **what** () const throw ()

### 4.87.1 Detailed Description

NoSuchItem.

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

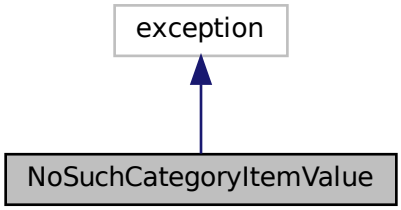
- C/services/core/include/configuration_manager.h

## 4.88 NoSuchCategoryItemValue Class Reference

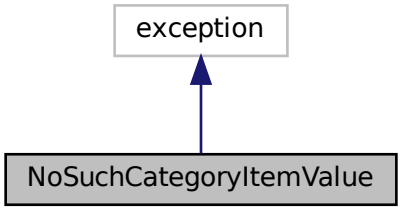
NoSuchCategoryItemValue.

```
#include <configuration_manager.h>
```

Inheritance diagram for NoSuchCategoryItemValue:



Collaboration diagram for NoSuchCategoryItemValue:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.88.1 Detailed Description

NoSuchCategoryItemValue.

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

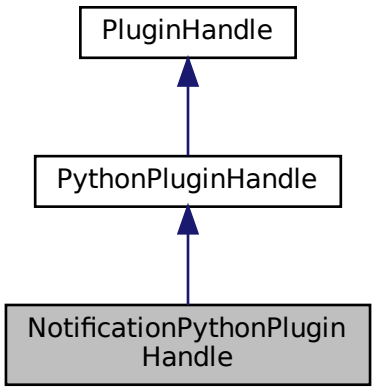
- C/services/core/include/configuration_manager.h

## 4.89 NotificationPythonPluginHandle Class Reference

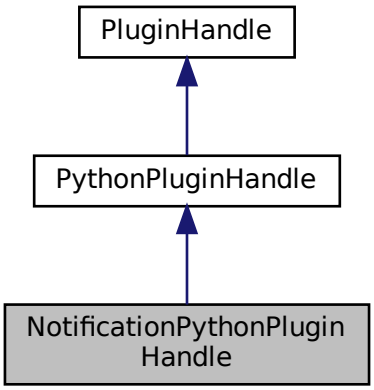
The PythonPluginHandle class is used to represent an interface to a plugin that is available as a python script.

```
#include <notification_python_plugin_handle.h>
```

Inheritance diagram for NotificationPythonPluginHandle:



Collaboration diagram for NotificationPythonPluginHandle:




## Public Member Functions

- NotificationPythonPluginHandle (const char ∗name, const char ∗path)

  *Constructor for NotificationPythonPluginHandle.*

## Additional Inherited Members

### 4.89.1 Detailed Description

The PythonPluginHandle class is used to represent an interface to a plugin that is available as a python script.

### 4.89.2 Constructor & Destructor Documentation

#### 4.89.2.1 NotificationPythonPluginHandle()

```
NotificationPythonPluginHandle::NotificationPythonPluginHandle (
            const char * pluginName,
            const char * pluginPathName )
```

Constructor for NotificationPythonPluginHandle.

- Load python interface library and initialize the interface

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

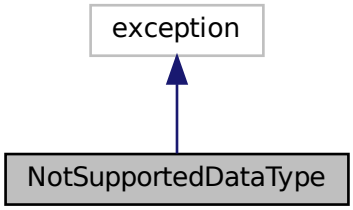
- C/services/common/include/notification_python_plugin_handle.h
- C/services/common/notification_python_plugin_handle.cpp

## 4.90 NotSupportedDataType Class Reference

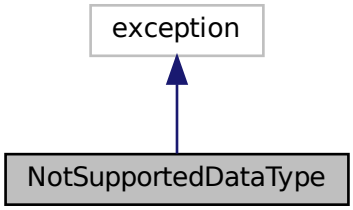
NotSupportedDataType.

```
#include <configuration_manager.h>
```

Inheritance diagram for NotSupportedDataType:




Collaboration diagram for NotSupportedDataType:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

### 4.90.1 Detailed Description

NotSupportedDataType.

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

- C/services/core/include/configuration_manager.h

## 4.91 OCS Class Reference

The OCS class.

```
#include <ocs.h>
```

### Public Member Functions

- **OCS** (const std::string &authorizationUrl)
- std::string OCSRetrieveAuthToken (const string &clientId, const string &clientSecret, bool logMessage=true)

  *Calls the ADH API to retrieve the authentication token.*
- int retrieveToken (const string &clientId, const string &clientSecret, bool logMessage=true)

  *Calls the OCS/ADH API to retrieve the authentication token related to the the clientId and clientSecret.*
- void extractToken (const string &response)

  *Extracts the OCS token from the JSON returned by the OCS API.*

### 4.91.1 Detailed Description

The OCS class.

### 4.91.2 Member Function Documentation

#### 4.91.2.1 extractToken()

```
void OCS::extractToken (
            const string & response )
```

Extracts the OCS token from the JSON returned by the OCS API.

**Parameters**

| | |
|---|---|
| *response* | JSON message generated by the OCS API containing the OCS token |

#### 4.91.2.2 OCSRetrieveAuthToken()

```
string OCS::OCSRetrieveAuthToken (
            const string & clientId,
            const string & clientSecret,
            bool logMessage = true )
```

Calls the ADH API to retrieve the authentication token.

**Parameters**

| | |
|---|---|
| *logMessage* | If true, log error messages (default: true) |

**Returns**

token Authorization token

#### 4.91.2.3 retrieveToken()

```
int OCS::retrieveToken (
            const string & clientId,
            const string & clientSecret,
            bool logMessage = true )
```

Calls the OCS/ADH API to retrieve the authentication token related to the the clientId and clientSecret.

**Parameters**

| | |
|---|---|
| *clientId* | Client Id code assigned by OCS/ADH using its GUI to the specific connection |
| *clientSecret* | Client Secret code assigned by OCS/ADH using its GUI to the specific connection |
| *logMessage* | If true, log error messages (default: true) |

**Returns**

HTTP Code

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

- C/plugins/north/OMF/include/ocs.h
- C/plugins/north/OMF/ocs.cpp

## 4.92 OMF Class Reference

The OMF class.

```
#include <omf.h>
```

### Public Member Functions

- OMF (const std::string &name, HttpSender &sender, const std::string &path, const long typeId, const std↩ ::string &producerToken)

  *Constructor: pass server URL path, OMF_type_id and producerToken.*
- **OMF** (const std::string &name, HttpSender &sender, const std::string &path, std::map< std::string, OMFDataTypes > &types, const std::string &producerToken)

- void **setOMFVersion** (std::string &omfversion)
- void **setSender** (HttpSender &sender)
- uint32_t sendToServer (const std::vector< Reading > &readings, bool skipSentDataTypes=true)

  *Send data to PI Server passing a vector of readings.*
- uint32_t sendToServer (const std::vector< Reading * > &readings, bool compression, bool skipSentData↩ Types=true)

  *Send all the readings to the PI Server.*
- uint32_t sendToServer (const Reading &reading, bool skipSentDataTypes=true)

  *Send a single reading to the PI Server.*
- uint32_t sendToServer (const Reading *reading, bool skipSentDataTypes=true)

  *Send a single reading pointer to the PI Server.*
- void setFormatType (const std::string &key, std::string &value)

  *Add the key (OMF type + OMF format) into a map.*
- void setPIServerEndpoint (const OMF_ENDPOINT PIServerEndpoint)

  *Set which PIServer component should be used for the communication.*
- void **setNamingScheme** (const NAMINGSCHEME_ENDPOINT namingScheme)
- std::string generateMeasurementId (const string &assetName)

  *Generate the container id for the given asset.*
- std::string generateSuffixType (string &assetName, long typeId)

  *Generate a suffix for the given asset in relation to the selected naming schema and the value of the type id.*
- long getNamingScheme (const string &assetName)

  *Retrieve the naming scheme for the given asset in relation to the end point selected the default naming scheme selected and the naming scheme of the asset itself.*
- string getHashStored (const string &assetName)

  *Retrieve the hash for the given asset in relation to the end point selected.*
- string getPathStored (const string &assetName)

  *Retrieve the current AF hierarchy for the given asset.*
- string getPathOrigStored (const string &assetName)

  *Retrieve the AF hierarchy in which given asset was created.*
- bool setPathStored (const string &assetName, string &afHierarchy)

  *Stores the current AF hierarchy for the given asset.*
- bool deleteAssetAFH (const string &assetName, string &path)

  *AFHierarchy - delete the link between 2 elements in the AF hierarchy.*
- bool createAssetAFH (const string &assetName, string &path)

  *AFHierarchy - create the link between 2 elements in the AF hierarchy.*
- void setDefaultAFLocation (const std::string &DefaultAFLocation)

  *Set the first level of hierarchy in Asset Framework in which the assets will be created, PI Web API only.*
- bool setAFMap (const std::string &AFMap)

  *Set the Names and Metadata rules to address where assets should be placed in the AF hierarchy.*
- void **setSendFullStructure** (const bool sendFullStructure)
- void setPrefixAFAsset (const std::string &prefixAFAsset)

  *Set the first level of hierarchy in Asset Framework in which the assets will be created, PI Web API only.*
- void **setDelimiter** (const std::string &delimiter)
- void **setDataActionCode** (const std::string &actionCode)
- std::string getFormatType (const std::string &key) const

  *Get from m_formatTypes map the key (OMF type + OMF format)*
- void setNotBlockingErrors (std::vector< std::string > &)

  *Set the list of errors considered not blocking in the communication with the PI Server.*
- std::string compress_string (const std::string &str, int compressionlevel=Z_DEFAULT_COMPRESSION)

  *Compress a string.*
- const long **getTypeId** () const

- bool isDataTypeError (const char ∗message)

  *Check for invalid/redefinition data type error.*
- bool **isPIstable** ()
- bool **isPIconnected** ()
- void **setPIconnected** (bool connectionStatus)
- std::size_t **getNumBlocks** ()
- void **setNumBlocks** (std::size_t numBlocks)
- void setMapObjectTypes (const std::vector< Reading ∗ > &data, std::map< std::string, Reading ∗ > &data↩ SuperSet)

  *Create a superset data map for each reading and found datapoints.*
- void unsetMapObjectTypes (std::map< std::string, Reading ∗ > &dataSuperSet) const

  *Cleanup the mapped object types for input data.*
- void **setStaticData** (std::vector< std::pair< std::string, std::string >> ∗staticData)
- void generateAFHierarchyPrefixLevel (string &path, string &prefix, string &AFHierarchyLevel)

  *Calculate the prefix to be used for AF objects and the last level of the hierarchies from a given AF path.*
- map< std::string, std::string > **getNamesRules** () const
- map< std::string, std::string > **getMetadataRulesExist** () const
- bool **getAFMapEmptyNames** () const
- bool **getAFMapEmptyMetadata** () const
- void **setLegacyMode** (bool legacy)

## Static Public Member Functions

- static std::string ApplyPIServerNamingRulesObj (const std::string &objName, bool ∗changed)

  *Check a PI Server object name and returns the proper name to use following the naming rules:*
- static std::string ApplyPIServerNamingRulesPath (const std::string &objName, bool ∗changed)

  *Check a PI Server path name and returns the proper name to use following the naming rules:*
- static std::string ApplyPIServerNamingRulesInvalidChars (const std::string &objName, bool ∗changed)

  *Check a PI Server name and returns the proper name to use following the naming rules.*
- static std::string variableValueHandle (const Reading &reading, std::string &AFHierarchy)

  *Evaluate the AF hierarchy provided and expand the variables in the form ${room:unknown}.*
- static bool extractVariable (string &strToHandle, string &variable, string &value, string &defaultValue)

  *Extracts a variable and its elements from a string, the variable will have the shape ${room:unknown}.*
- static void reportAsset (const string &asset, const string &level, const string &msg)

  *Report an error related to an asset if the asset has not already been reported.*

### 4.92.1 Detailed Description

The OMF class.

Implements the OMF protocol

### 4.92.2 Member Function Documentation

#### 4.92.2.1 ApplyPIServerNamingRulesInvalidChars()

```
std::string OMF::ApplyPIServerNamingRulesInvalidChars (
            const std::string & objName,
            bool * changed )  [static]
```

Check a PI Server name and returns the proper name to use following the naming rules.

Invalid chars: Control characters plus: ∗ ? ; { } [ ] | \ ‘ ’ "

**Parameters**

| | |
|---|---|
| *objName* | The object name to verify |
| *changed* | if not null, it is set to true if a change occurred |

**Returns**

Object name following the PI Server naming rules

#### 4.92.2.2 ApplyPIServerNamingRulesObj()

```
std::string OMF::ApplyPIServerNamingRulesObj (
            const std::string & objName,
            bool * changed )  [static]
```

Check a PI Server object name and returns the proper name to use following the naming rules:

- Blank names are not permitted, substituted with '_'
- Trailing spaces are removed
- Maximum name length is 200 characters.
- Valid chars
- Names cannot begin with '__', These are reserved for system use, substituted with single '_'

Note: Names on PI-Server side are not case sensitive

**Parameters**

| | |
|---|---|
| *objName* | The object name to verify |
| *changed* | if not null, it is set to true if a change occur |

**Returns**

Object name following the PI Server naming rules

Names cannot begin with '__'. These are reserved for system use.

#### 4.92.2.3 ApplyPIServerNamingRulesPath()

```
std::string OMF::ApplyPIServerNamingRulesPath (
            const std::string & objName,
            bool * changed )  [static]
```

Check a PI Server path name and returns the proper name to use following the naming rules:

- Blank names are not permitted, substituted with '_'
- Trailing spaces are removed
- Maximum name length is 200 characters.
- Valid chars
- Names cannot begin with '__', These are reserved for system use, substituted with single '_'

Names on PI-Server side are not case sensitive

**Parameters**

| | |
|---|---|
| *objName* | The object name to verify |
| *changed* | if not null, it is set to true if a change occurred |

**Returns**

Object name following the PI Server naming rules

Names cannot begin with '__'. These are reserved for system use.

#### 4.92.2.4 compress_string()

```
std::string OMF::compress_string (
            const std::string & str,
            int compressionlevel = Z_DEFAULT_COMPRESSION )
```

Compress a string.

**Parameters**

| | |
|---|---|
| *str* | Input STL string that is to be compressed |
| *compressionlevel* | zlib/gzip Compression level |

**Returns**

str gzip compressed binary data

#### 4.92.2.5 createAssetAFH()

```
bool OMF::createAssetAFH (
            const string & assetName,
            string & path )
```

AFHierarchy - create the link between 2 elements in the AF hierarchy.

#### 4.92.2.6 deleteAssetAFH()

```
bool OMF::deleteAssetAFH (
            const string & assetName,
            string & path )
```

AFHierarchy - delete the link between 2 elements in the AF hierarchy.

#### 4.92.2.7 extractVariable()

```
bool OMF::extractVariable (
            string & strToHandle,
            string & variable,
            string & value,
            string & defaultValue )  [static]
```

Extracts a variable and its elements from a string, the variable will have the shape ${room:unknown}.

**Parameters**

| | |
|---|---|
| *strToHandle* | Source string from which the variable should be extracted |
| *variable* | Variable found in the form ${room:unknown} |
| *value* | Value of the variable, left part , room in this case ${room:unknown} |
| *defaultValue* | Default value of the variable, right part , unknown in this case ${room:unknown} |

**Returns**

True a variable is found in the source string

#### 4.92.2.8 generateAFHierarchyPrefixLevel()

```
void OMF::generateAFHierarchyPrefixLevel (
            string & path,
            string & prefix,
            string & AFHierarchyLevel )
```

Calculate the prefix to be used for AF objects and the last level of the hierarchies from a given AF path.

**Parameters**

| | |
|---|---|
| *path* | Path to evaluate |
| *out/prefix* | Calculated prefix |
| *out/AFHierarchyLevel* | last level of the hierarchies evaluated form the path |

#### 4.92.2.9 generateMeasurementId()

```
std::string OMF::generateMeasurementId (
            const string & assetName )
```

Generate the container id for the given asset.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the container id should be generated |

**Returns**

Container id for the requested asset

#### 4.92.2.10 generateSuffixType()

```
std::string OMF::generateSuffixType (
            string & assetName,
            long typeId )
```

Generate a suffix for the given asset in relation to the selected naming schema and the value of the type id.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the suffix should be generated |
| *typeId* | Type id of the asset |

**Returns**

Suffix to be used for the given asset

#### 4.92.2.11 getFormatType()

```
std::string OMF::getFormatType (
            const std::string & key ) const
```

Get from m_formatTypes map the key (OMF type + OMF format)

**Parameters**

| | |
|---|---|
| *key* | The OMF type for which the format is requested |

**Returns**

The defined OMF format for the requested type

#### 4.92.2.12 getHashStored()

```
string OMF::getHashStored (
            const string & assetName )
```

Retrieve the hash for the given asset in relation to the end point selected.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the hash should be retrieved |

**Returns**

Hash of the given asset

#### 4.92.2.13 getNamingScheme()

```
long OMF::getNamingScheme (
            const string & assetName )
```

Retrieve the naming scheme for the given asset in relation to the end point selected the default naming scheme selected and the naming scheme of the asset itself.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the naming schema should be retrieved |

**Returns**

Naming schema of the given asset

#### 4.92.2.14 getPathOrigStored()

```
string OMF::getPathOrigStored (
            const string & assetName )
```

Retrieve the AF hierarchy in which given asset was created.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the path should be retrieved |

**Returns**

Path of the given asset

#### 4.92.2.15 getPathStored()

```
string OMF::getPathStored (
            const string & assetName )
```

Retrieve the current AF hierarchy for the given asset.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the path should be retrieved |

**Returns**

Path of the given asset

#### 4.92.2.16 isDataTypeError()

```
bool OMF::isDataTypeError (
            const char * message )
```

Check for invalid/redefinition data type error.

**Parameters**

| | |
|---|---|
| *message* | Server reply message for data type creation |

**Returns**

True for data type error, false otherwise

#### 4.92.2.17 reportAsset()

```
void OMF::reportAsset (
            const string & asset,
```

```
            const string & level,
            const string & msg )  [static]
```

Report an error related to an asset if the asset has not already been reported.

**Parameters**

| | |
|---|---|
| *asset* | The asset name |
| *level* | The level to log the message at |
| *msg* | The message to log |

#### 4.92.2.18 sendToServer() [1/4]

```
uint32_t OMF::sendToServer (
            const Reading & reading,
            bool skipSentDataTypes = true )
```

Send a single reading to the PI Server.

Note: this overload is never called.

**Parameters**

| | |
|---|---|
| *reading* | A reading to send |

**Returns**

1 = on success, 0 otherwise

#### 4.92.2.19 sendToServer() [2/4]

```
uint32_t OMF::sendToServer (
            const Reading * reading,
            bool skipSentDataTypes = true )
```

Send a single reading pointer to the PI Server.

Note: this overload is never called.

**Parameters**

| | |
|---|---|
| *reading* | A reading pointer to send |

**Returns**

1 = on success, 0 otherwise

#### 4.92.2.20 sendToServer() [3/4]

```
uint32_t OMF::sendToServer (
            const std::vector< Reading * > & readings,
            bool compression,
            bool skipSentDataTypes = true )
```

Send all the readings to the PI Server.

**Parameters**

| | |
|---|---|
| *readings* | A vector of readings data pointers |
| *skipSendDataTypes* | Send datatypes only once (default is true) |
| *compression* | If true, compress JSON payload before sending to PI |

**Returns**

Number of readings sent on success, 0 otherwise

Types messages sent, now transform each reading to OMF format.

After formatting the new vector of data can be sent with one message only

#### 4.92.2.21 sendToServer() [4/4]

```
uint32_t OMF::sendToServer (
            const std::vector< Reading > & readings,
            bool skipSentDataTypes = true )
```

Send data to PI Server passing a vector of readings.

Send all the readings to the PI Server.

Data sending is composed by a few phases handled by private methods.

Note: DataTypes are sent only once by using an in memory key map, being the key = assetName + typeId. Passing false to skipSentDataTypes changes the logic.

Returns the number of processed readings.

Note: this overload is never called.

**Parameters**

| | |
|---|---|
| *readings* | A vector of readings data |
| *skipSendDataTypes* | Send datatypes only once (default is true) |

**Returns**

Number of readings sent on success, 0 otherwise

#### 4.92.2.22 setFormatType()

```
void OMF::setFormatType (
            const std::string & key,
            std::string & value )
```

Add the key (OMF type + OMF format) into a map.

**Parameters**

| | |
|---|---|
| *key* | The OMF type, key of the map |
| *value* | The OMF format to set for the specific OMF type |

#### 4.92.2.23 setMapObjectTypes()

```
void OMF::setMapObjectTypes (
            const std::vector< Reading * > & data,
            std::map< std::string, Reading * > & dataSuperSet )
```

Create a superset data map for each reading and found datapoints.

The output map is filled with a Reading object containing all the datapoints found for each asset in the input reading set. The datapoints have a fake value based on the datapoint type

**Parameters**

| | |
|---|---|
| *readings* | Current input readings data |
| *dataSuperSet* | Map to store all datapoints for an assetname |

#### 4.92.2.24 setPathStored()

```
bool OMF::setPathStored (
            const string & assetName,
            string & afHierarchy )
```

Stores the current AF hierarchy for the given asset.

**Parameters**

| | |
|---|---|
| *assetName* | Asset for which the path should be retrieved |
| *afHierarchy* | Current AF hierarchy of the asset |

**Returns**

True if the operation has success

#### 4.92.2.25 unsetMapObjectTypes()

```
void OMF::unsetMapObjectTypes (
            std::map< std::string, Reading * > & dataSuperSet ) const
```

Cleanup the mapped object types for input data.

**Parameters**

| | |
|---|---|
| *dataSuperSet* | The mapped object to cleanup |

#### 4.92.2.26 variableValueHandle()

```
std::string OMF::variableValueHandle (
            const Reading & reading,
            std::string & AFHierarchy )  [static]
```

Evaluate the AF hierarchy provided and expand the variables in the form ${room:unknown}.

**Parameters**

| | |
|---|---|
| *reading* | Asset reading that should be considered from which to extract the metadata values |
| *AFHierarchy* | AF hierarchy containing the variable to be expanded |

**Returns**

True if variables were found and expanded

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

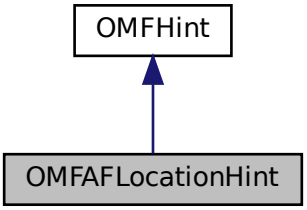
- C/plugins/north/OMF/include/omf.h
- C/plugins/north/OMF/omf.cpp

## 4.93 OMFAFLocationHint Class Reference

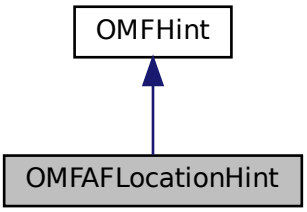
A AFLocation hint, tells us in which Asset Framework hierarchy the asset should be created.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFAFLocationHint:




Collaboration diagram for OMFAFLocationHint:




### Public Member Functions

- **OMFAFLocationHint** (const std::string &name)

### Additional Inherited Members

### 4.93.1 Detailed Description

A AFLocation hint, tells us in which Asset Framework hierarchy the asset should be created.

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

- C/plugins/north/OMF/include/OMFHint.h

## 4.94 OMFBuffer Class Reference

Buffer class designed to hold OMF payloads that can grow as required but have minimal copy semantics.

```
#include <omfbuffer.h>
```

### Public Member Functions

- OMFBuffer ()

  *Buffer class designed to hold OMF payloads that can as required but have minimal copy semantics.*
- ∼OMFBuffer ()

  *OMFBuffer destructor.*
- bool **isEmpty** ()
- void append (const char)

  *Append a character to a buffer.*
- void append (const char ∗)

  *Append a character string to a buffer.*
- void append (const int)

  *Append an integer to a buffer.*
- void append (const unsigned int)

  *Append an unsigned integer to a buffer.*
- void append (const long)

  *Append a long to a buffer.*
- void append (const unsigned long)

  *Append an unsigned long to a buffer.*
- void append (const double)

  *Append a double to a buffer.*
- void **append** (const std::string &)
- void quote (const std::string &)

  *Quote and append a string to a buffer.*
- const char ∗ coalesce ()

  *Create a coalesced buffer from the buffer chain.*
- void clear ()

  *Clear all the buffers from the OMFBuffer and allow it to be reused.*

### 4.94.1 Detailed Description

Buffer class designed to hold OMF payloads that can grow as required but have minimal copy semantics.

TODO Add a coalesce and compress public entry point

### 4.94.2 Constructor & Destructor Documentation

#### 4.94.2.1 OMFBuffer()

```
OMFBuffer::OMFBuffer ( )
```

Buffer class designed to hold OMF payloads that can as required but have minimal copy semantics.

OMFBuffer constructor

### 4.94.3 Member Function Documentation

#### 4.94.3.1 append() [1/7]

```
void OMFBuffer::append (
            const char * data )
```

Append a character string to a buffer.

@para data The string to append to the buffer

#### 4.94.3.2 append() [2/7]

```
void OMFBuffer::append (
            const char data )
```

Append a character to a buffer.

**Parameters**

| | |
|---|---|
| *data* | The character to append to the buffer |

#### 4.94.3.3 append() [3/7]

```
void OMFBuffer::append (
            const double value )
```

Append a double to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The double value to append to the buffer |

#### 4.94.3.4 append() [4/7]

```
void OMFBuffer::append (
            const int value )
```

Append an integer to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The value to append to the buffer |

#### 4.94.3.5 append() [5/7]

```
void OMFBuffer::append (
            const long value )
```

Append a long to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The long value to append to the buffer |

#### 4.94.3.6 append() [6/7]

```
void OMFBuffer::append (
            const unsigned int value )
```

Append an unsigned integer to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The unsigned long value to append to the buffer |

#### 4.94.3.7 append() [7/7]

```
void OMFBuffer::append (
            const unsigned long value )
```

Append an unsigned long to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The value to append to the buffer |

#### 4.94.3.8 coalesce()

```
const char * OMFBuffer::coalesce ( )
```

Create a coalesced buffer from the buffer chain.

The buffer returned has been created using the new[] operator and must be deleted by the caller.

**Returns**

char∗ The OMF payload in a single buffer

#### 4.94.3.9 quote()

```
void OMFBuffer::quote (
            const std::string &  )
```

Quote and append a string to a buffer.

**Parameters**

| | |
|---|---|
| *str* | The string to quote and append to the buffer |

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

- C/plugins/north/OMF/include/omfbuffer.h
- C/plugins/north/OMF/omfbuffer.cpp

## 4.95 OMFData Class Reference

The OMFData class.

```
#include <omf.h>
```

### Public Member Functions

- OMFData (OMFBuffer &payload, const Reading &reading, string measurementId, bool needDelim, const OMF_ENDPOINT PIServerEndpoint=ENDPOINT_CR, const std::string &DefaultAFLocation=std::string(), OMFHints ∗hints=NULL)

  *OMFData constructor, generates the OMF message containing the data.*
- bool **hasData** ()

### 4.95.1 Detailed Description

The OMFData class.

A reading is formatted with OMF specifications using the original type creation scheme implemented by the OMF plugin

There is no good reason to retain this class any more, it is here mostly to reduce the scope of the change when introducing the OMFBuffer

### 4.95.2 Constructor & Destructor Documentation

#### 4.95.2.1 OMFData()

```
OMFData::OMFData (
            OMFBuffer & payload,
            const Reading & reading,
            string measurementId,
            bool needDelim,
            const OMF_ENDPOINT PIServerEndpoint = ENDPOINT_CR,
            const std::string & DefaultAFLocation = std::string(),
            OMFHints * hints = NULL )
```

OMFData constructor, generates the OMF message containing the data.

**Parameters**

| | |
|---|---|
| *reading* | Reading for which the OMF message must be generated |
| *measurementId* | Name/Reference of the object of the Data Archive at which the data must be assigned |
| *PIServerEndpoint* | End point for which the OMF message must be prepared among: PIWebAPI, ADH, OCS, EDS... |
| *AFHierarchyPrefix* | Unused at the current stage |
| *hints* | OMF hints for the specific reading for changing the behaviour of the operation |

This loop creates: "dataName": {"type": "dataType"},

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

- C/plugins/north/OMF/include/omf.h
- C/plugins/north/OMF/omf.cpp

## 4.96 OMFDataTypes Class Reference

Per asset dataTypes - This class is used in a std::map where assetName is a key.

```
#include <omf.h>
```

### Public Attributes

- long **typeId**
- std::string **types**
- unsigned long **typesShort**
- long **namingScheme**
- string **afhHash**
- string **afHierarchy**
- string **afHierarchyOrig**
- unsigned short **hintChkSum**

### 4.96.1 Detailed Description

Per asset dataTypes - This class is used in a std::map where assetName is a key.

- typeId = id of the type, it is incremented if the type is redefined
- types = is a JSON string with datapoint names and types
- typesShort = a numeric representation of the type used to quickly identify if a type has changed
- namingScheme = Naming schema of the asset, valid options are Concise, Backward compatibility ..
- afhHash = Asset hash based on the AF hierarchy
- afHierarchy = Current position of the asset in the AF hierarchy
- afHierarchyOrig = Original position of the asset in the AF hierarchy, in case the asset was moved
- hintChkSum = Checksum of the OMF hints

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

- C/plugins/north/OMF/include/omf.h

## 4.97 OMFError Class Reference

An encapsulation of an error return from an OMF call.

```
#include <omferror.h>
```

## Public Member Functions

- OMFError ()

  *Constructors.*
- **OMFError** (const std::string &json)
- ~OMFError ()

  *Destructor for the error class.*
- unsigned int **messageCount** ()
- std::string getMessage (unsigned int offset)

  *Return the error message for the given message.*
- std::string getEventReason (unsigned int offset)

  *Return the error reason for the given message.*
- std::string getEventSeverity (unsigned int offset)

  *Get the event severity for a given message.*
- std::string getEventExceptionType (unsigned int offset)

  *Get the event exception type for a given message.*
- std::string getEventExceptionMessage (unsigned int offset)

  *Get the event exception message for a given message.*
- int getHttpCode ()

  *Return the most severe HTTP Code from all messages.*
- void setFromHttpResponse (const std::string &json)

  *Parse error information from an OMF POST response JSON document.*
- bool hasErrors ()

  *The error report contains at least one error level event.*
- bool hasMessages ()

  *The error report contains at least one message.*
- bool Log (const std::string &mainMessage, bool filterDuplicates=true)

  *Log all available messages.*

### 4.97.1 Detailed Description

An encapsulation of an error return from an OMF call.

The class parses the JSON response and gives access to portion of that JSON response.

### 4.97.2 Member Function Documentation

#### 4.97.2.1 getEventExceptionMessage()

```
string OMFError::getEventExceptionMessage (
            unsigned int offset )
```

Get the event exception message for a given message.

**Parameters**

| | |
|---|---|
| *offset* | The message to examine |

**Returns**

string The event exception message

#### 4.97.2.2 getEventExceptionType()

```
string OMFError::getEventExceptionType (
            unsigned int offset )
```

Get the event exception type for a given message.

**Parameters**

| | |
|---|---|
| *offset* | The message to examine |

**Returns**

string The event exception type

#### 4.97.2.3 getEventReason()

```
string OMFError::getEventReason (
            unsigned int offset )
```

Return the error reason for the given message.

**Parameters**

| | |
|---|---|
| *offset* | The error within the report to return |

**Returns**

string The event reason

#### 4.97.2.4 getEventSeverity()

```
string OMFError::getEventSeverity (
            unsigned int offset )
```

Get the event severity for a given message.

**Parameters**

| | |
|---|---|
| *offset* | The message to examine |

**Returns**

string The event severity

#### 4.97.2.5 getHttpCode()

```
int OMFError::getHttpCode ( )
```

Return the most severe HTTP Code from all messages.

PI Web API HTTP Codes are usually the same within one HTTP response.

**Returns**

HTTP Code

#### 4.97.2.6 getMessage()

```
string OMFError::getMessage (
            unsigned int offset )
```

Return the error message for the given message.

**Parameters**

| | |
|---|---|
| *offset* | The error within the report to return |

**Returns**

string The event message

#### 4.97.2.7 Log()

```
bool OMFError::Log (
            const std::string & mainMessage,
            bool filterDuplicates = true )
```

Log all available messages.

**Parameters**

| | |
|---|---|
| *mainMessage* | Top-level message when reporting an error |
| *filterDuplicates* | If true, do not log duplicate messages |

**Returns**

True if OMFError object holds at least one message

#### 4.97.2.8 setFromHttpResponse()

```
void OMFError::setFromHttpResponse (
            const std::string & json )
```

Parse error information from an OMF POST response JSON document.

**Parameters**

| | |
|---|---|
| *json* | JSON response document from an OMF POST |

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

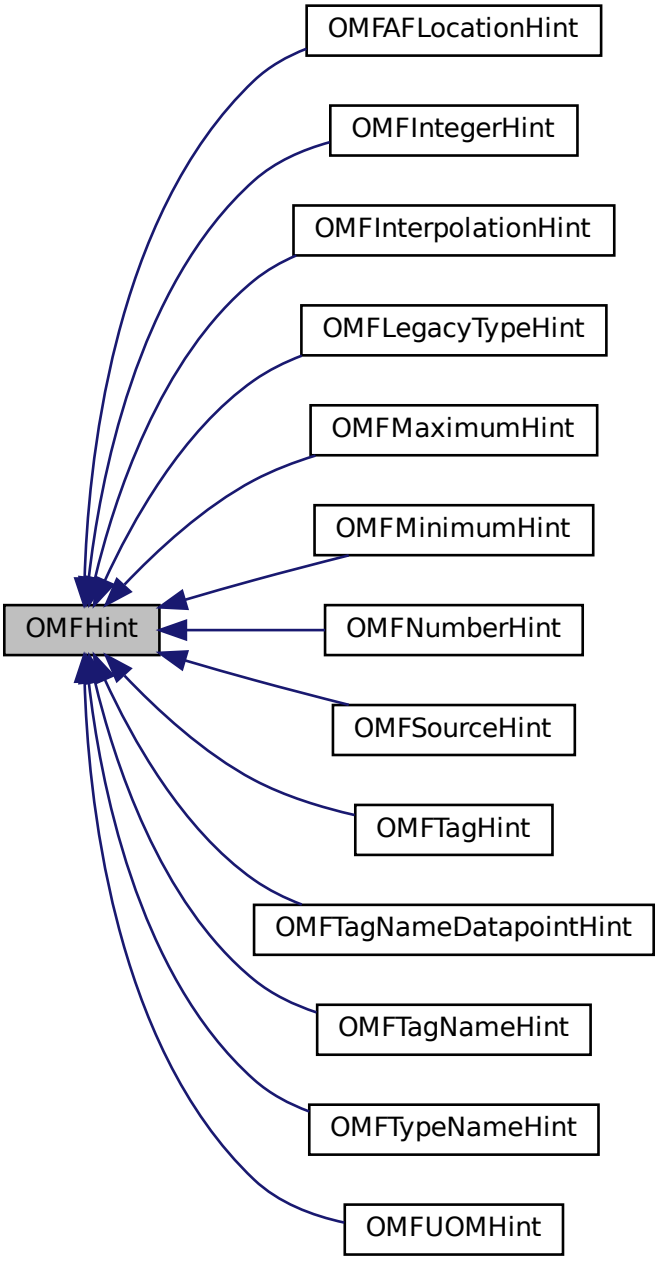
- C/plugins/north/OMF/include/omferror.h
- C/plugins/north/OMF/OMFError.cpp

## 4.98 OMFHint Class Reference

Virtual base class for an OMF Hint.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFHint:




## Public Member Functions

- const std::string & **getHint** () const

## Protected Attributes

- std::string **m_hint**

### 4.98.1 Detailed Description

Virtual base class for an OMF Hint.

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

- C/plugins/north/OMF/include/OMFHint.h

## 4.99 OMFHints Class Reference

A set of hints for a reading.

```
#include <OMFHint.h>
```

### Public Member Functions

- OMFHints (const std::string &hints)

  *Decodes the OMFhint in JSON format assigning the values to the memory structures: m_chksum, m_hints and m_datapointHints.*
- ~OMFHints ()

  *Destructor for Hints class.*
- const std::vector< OMFHint * > & **getHints** () const
- const std::vector< OMFHint * > & **getHints** (const std::string &) const
- const unsigned short **getChecksum** ()

### Static Public Member Functions

- static string getHintForChecksum (const string &hint)

  *Extracts from a complete OMF hint the part on which the checksum should be generated, for example, it will remove the section related to the AFLocation hint to avoid the creation of a new type when the value changes.*

### 4.99.1 Detailed Description

A set of hints for a reading.

documentation available at: https://fledge-iot.readthedocs.io/en/latest/plugins/fledge-filter-on
html https://fledge-iot.readthedocs.io/en/latest/OMF.html#omf-hints

### 4.99.2 Constructor & Destructor Documentation

#### 4.99.2.1 OMFHints()

```
OMFHints::OMFHints (
            const std::string & hints )
```

Decodes the OMFhint in JSON format assigning the values to the memory structures: m_chksum, m_hints and m_datapointHints.

**Parameters**

| | |
|---|---|
| *hint* | OMF hint in JSON format |

### 4.99.3 Member Function Documentation

#### 4.99.3.1 getHintForChecksum()

```
string OMFHints::getHintForChecksum (
            const string & hint )  [static]
```

Extracts from a complete OMF hint the part on which the checksum should be generated, for example, it will remove the section related to the AFLocation hint to avoid the creation of a new type when the value changes.

**Parameters**

| | |
|---|---|
| *hint* | Original/complete OMF hint |

**Returns**

OMF hint that should be considered for the calculation of the checksum

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

- C/plugins/north/OMF/include/OMFHint.h
- C/plugins/north/OMF/omfhints.cpp

## 4.100 OMFInformation Class Reference

A class that holds the configuration information for the OMF plugin.

```
#include <omfinfo.h>
```

### Public Member Functions

- OMFInformation (ConfigCategory *configData)

  *Constructor for the OMFInformation class.*
- ~OMFInformation ()

  *Destructor for the OMFInformation class.*
- void start (const std::string &storedData)

  *The plugin start entry point has been called.*
- uint32_t send (const vector< Reading * > &readings)

  *Send data to the OMF endpoint.*
- std::string saveData ()

  *Return the data to be persisted.*

### 4.100.1 Detailed Description

A class that holds the configuration information for the OMF plugin.

Note this is the first stage of refactoring the OMF plugins and represents the CONNECTOR_INFO structure of original plugin as a class

### 4.100.2 Constructor & Destructor Documentation

#### 4.100.2.1 OMFInformation()

```
OMFInformation::OMFInformation (
            ConfigCategory * configData )
```

Constructor for the OMFInformation class.

Extract host, port, path from URL

Add static data Split the string up into each pair

### 4.100.3 Member Function Documentation

#### 4.100.3.1 saveData()

```
string OMFInformation::saveData ( )
```

Return the data to be persisted.

**Returns**

string The data to persist

#### 4.100.3.2 send()

```
uint32_t OMFInformation::send (
            const vector< Reading * > & readings )
```

Send data to the OMF endpoint.

**Parameters**

| | |
|---|---|
| *readings* | The block of readings to send |

**Returns**

uint32_t The number of readings sent

#### 4.100.3.3 start()

```
void OMFInformation::start (
            const std::string & storedData )
```

The plugin start entry point has been called.

**Parameters**

| | |
|---|---|
| *storedData* | The data that has been persisted by a previous execution of the plugin |

Select the transport library based on the authentication method and transport encryption requirements.

LibcurlHttps is used to integrate Kerberos as the SimpleHttp does not support it the Libcurl integration implements only HTTPS not HTTP currently. We use SimpleHttp or SimpleHttps, as appropriate for the URL given, if not using Kerberos

The handler is allocated using "Hostname : port", connect_timeout and request_timeout. Default is no timeout

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

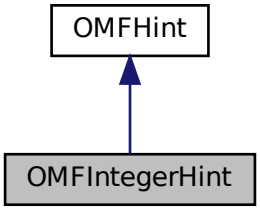
- C/plugins/north/OMF/include/omfinfo.h
- C/plugins/north/OMF/omfinfo.cpp

## 4.101 OMFIntegerHint Class Reference

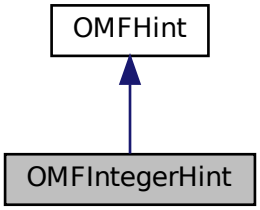
A integer hint, defines how integer type should be defined, int64, int32 or int16.

`#include <OMFHint.h>`

Inheritance diagram for OMFIntegerHint:



Collaboration diagram for OMFIntegerHint:




### Public Member Functions

- **OMFIntegerHint** (const std::string &type)

### Additional Inherited Members

### 4.101.1 Detailed Description

A integer hint, defines how integer type should be defined, int64, int32 or int16.

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

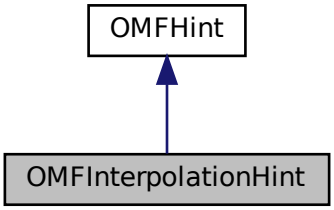
- C/plugins/north/OMF/include/OMFHint.h

## 4.102 OMFInterpolationHint Class Reference

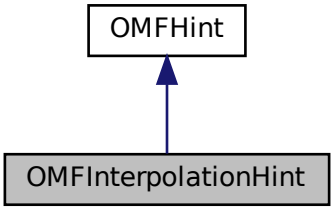
A interpolation hint, defines the interpolation value for a property.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFInterpolationHint:



Collaboration diagram for OMFInterpolationHint:




### Public Member Functions

- **OMFInterpolationHint** (const std::string &name)

### Additional Inherited Members

### 4.102.1 Detailed Description

A interpolation hint, defines the interpolation value for a property.

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

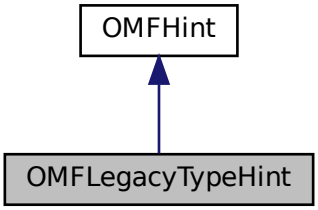
- C/plugins/north/OMF/include/OMFHint.h

## 4.103 OMFLegacyTypeHint Class Reference

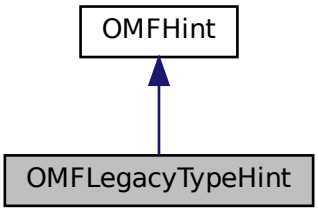
A Legacy type hint, tells the OMF plugin to send complex types for this asset.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFLegacyTypeHint:



Collaboration diagram for OMFLegacyTypeHint:




### Public Member Functions

- **OMFLegacyTypeHint** (const std::string &name)

### Additional Inherited Members

### 4.103.1 Detailed Description

A Legacy type hint, tells the OMF plugin to send complex types for this asset.

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

- C/plugins/north/OMF/include/OMFHint.h

## 4.104 OMFLinkedData Class Reference

The OMFLinkedData class.

```
#include <omflinkeddata.h>
```

### Public Member Functions

- **OMFLinkedData** (std::unordered_map< std::string, LALookup > *linkedAssetState, const OMF_ENDPOINT PIServerEndpoint=ENDPOINT_CR)
- bool processReading (OMFBuffer &payload, bool needDelim, const Reading &reading, const std::string &DefaultAFLocation=std::string(), OMFHints *hints=NULL)

  *OMFLinkedData constructor, generates the OMF message containing the data.*
- void buildLookup (const std::vector< Reading * > &reading)

  *If the entries are needed in the lookup table for this block of readings then create them.*
- void **setSendFullStructure** (const bool sendFullStructure)
- bool flushContainers (HttpSender &sender, const std::string &path, std::vector< std::pair< std::string, std::string > > &header, OMFError &error, bool *isConnected)

  *Flush the container definitions that have been built up.*
- std::size_t clearLALookup (const std::vector< Reading * > &reading, std::size_t startIndex, std::size_t numReadings, std::string &delimiter)

  *Clear selected Reading and Datapoint information from the linked asset state map.*
- void **setDelimiter** (const std::string &delimiter)
- void **setFormats** (const std::string &doubleFormat, const std::string &integerFormat)
- void **setStaticData** (std::vector< std::pair< std::string, std::string >> *staticData)

### 4.104.1 Detailed Description

The OMFLinkedData class.

A reading is formatted with OMF specifications using the linked type creation scheme supported for OMF Version 1.2 onwards.

This is based on the new mechanism discussed at AVEVA World 2022 and the mechanism is detailed in the Google Doc, https://docs.google.com/document/d/1w0e7VRqX7xzc0lEBLq-sYhgaHE0A↩BasOa6EC9dJMrMs/edit

The principle is to use links to containers in OMF with each container being a single data point in the asset. There are no specific types for the assets, they share a set of base types via these links. This should allow for readings that have different sets of datapoints for each asset.

It is also a goal of this mechanism to move away from the need to persist state data between invocations and make the process more robust.

### 4.104.2 Member Function Documentation

#### 4.104.2.1 buildLookup()

```
void OMFLinkedData::buildLookup (
            const std::vector< Reading * > & reading )
```

If the entries are needed in the lookup table for this block of readings then create them.

**Parameters**

| | |
|---|---|
| *readings* | A block of readings to process |

This loop creates the data values for each of the datapoints in the reading.

#### 4.104.2.2 clearLALookup()

```
std::size_t OMFLinkedData::clearLALookup (
            const std::vector< Reading * > & readings,
            std::size_t startIndex,
            std::size_t numReadings,
            std::string & delimiter )
```

Clear selected Reading and Datapoint information from the linked asset state map.

**Parameters**

| | |
|---|---|
| *readings* | Vector of Readings |
| *startIndex* | Start index into Readings |
| *numReadings* | Number of Readings to clear |

**Returns**

Number of asset and datapoint entries cleared

#### 4.104.2.3 flushContainers()

```
bool OMFLinkedData::flushContainers (
            HttpSender & sender,
            const std::string & path,
            std::vector< std::pair< std::string, std::string > > & header,
            OMFError & error,
            bool * isConnected )
```

Flush the container definitions that have been built up.

**Parameters**

| | |
|---|---|
| *sender* | HTTP client |
| *path* | REST server URL |
| *header* | REST call headers |
| *error* | OMFError object with parsed PI Web API HTTP response |
| *isConnected* | Set to false if REST call shows loss of connection to PI |

**Returns**

true if the containers were successfully flushed

#### 4.104.2.4 processReading()

```
bool OMFLinkedData::processReading (
            OMFBuffer & payload,
            bool needDelim,
            const Reading & reading,
            const std::string & DefaultAFLocation = std::string(),
            OMFHints * hints = NULL )
```

OMFLinkedData constructor, generates the OMF message containing the data.

**Parameters**

| | |
|---|---|
| *payload* | The buffer into which to populate the payload |
| *delim* | Add a delimiter before outputting anything |
| *reading* | Reading for which the OMF message must be generated |
| *AFHierarchyPrefix* | Unused at the current stage |
| *hints* | OMF hints for the specific reading for changing the behaviour of the operation |

This loop creates the data values for each of the datapoints in the reading.

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

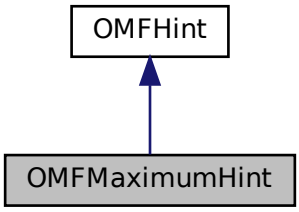
- C/plugins/north/OMF/include/omflinkeddata.h
- C/plugins/north/OMF/linkdata.cpp

## 4.105 OMFMaximumHint Class Reference

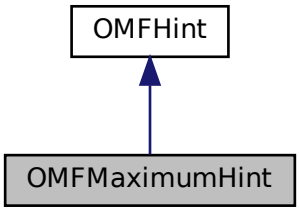
A maximum hint, defines the maximum value for a property.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFMaximumHint:




Collaboration diagram for OMFMaximumHint:




### Public Member Functions

- **OMFMaximumHint** (const std::string &name)

### Additional Inherited Members

### 4.105.1 Detailed Description

A maximum hint, defines the maximum value for a property.

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

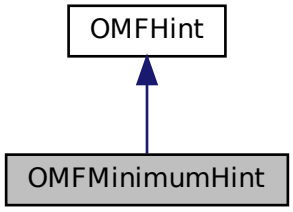
- C/plugins/north/OMF/include/OMFHint.h

## 4.106 OMFMinimumHint Class Reference

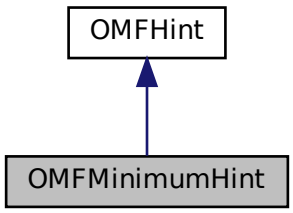
A minimum hint, defines the minimum value for a property.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFMinimumHint:




Collaboration diagram for OMFMinimumHint:




### Public Member Functions

- **OMFMinimumHint** (const std::string &name)

### Additional Inherited Members

### 4.106.1 Detailed Description

A minimum hint, defines the minimum value for a property.

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

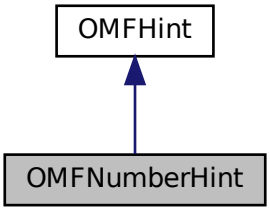
- C/plugins/north/OMF/include/OMFHint.h

## 4.107 OMFNumberHint Class Reference

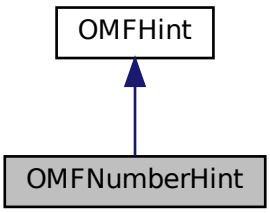
A number hint, defines how number type should be defined, float64 or float32.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFNumberHint:




Collaboration diagram for OMFNumberHint:




### Public Member Functions

- **OMFNumberHint** (const std::string &type)

### Additional Inherited Members

### 4.107.1 Detailed Description

A number hint, defines how number type should be defined, float64 or float32.

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

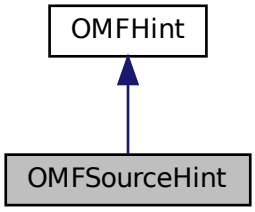
- C/plugins/north/OMF/include/OMFHint.h

## 4.108 OMFSourceHint Class Reference

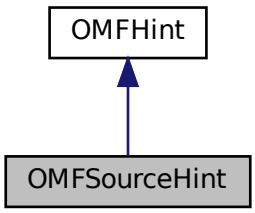
A Source hint, defines the data source for the asset or datapoint.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFSourceHint:




Collaboration diagram for OMFSourceHint:




### Public Member Functions

- **OMFSourceHint** (const std::string &name)

### Additional Inherited Members

### 4.108.1 Detailed Description

A Source hint, defines the data source for the asset or datapoint.

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

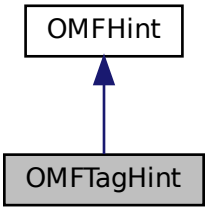
- C/plugins/north/OMF/include/OMFHint.h

## 4.109 OMFTagHint Class Reference

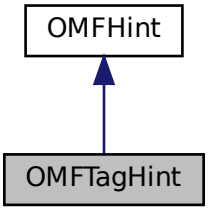
A tag hint, used to define an existing OMF container or tag to use.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFTagHint:




Collaboration diagram for OMFTagHint:




### Public Member Functions

- **OMFTagHint** (const std::string &tag)

### Additional Inherited Members

### 4.109.1 Detailed Description

A tag hint, used to define an existing OMF container or tag to use.

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

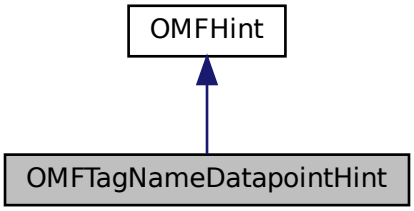
- C/plugins/north/OMF/include/OMFHint.h

## 4.110 OMFTagNameDatapointHint Class Reference

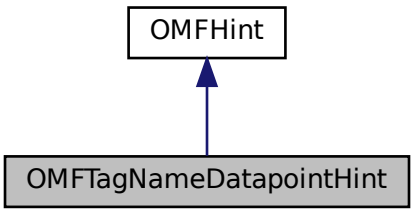
A tag name hint, defines which PI Tag to use for a Datapoint.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFTagNameDatapointHint:




Collaboration diagram for OMFTagNameDatapointHint:




### Public Member Functions

- **OMFTagNameDatapointHint** (const std::string &name)

### Additional Inherited Members

### 4.110.1 Detailed Description

A tag name hint, defines which PI Tag to use for a Datapoint.

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

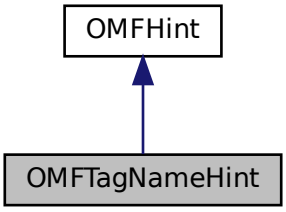
- C/plugins/north/OMF/include/OMFHint.h

## 4.111 OMFTagNameHint Class Reference

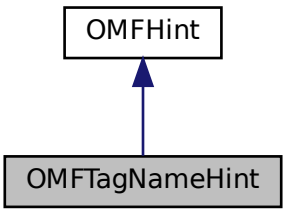
A tag name hint, tells us which Container name to use in PI.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFTagNameHint:




Collaboration diagram for OMFTagNameHint:




### Public Member Functions

- **OMFTagNameHint** (const std::string &name)

### Additional Inherited Members

### 4.111.1 Detailed Description

A tag name hint, tells us which Container name to use in PI.

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

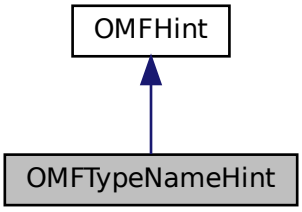
- C/plugins/north/OMF/include/OMFHint.h

## 4.112 OMFTypeNameHint Class Reference

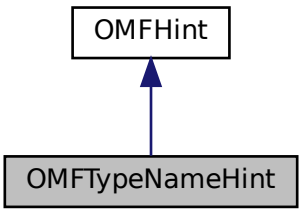
A Type name hint, tells us how to name the types we use.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFTypeNameHint:




Collaboration diagram for OMFTypeNameHint:




### Public Member Functions

- **OMFTypeNameHint** (const std::string &name)

### Additional Inherited Members

### 4.112.1 Detailed Description

A Type name hint, tells us how to name the types we use.

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

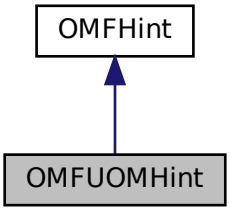
- C/plugins/north/OMF/include/OMFHint.h

## 4.113 OMFUOMHint Class Reference

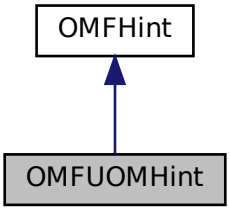
A unit of measurement hint, defines the unit of measurement for a datapoint.

```
#include <OMFHint.h>
```

Inheritance diagram for OMFUOMHint:




Collaboration diagram for OMFUOMHint:




### Public Member Functions

- **OMFUOMHint** (const std::string &name)

### Additional Inherited Members

### 4.113.1 Detailed Description

A unit of measurement hint, defines the unit of measurement for a datapoint.

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

- C/plugins/north/OMF/include/OMFHint.h

## 4.114 PerfMon Class Reference

### Public Member Functions

- PerfMon (const std::string &name)

  *Constructor for an individual performance monitor.*
- void addValue (long value)

  *Collect a new value for the performance monitor.*
- int getValues (InsertValues &values)

  *Return the performance values to insert.*

### 4.114.1 Constructor & Destructor Documentation

#### 4.114.1.1 PerfMon()

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

Constructor for an individual performance monitor.

**Parameters**

| | |
|---|---|
| *name* | The name of the performance monitor |

### 4.114.2 Member Function Documentation

#### 4.114.2.1 addValue()

```
void PerfMon::addValue (
            long value )
```

Collect a new value for the performance monitor.

**Parameters**

| | |
|---|---|
| *value* | The new value |

#### 4.114.2.2 getValues()

```
int PerfMon::getValues (
            InsertValues & values )
```

Return the performance values to insert.

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

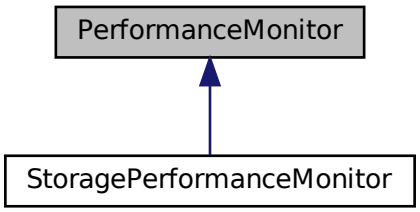
- C/services/common/include/perfmonitors.h
- C/services/common/perfmonitor.cpp

## 4.115 PerformanceMonitor Class Reference

Class to handle the performance monitors.

```
#include <perfmonitors.h>
```

Inheritance diagram for PerformanceMonitor:




### Public Member Functions

- PerformanceMonitor (const std::string &service, StorageClient *storage)

  *Constructor for the performance monitors.*
- virtual void **writeData** (const std::string &table, const InsertValues &values)
- virtual ~PerformanceMonitor ()

  *Destructor for the performance monitor.*
- void collect (const std::string &name, long value)

  *Collect a performance monitor.*
- void setCollecting (bool state)

  *Set the collection state of the performance monitors.*
- void writeThread ()

  *The thread that runs to write database values.*
- bool **isCollecting** ()

### 4.115.1 Detailed Description

Class to handle the performance monitors.

### 4.115.2 Constructor & Destructor Documentation

#### 4.115.2.1 PerformanceMonitor()

```
PerformanceMonitor::PerformanceMonitor (
            const std::string & service,
            StorageClient * storage )
```

Constructor for the performance monitors.

**Parameters**

| | |
|---|---|
| *service* | The name of the service |
| *storage* | Point to the storage client class for the service |

### 4.115.3 Member Function Documentation

#### 4.115.3.1 collect()

```
void PerformanceMonitor::collect (
            const std::string & name,
            long value )  [inline]
```

Collect a performance monitor.

**Parameters**

| | |
|---|---|
| *name* | Name of the monitor |
| *calue* | Value of the monitor |

#### 4.115.3.2 setCollecting()

```
void PerformanceMonitor::setCollecting (
            bool state )
```

Set the collection state of the performance monitors.

**Parameters**

| | |
|---|---|
| *state* | The required collection state |

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

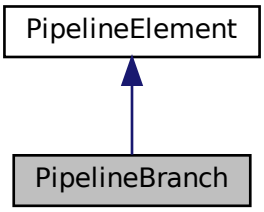
- C/services/common/include/perfmonitors.h
- C/services/common/perfmonitor.cpp

## 4.116 PipelineBranch Class Reference

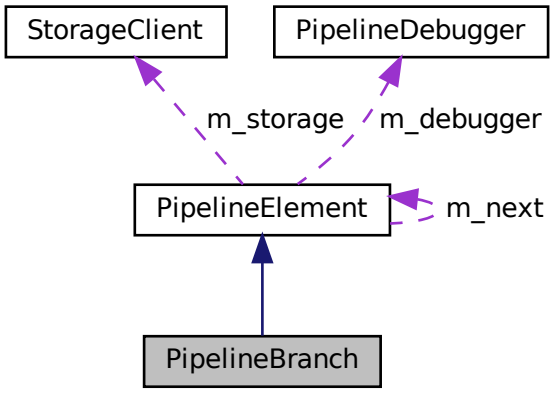
A pipeline element that represents a branch in the pipeline.

```
#include <pipeline_element.h>
```

Inheritance diagram for PipelineBranch:




Collaboration diagram for PipelineBranch:



## Public Member Functions

- PipelineBranch (FilterPipeline *parent)

  *Constructor for a branch in a filter pipeline.*
- ~PipelineBranch ()

  *Destructor for the pipeline branch.*
- void ingest (READINGSET *readingSet)

  *Ingest a set of readings and pass on in the pipeline.*
- std::string **getName** ()
- bool setupConfiguration (ManagementClient *mgtClient, std::vector< std::string > &children)

  *Setup the configuration for a branch in a pipeline.*
- bool setup (ManagementClient *mgmt, void *ingest, std::map< std::string, PipelineElement * > &categories)

  *Setup the configuration categories for the branch element of a pipeline.*
- bool init (OUTPUT_HANDLE *outHandle, OUTPUT_STREAM output)

  *Initialise the pipeline branch.*
- void shutdown (ServiceHandler *serviceHandler, ConfigHandler *configHandler)

  *Setup the configuration categories for the branch element of a pipeline.*
- bool isReady ()

  *Return if the branch is ready to be executed.*
- bool **isBranch** ()
- std::vector< PipelineElement * > & **getBranchElements** ()
- void **setFunctions** (void *onward, void *use, void *ingest)

## Additional Inherited Members

### 4.116.1 Detailed Description

A pipeline element that represents a branch in the pipeline.

### 4.116.2 Constructor & Destructor Documentation

#### 4.116.2.1 ~PipelineBranch()

```
PipelineBranch::~PipelineBranch ( )
```

Destructor for the pipeline branch.

If the pipeline is not already shutdown then shut it down Delete the thread if it exists.

### 4.116.3 Member Function Documentation

#### 4.116.3.1 ingest()

```
void PipelineBranch::ingest (
            READINGSET * readingSet )  [virtual]
```

Ingest a set of readings and pass on in the pipeline.

Create a deep copy and queue the copy into the branched pipeline.

**Parameters**

| | |
|---|---|
| *readingSet* | The set of readings to ingest |

Implements PipelineElement.

#### 4.116.3.2 init()

```
bool PipelineBranch::init (
            OUTPUT_HANDLE * outHandle,
            OUTPUT_STREAM output )  [virtual]
```

Initialise the pipeline branch.

Initialise the elements of the child pipeline Spawn a thread to excute the child pipeline.

**Parameters**

| | |
|---|---|
| *config* | The filter configuration |
| *outHandle* | The pipeline element on the "main branch" |
| *output* | |

Implements PipelineElement.

#### 4.116.3.3 setup()

```
bool PipelineBranch::setup (
            ManagementClient * mgmt,
            void * ingest,
            std::map< std::string, PipelineElement * > & categories )  [virtual]
```

Setup the configuration categories for the branch element of a pipeline.

The branch itself has no category, but it must call the setup method on all items in the child branch of the piepline.

**Parameters**

| | |
|---|---|
| *mgmt* | The management client |
| *ingest* | The configuration handler for our service |
| *filterCategories* | A map of the category names to pipeline elements |

Implements PipelineElement.

#### 4.116.3.4 setupConfiguration()

```
bool PipelineBranch::setupConfiguration (
            ManagementClient * mgtClient,
            std::vector< std::string > & children )  [virtual]
```

Setup the configuration for a branch in a pipeline.

**Parameters**

| | |
|---|---|
| *mgtClient* | The management client |
| *children* | A vector to fill with child configuration categories |

Reimplemented from PipelineElement.

#### 4.116.3.5 shutdown()

```
void PipelineBranch::shutdown (
            ServiceHandler * serviceHandler,
            ConfigHandler * configHandler )  [virtual]
```

Setup the configuration categories for the branch element of a pipeline.

The branch itself has no category, but it must call the setup method on all items in the child branch of the piepline.

**Parameters**

| | |
|---|---|
| *mgmt* | The management client |
| *ingest* | The configuration handler for our service |
| *filterCategories* | A map of the category names to pipeline elements |

Implements PipelineElement.

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

- C/common/include/pipeline_element.h
- C/common/pipeline_branch.cpp

## 4.117 PipelineDebugger Class Reference

The debugger class for elements in a pipeline.

```
#include <pipeline_debugger.h>
```

### Public Types

- enum **debuggerActions** { **NoAction**, **Block** }
- typedef enum PipelineDebugger::debuggerActions **DebuggerActions**

## Public Member Functions

- PipelineDebugger ()

  *Constructor for the pipeline element debugger.*
- ~PipelineDebugger ()

  *Destructor for the pipeline element debugger.*
- DebuggerActions process (ReadingSet *readingSet)

  *Process a reading set as it flows through the pipeline.*
- void setBuffer (unsigned int size)

  *Set the size of the circular buffer used to buffer the data flowing in the pipeline.*
- void clearBuffer ()

  *Remove the circular buffer of readings and stop the process of storing future readings.*
- std::vector< std::shared_ptr< Reading > > fetchBuffer ()

  *Fetch the current contents of the circular buffer.*

## 4.117.1 Detailed Description

The debugger class for elements in a pipeline.

## 4.117.2 Member Function Documentation

### 4.117.2.1 fetchBuffer()

```
std::vector< std::shared_ptr< Reading > > PipelineDebugger::fetchBuffer ( )
```

Fetch the current contents of the circular buffer.

A vector of shared pointers is returned to alleviate the need to copy the readings.

**Returns**

vector<shared_ptr<Reading> The readings that are returned

### 4.117.2.2 process()

```
PipelineDebugger::DebuggerActions PipelineDebugger::process (
            ReadingSet * readings )
```

Process a reading set as it flows through the pipeline.

The main purpose here is to buffer the readings in the circular buffer in order to allow later examination of the data.

**Parameters**

| | |
|---|---|
| *readings* | The reading set flowing into the pipeline element |

**Returns**

DebuggerActions Action signal to the pipeline

#### 4.117.2.3 setBuffer()

```
void PipelineDebugger::setBuffer (
            unsigned int size )
```

Set the size of the circular buffer used to buffer the data flowing in the pipeline.

**Parameters**

| | |
|---|---|
| *size* | The number of readings to buffer |

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

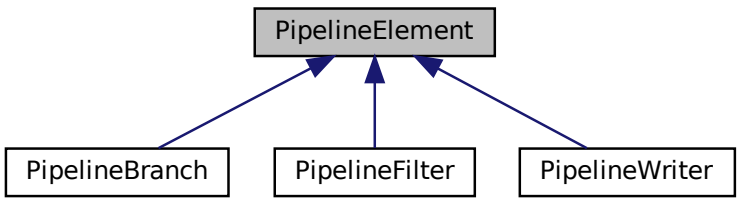
- C/common/include/pipeline_debugger.h
- C/common/pipeline_debugger.cpp

## 4.118 PipelineElement Class Reference

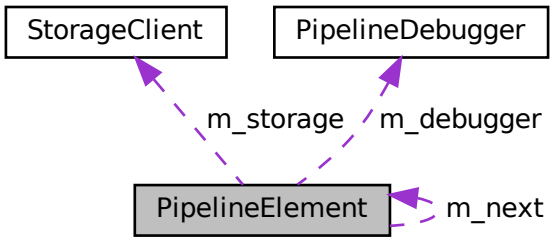
The base pipeline element class.

```
#include <pipeline_element.h>
```

Inheritance diagram for PipelineElement:



Collaboration diagram for PipelineElement:




## Public Member Functions

- void **setNext** (PipelineElement ∗next)
- PipelineElement ∗ **getNext** ()
- void **setService** (const std::string &serviceName)
- void **setStorage** (StorageClient ∗storage)
- bool attachDebugger ()

  *Attach a debugger class to the pipeline.*
- void detachDebugger ()

  *Detach a pipeline debugger from the pipeline element.*
- void setDebuggerBuffer (unsigned int size)

  *Setup the size of the debug buffer.*
- std::vector< std::shared_ptr< Reading > > getDebuggerBuffer ()

  *Fetch the content of the debugger buffer.*
- virtual bool **setupConfiguration** (ManagementClient ∗, std::vector< std::string > &)
- virtual bool **isFilter** ()
- virtual bool **isBranch** ()
- virtual void **ingest** (READINGSET ∗readingSet)=0
- virtual bool **setup** (ManagementClient ∗mgmt, void ∗ingest, std::map< std::string, PipelineElement ∗ > &categories)=0
- virtual bool **init** (OUTPUT_HANDLE ∗outHandle, OUTPUT_STREAM output)=0
- virtual void **shutdown** (ServiceHandler ∗serviceHandler, ConfigHandler ∗configHandler)=0
- virtual void **reconfigure** (const std::string &)
- virtual std::string **getName** ()=0
- virtual bool **isReady** ()=0

## Static Public Member Functions

- static void **ingest** (void ∗handle, READINGSET ∗readings)

## Protected Attributes

- std::string **m_serviceName**
- PipelineElement ∗ **m_next**
- StorageClient ∗ **m_storage**
- PipelineDebugger ∗ **m_debugger**

### 4.118.1 Detailed Description

The base pipeline element class.

### 4.118.2 Member Function Documentation

#### 4.118.2.1 attachDebugger()

```
bool PipelineElement::attachDebugger ( )
```

Attach a debugger class to the pipeline.

**Returns**

bool True if a debugger is attached to the element

#### 4.118.2.2 getDebuggerBuffer()

```
vector< shared_ptr< Reading > > PipelineElement::getDebuggerBuffer ( )
```

Fetch the content of the debugger buffer.

**Returns**

vector<shared_ptr<ReadingSet>> The current contents of the debugger buffer

#### 4.118.2.3 setDebuggerBuffer()

```
void PipelineElement::setDebuggerBuffer (
            unsigned int size )
```

Setup the size of the debug buffer.

**Parameters**

| | |
|---|---|
| *size* | Number of readings to buffer |

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

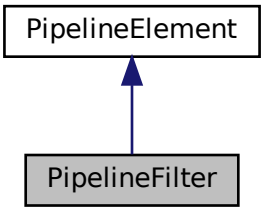
- C/common/include/pipeline_element.h
- C/common/pipeline_element.cpp

## 4.119 PipelineFilter Class Reference

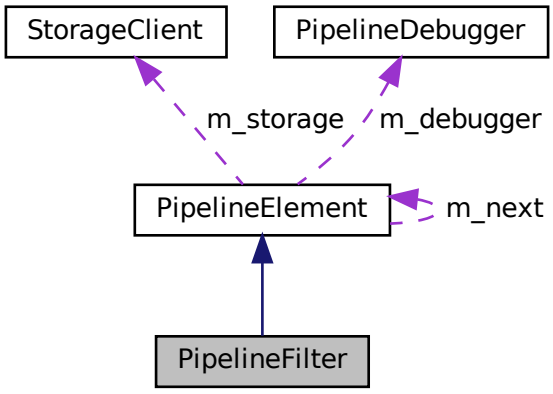
A pipeline element the runs a filter plugin.

```
#include <pipeline_element.h>
```

Inheritance diagram for PipelineFilter:




Collaboration diagram for PipelineFilter:




### Public Member Functions

- PipelineFilter (const std::string &name, const ConfigCategory &filterDetails)

  *Construct the PipelineFilter class.*
- ~PipelineFilter ()

  *Destructor for the pipeline filter element.*
- bool setupConfiguration (ManagementClient *mgtClient, std::vector< std::string > &children)

  *Setup the configuration for a filter in a pipeline.*
- void **ingest** (READINGSET *readingSet)

- bool setup (ManagementClient ∗mgmt, void ∗ingest, std::map< std::string, PipelineElement ∗ > &categories)

  *Setup the configuration categories for the filter element in a pipeline.*
- bool init (OUTPUT_HANDLE ∗outHandle, OUTPUT_STREAM output)

  *Initialise the pipeline filter ready for ingest of data.*
- void shutdown (ServiceHandler ∗serviceHandler, ConfigHandler ∗configHandler)

  *Shutdown a pipeline element that is a filter.*
- void reconfigure (const std::string &newConfig)

  *Reconfigure method.*
- bool **isFilter** ()
- std::string **getCategoryName** ()
- bool **persistData** ()
- void **setPluginData** (PluginData ∗data)
- std::string **getPluginData** ()
- void **setServiceName** (const std::string &name)
- std::string **getName** ()
- bool **isReady** ()

## Additional Inherited Members

### 4.119.1 Detailed Description

A pipeline element the runs a filter plugin.

### 4.119.2 Constructor & Destructor Documentation

#### 4.119.2.1 PipelineFilter()

```
PipelineFilter::PipelineFilter (
            const std::string & name,
            const ConfigCategory & filterDetails )
```

Construct the PipelineFilter class.

This is the specialisation of the PipelineElement that represents a running filter in the pipeline.

### 4.119.3 Member Function Documentation

#### 4.119.3.1 init()

```
bool PipelineFilter::init (
            OUTPUT_HANDLE * outHandle,
            OUTPUT_STREAM output )  [virtual]
```

Initialise the pipeline filter ready for ingest of data.

**Parameters**

| | |
|---|---|
| *outHandle* | The pipeline element we are sending the data to |
| *output* | |

Implements PipelineElement.

#### 4.119.3.2 setup()

```
bool PipelineFilter::setup (
            ManagementClient * mgmt,
            void * ingest,
            std::map< std::string, PipelineElement * > & categories )  [virtual]
```

Setup the configuration categories for the filter element in a pipeline.

**Parameters**

| | |
|---|---|
| *mgmt* | The Management client |
| *ingest* | The service handler for our service |
| *filterCatiegories* | A map of the category name to pipeline element |

Implements PipelineElement.

#### 4.119.3.3 setupConfiguration()

```
bool PipelineFilter::setupConfiguration (
            ManagementClient * mgtClient,
            std::vector< std::string > & children )  [virtual]
```

Setup the configuration for a filter in a pipeline.

**Parameters**

| | |
|---|---|
| *mgtClient* | The managament client |
| *children* | A vector to fill with child configuration categories |

Reimplemented from PipelineElement.

#### 4.119.3.4 shutdown()

```
void PipelineFilter::shutdown (
```

```
ServiceHandler * serviceHandler,
ConfigHandler * configHandler )  [virtual]
```

Shutdown a pipeline element that is a filter.

Remove registration for categories of interest, persist and plugin data that needs persisting and call shutdown on the plugin itself.

**Parameters**

| | |
|---|---|
| *serviceHandler* | The service handler of the service that is hostign the pipeline |
| *configHandler* | The config handler for the service from which we unregister |

Implements PipelineElement.

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

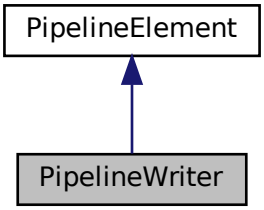
- C/common/include/pipeline_element.h
- C/common/pipeline_filter.cpp

## 4.120 PipelineWriter Class Reference

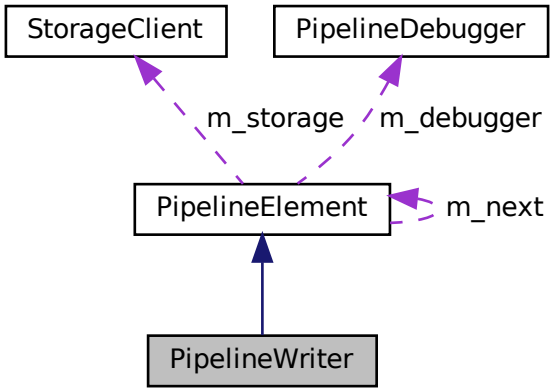
A pipeline element that writes to a storage service or buffer.

```
#include <pipeline_element.h>
```

Inheritance diagram for PipelineWriter:



Collaboration diagram for PipelineWriter:




## Public Member Functions

- PipelineWriter ()

  *Constructor for the pipeline writer, the element that sits at the end of every pipeline and branch.*
- std::string **getName** ()
- void ingest (READINGSET *readingSet)

  *Ingest into a pipeline writer.*
- bool setup (ManagementClient *mgmt, void *ingest, std::map< std::string, PipelineElement * > &categories)

  *Setup the pipeline writer.*
- bool init (OUTPUT_HANDLE *outHandle, OUTPUT_STREAM output)

  *Initialise the pipeline writer.*
- void shutdown (ServiceHandler *serviceHandler, ConfigHandler *configHandler)

  *Shutdown the pipeline writer.*
- bool isReady ()

  *Return if the pipeline writer is ready to receive data.*

## Additional Inherited Members

### 4.120.1 Detailed Description

A pipeline element that writes to a storage service or buffer.

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

- C/common/include/pipeline_element.h
- C/common/pipeline_writer.cpp

## 4.121 PIWebAPI Class Reference

The PIWebAPI class.

```
#include <piwebapi.h>
```

### Public Member Functions

- void **setAuthMethod** (std::string &authMethod)
- void **setAuthBasicCredentials** (std::string &authBasicCredentials)
- int GetVersion (const string &host, string &version, bool logMessage=true)

  *Calls the PI Web API to retrieve the version.*
- string ExtractVersion (const string &response)

  *Extracts the PIWebAPI version from the JSON returned by the PIWebAPI api.*
- string errorMessageHandler (const string &msg)

  *Handles PI Web API error message considering the possible cases:*

### 4.121.1 Detailed Description

The PIWebAPI class.

### 4.121.2 Member Function Documentation

#### 4.121.2.1 errorMessageHandler()

```
string PIWebAPI::errorMessageHandler (
            const string & msg )
```

Handles PI Web API error message considering the possible cases:

- removes all the LF CR and extracts spaces
- substitutes a message with a different one using an hardcoded vector
- in the case of the presence of an HTTP code adds the corresponding text using the Simple-Web-Server functionalities
- Handles PI Web API json error message extracting significant parts to produce a meaningful and concise message

**Parameters**

| | |
|---|---|
| *json* | JSON message generated by PIWebAPI containing the error |

**Returns**

The concise and meaningful error message

#### 4.121.2.2 ExtractVersion()

```
std::string PIWebAPI::ExtractVersion (
            const string & response )
```

Extracts the PIWebAPI version from the JSON returned by the PIWebAPI api.

**Parameters**

| | |
|---|---|
| *response* | JSON message generated by the PIWebAPI API containing the version |

**Returns**

The version of the PIWebAPI server

#### 4.121.2.3 GetVersion()

```
int PIWebAPI::GetVersion (
            const string & host,
            string & version,
            bool logMessage = true )
```

Calls the PI Web API to retrieve the version.

**Parameters**

| | |
|---|---|
| *host* | Reference of the server running PI Web API in the format: hostName + ":" + port |
| *version* | The returned version string of the PI Web API server |
| *logMessage* | If true, log an error message if there is a failure (default: true) |

**Returns**

HTTP response status code

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

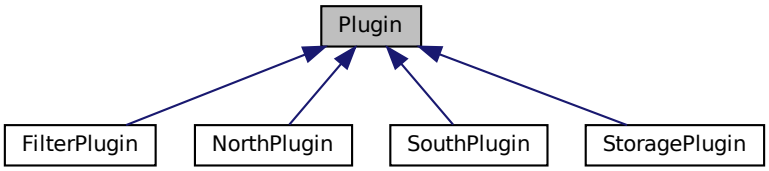
- C/plugins/common/include/piwebapi.h
- C/plugins/common/piwebapi.cpp

## 4.122 Plugin Class Reference

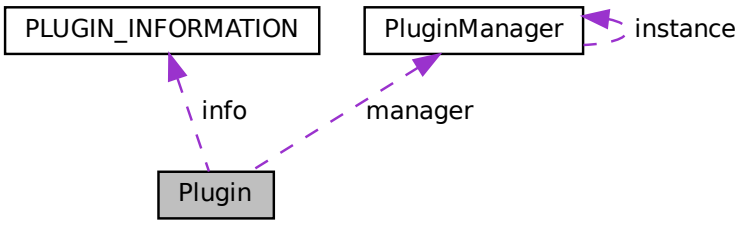
A generic representation of a plugin.

```
#include <plugin.h>
```

Inheritance diagram for Plugin:




Collaboration diagram for Plugin:




### Public Member Functions

- **Plugin** (PLUGIN_HANDLE handle)
- const PLUGIN_INFORMATION * **getInfo** ()
- PLUGIN_HANDLE **getHandle** ()

### Protected Attributes

- PLUGIN_HANDLE **handle**
- PluginManager * **manager**
- PLUGIN_INFORMATION * **info**

### 4.122.1 Detailed Description

A generic representation of a plugin.

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

- C/services/common/include/plugin.h
- C/services/common/plugin.cpp

## 4.123 PLUGIN_ERROR Struct Reference

Structure used by plugins to return error information.

```
#include <plugin_api.h>
```

### Public Attributes

- char * **message**
- char * **entryPoint**
- bool **retryable**

### 4.123.1 Detailed Description

Structure used by plugins to return error information.

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

- C/services/common/include/plugin_api.h

## 4.124 PLUGIN_INFORMATION Struct Reference

The plugin infiornation structure, used to return information from a plugin during the laod and configuration stage.

```
#include <plugin_api.h>
```

### Public Attributes

- const char * name

  *The name of the plugin.*
- const char * version

  *The release version of the plugin.*
- unsigned int options

  *The set of option flags that apply to this plugin.*
- const char * type

  *The plugin type, this is one of storage, south, filter, north, notificationRule or notificationDelivery.*
- const char * interface

  *The interface version of this plugin.*
- const char * config

  *The default JSON configuration category for this plugin.*

### 4.124.1 Detailed Description

The plugin infiornation structure, used to return information from a plugin during the laod and configuration stage.

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

- C/services/common/include/plugin_api.h

## 4.125 plugin_parameter Struct Reference

Pass a name/value pair to a plugin.

```
#include <plugin_api.h>
```

### Public Attributes

- std::string **name**
- std::string **value**

### 4.125.1 Detailed Description

Pass a name/value pair to a plugin.

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

- C/services/common/include/plugin_api.h

## 4.126 PluginData Class Reference

### Public Member Functions

- PluginData (StorageClient *client)

  *PluginData constructor.*
- std::string loadStoredData (const std::string &key)

  *Load stored data for a given key.*
- bool persistPluginData (const std::string &key, const std::string &data, const std::string &service_name)

  *Store plugin data for a given key.*

### 4.126.1 Constructor & Destructor Documentation

#### 4.126.1.1 PluginData()

```
PluginData::PluginData (
            StorageClient * client )
```

PluginData constructor.

**Parameters**

| | |
|---|---|
| *client* | StorageClient pointer |

## 4.126.2 Member Function Documentation

### 4.126.2.1 loadStoredData()

```
string PluginData::loadStoredData (
            const std::string & key )
```

Load stored data for a given key.

**Parameters**

| | |
|---|---|
| *key* | Given key for data load |

**Returns**

JSON string with found data or empty JSON data

### 4.126.2.2 persistPluginData()

```
bool PluginData::persistPluginData (
            const std::string & key,
            const std::string & data,
            const std::string & service_name )
```

Store plugin data for a given key.

**Parameters**

| | |
|---|---|
| *key* | The given key |
| *data* | The JSON data to save (as string) |
| *service_name* | The name of service |

**Returns**

true on success, false otherwise.

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

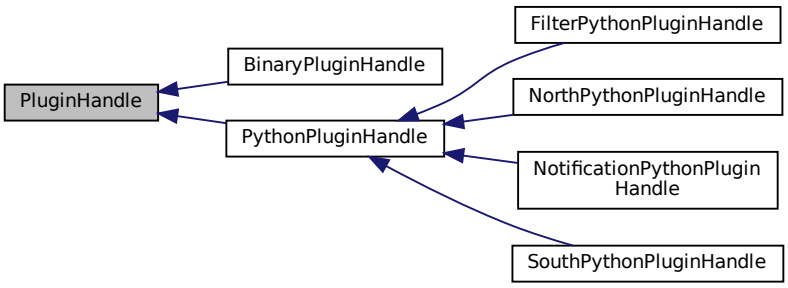
- C/common/include/plugin_data.h
- C/common/plugin_data.cpp

## 4.127 PluginHandle Class Reference

The PluginHandle class is used to represent an opaque handle to a plugin instance.

```
#include <plugin_handle.h>
```

Inheritance diagram for PluginHandle:




### Public Member Functions

- virtual void ∗ **GetInfo** ()=0
- virtual void ∗ **ResolveSymbol** (const char ∗sym)=0
- virtual void ∗ **getHandle** ()=0

### 4.127.1 Detailed Description

The PluginHandle class is used to represent an opaque handle to a plugin instance.

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

- C/services/common/include/plugin_handle.h

## 4.128 PluginManager Class Reference

The manager for plugins.

```
#include <plugin_manager.h>
```

Collaboration diagram for PluginManager:

## Public Member Functions

- PLUGIN_HANDLE loadPlugin (const std::string &name, const std::string &type)

  *Load a given plugin.*
- void **unloadPlugin** (PLUGIN_HANDLE handle)
- void ∗ resolveSymbol (PLUGIN_HANDLE handle, const std::string &symbol)

  *Resolve a symbol within the plugin.*
- PLUGIN_HANDLE findPluginByName (const std::string &name)

  *Find a loaded plugin by name.*
- PLUGIN_HANDLE findPluginByType (const std::string &type)

  *Find a loaded plugin by type.*
- PLUGIN_INFORMATION ∗ getInfo (const PLUGIN_HANDLE)

  *Return the information for a named plugin.*
- void getInstalledPlugins (const std::string &type, std::list< std::string > &plugins)

  *Get the installed plugins in the given plugin type subdirectory of "plugins" under FLEDGE_ROOT Plugin type is one of: south, north, filter, notificationRule, notificationDelivery.*
- void setPluginType (tPluginType type)

  *Set Plugin Type.*
- PLUGIN_TYPE **getPluginImplType** (const PLUGIN_HANDLE hndl)
- std::vector< std::string > getPluginsByFlags (const std::string &type, unsigned int flags)

  *Return a list of plugins matching the criteria of plugin type and plugin flags.*

## Static Public Member Functions

- static PluginManager ∗ getInstance ()

  *PluginManager Singleton implementation.*

## Static Public Attributes

- static PluginManager ∗ **instance** = 0

### 4.128.1 Detailed Description

The manager for plugins.

This manager is a singleton and is responsible for loading, tracking and unloading the plugins within the system.

### 4.128.2 Member Function Documentation

#### 4.128.2.1 getInstalledPlugins()

```
void PluginManager::getInstalledPlugins (
            const std::string & type,
            std::list< std::string > & plugins )
```

Get the installed plugins in the given plugin type subdirectory of "plugins" under FLEDGE_ROOT Plugin type is one of: south, north, filter, notificationRule, notificationDelivery.

**Parameters**

| | |
|---|---|
| *type* | The plugin type |
| *plugins* | The output plugin list name to fill |

Get all sub directory names in path: path = plugins/filter/ delta scale

Plugin filename is libdelta.so, libscale.so Plugin name is the subdirecory name in path Skip directory starting with '_' or with name 'common'

#### 4.128.2.2 getPluginsByFlags()

```
std::vector< string > PluginManager::getPluginsByFlags (
            const std::string & type,
            unsigned int flags )
```

Return a list of plugins matching the criteria of plugin type and plugin flags.

**Parameters**

| | |
|---|---|
| *type* | The plugin type to match |
| *flags* | A bitmask of flags to match |

**Returns**

vector<string> A list of matching plugin names

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

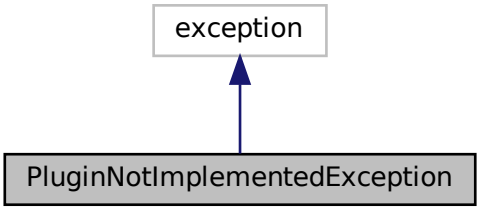
- C/services/common/include/plugin_manager.h
- C/services/common/plugin_manager.cpp

## 4.129 PluginNotImplementedException Class Reference

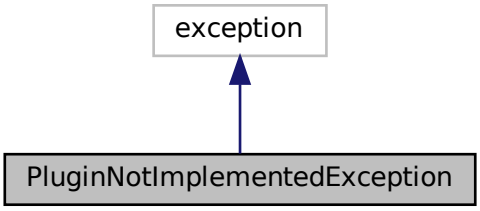
Implementation of PluginNotImplementedException.

```
#include <plugin_exception.h>
```

Inheritance diagram for PluginNotImplementedException:



Collaboration diagram for PluginNotImplementedException:




### Public Member Functions

- **PluginNotImplementedException** (const char ∗error="Functionality not implemented yet!")
- const char ∗ **what** () const noexcept

### 4.129.1 Detailed Description

Implementation of PluginNotImplementedException.

This exception should be thrown when a feature is not implemented yet.

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

- C/services/common/include/plugin_exception.h

## 4.130 ProfileItem Class Reference

### Public Member Functions

- **ProfileItem** (const std::string &reference)
- void **complete** ()
- unsigned long **getDuration** ()
- const std::string & **getReference** () const
- const std::string & **getTs** () const
- **ProfileItem** (const std::string &reference)
- void **complete** ()
- unsigned long **getDuration** ()
- const std::string & **getReference** () const
- const std::string & **getTs** () const

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

- C/plugins/storage/sqlite/include/profile.h

## 4.131 PurgeConfiguration Class Reference

### Public Member Functions

- void exclude (const std::string &asset)

  *Add an asset to the exclusion list.*
- bool **hasExclusions** ()
- bool isExcluded (const std::string &asset)

  *Check if the named asset appears in the exclusion list.*
- void minimumRetained (uint32_t minimum)

  *Set the minimum number of rows to retian for each asset.*
- uint32_t **getMinimumRetained** ()

### Static Public Member Functions

- static PurgeConfiguration * getInstance ()

  *Return the singleton instance of the PurgeConfiguration class for this plugin.*

### 4.131.1 Member Function Documentation

#### 4.131.1.1 exclude()

```
void PurgeConfiguration::exclude (
            const std::string & asset )
```

Add an asset to the exclusion list.

**Parameters**

| | |
|---|---|
| *asset* | the asset to add to the exclusion list |

#### 4.131.1.2 getInstance()

```
PurgeConfiguration * PurgeConfiguration::getInstance ( )  [static]
```

Return the singleton instance of the PurgeConfiguration class for this plugin.

**Returns**

PurgeConfiguration* singleton instance

#### 4.131.1.3 isExcluded()

```
bool PurgeConfiguration::isExcluded (
            const std::string & asset )
```

Check if the named asset appears in the exclusion list.

**Parameters**

| | |
|---|---|
| *asset* | Asset to check for exclusion |

**Returns**

True if the asset is excluded

#### 4.131.1.4 minimumRetained()

```
void PurgeConfiguration::minimumRetained (
            uint32_t minimum )
```

Set the minimum number of rows to retian for each asset.

**Parameters**

| | |
|---|---|
| *minimum* | Minimum number of rows to retain |

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

- C/plugins/storage/sqlite/common/include/purge_configuration.h
- C/plugins/storage/sqlite/common/purge_configuration.cpp

## 4.132 PurgeResult Class Reference

### Public Member Functions

- PurgeResult (const std::string &json)

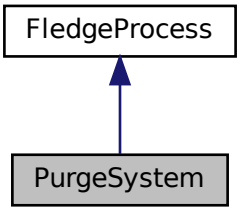
  *Construct a purge result from a JSON document returned from the Fledge storage service.*
- unsigned long **getRemoved** () const
- unsigned long **getUnsentPurged** () const
- unsigned long **getUnsentRetained** () const
- unsigned long **getRemaining** () const

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

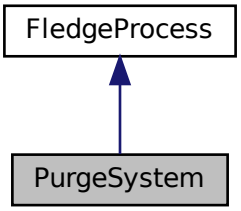
- C/common/include/purge_result.h
- C/common/purge_result.cpp

## 4.133 PurgeSystem Class Reference

Inheritance diagram for PurgeSystem:




Collaboration diagram for PurgeSystem:




### Public Member Functions

- PurgeSystem (int argc, char **argv)

  *Constructor for Purge system.*
- void run ()

  *PurgeSystem run method, called by the base class to start the process and do the actual work.*

### Additional Inherited Members

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

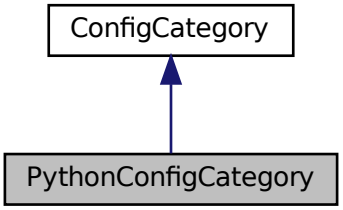
- C/tasks/purge_system/include/purge_system.h
- C/tasks/purge_system/purge_system.cpp

## 4.134 PythonConfigCategory Class Reference

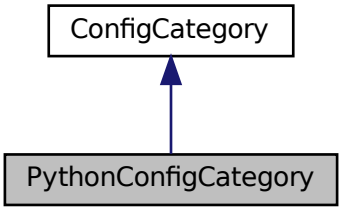
A wrapper class for a ConfigCategory to convert to and from Python objects.

```
#include <pythonconfigcategory.h>
```

Inheritance diagram for PythonConfigCategory:




Collaboration diagram for PythonConfigCategory:




### Public Member Functions

- PythonConfigCategory (PyObject *pyConfig)

  *Construct a PythonConfigCategory from a DICT object returned by Python code.*
- PyObject * toPython ()

  *Convert a ConfigCategory, into a PyObject structure that can be passed to embedded Python code.*

### Additional Inherited Members

### 4.134.1 Detailed Description

A wrapper class for a ConfigCategory to convert to and from Python objects.

### 4.134.2 Constructor & Destructor Documentation

#### 4.134.2.1 PythonConfigCategory()

```
PythonConfigCategory::PythonConfigCategory (
            PyObject * config )
```

Construct a PythonConfigCategory from a DICT object returned by Python code.

The PythonConfigCategory acts as a wrapper on the ConfigCategory class to convert to and from configuration categories in C and Python.

**Parameters**

| | |
|---|---|
| *pyConfig* | The Python DICT |

### 4.134.3 Member Function Documentation

#### 4.134.3.1 toPython()

```
PyObject * PythonConfigCategory::toPython ( )
```

Convert a ConfigCategory, into a PyObject structure that can be passed to embedded Python code.

**Returns**

PyObject∗ The Python representation of the configuration category as a DICT

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

- C/common/include/pythonconfigcategory.h
- C/common/pythonconfigcategory.cpp

## 4.135 PythonModule Class Reference

This class represents the loaded Python module with interpreter initialisation flag.

```
#include <python_plugin_common_interface.h>
```

### Public Member Functions

- **PythonModule** (PyObject ∗module, bool init, string name, string type, PyThreadState ∗state)
- void **setCategoryName** (string category)
- string & **getCategoryName** ()

### Public Attributes

- PyObject ∗ **m_module**
- bool **m_init**
- string **m_name**
- string **m_type**
- PyThreadState ∗ **m_tState**
- string **m_categoryName**

### 4.135.1 Detailed Description

This class represents the loaded Python module with interpreter initialisation flag.

That flag is checked in PluginInterfaceCleanup before removing Python interpreter.

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

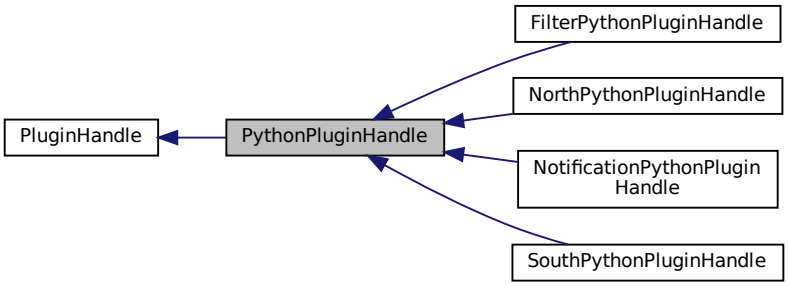
- C/services/common-plugin-interfaces/python/include/python_plugin_common_interface.h

## 4.136 PythonPluginHandle Class Reference

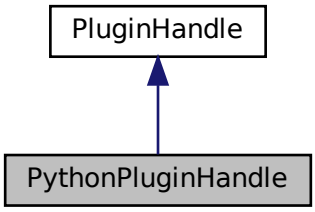
The PythonPluginHandle class is the base class used to represent an interface to a plugin that is available as a python script.

```
#include <python_plugin_handle.h>
```

Inheritance diagram for PythonPluginHandle:



Collaboration diagram for PythonPluginHandle:




## Public Member Functions

- **PythonPluginHandle** (const char ∗name, const char ∗path)
- ∼PythonPluginHandle ()

  *Base destructor.*
- void ∗ ResolveSymbol (const char ∗sym)

  *Gets function pointer from loaded python interface library that can be invoked to call 'sym' function in python plugin.*
- void ∗ GetInfo ()

  *Returns function pointer that can be invoked to call 'plugin_info' function in python plugin.*
- void ∗ **getHandle** ()

## Public Attributes

- void ∗ **m_hndl**
- std::string **m_interfaceObjName**
- std::string **m_name**
- std::string **m_type**

### 4.136.1 Detailed Description

The PythonPluginHandle class is the base class used to represent an interface to a plugin that is available as a python script.

### 4.136.2 Constructor & Destructor Documentation

#### 4.136.2.1 ∼PythonPluginHandle()

```
PythonPluginHandle::~PythonPluginHandle ( )  [inline]
```

Base destructor.

- Call cleanup on python plugin interface
- Close python plugin interface library handle

### 4.136.3 Member Function Documentation

#### 4.136.3.1 ResolveSymbol()

```
void* PythonPluginHandle::ResolveSymbol (
            const char * sym )  [inline], [virtual]
```

Gets function pointer from loaded python interface library that can be invoked to call 'sym' function in python plugin.

**Parameters**

| | |
|---|---|
| *sym* | The symbol to fetch |

Implements PluginHandle.

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

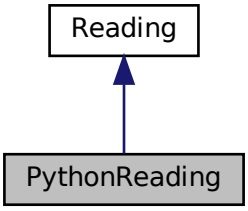
- C/services/common/include/python_plugin_handle.h

## 4.137 PythonReading Class Reference

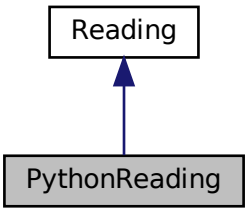
A wrapper class for a Reading to convert to and from Python objects.

```
#include <pythonreading.h>
```

Inheritance diagram for PythonReading:



Collaboration diagram for PythonReading:




## Public Member Functions

- PythonReading (PyObject *pyReading)

  *Construct a PythonReading from a DICT object returned by Python code.*
- PyObject * toPython (bool changeKeys=false, bool bytesString=false)

  *Convert a PythonReading, which is just a Reading, into a PyObject structure that can be passed to embedded Python code.*

## Static Public Member Functions

- static std::string errorMessage ()

  *Retrieve the error message last raised in Python.*
- static bool isArray (PyObject *)

  *Return true of the Python object is an array.*

## Static Public Attributes

- static bool **doneNumPyImport** = false

## Additional Inherited Members

### 4.137.1 Detailed Description

A wrapper class for a Reading to convert to and from Python objects.

### 4.137.2 Constructor & Destructor Documentation

#### 4.137.2.1 PythonReading()

```
PythonReading::PythonReading (
            PyObject * pyReading )
```

Construct a PythonReading from a DICT object returned by Python code.

The PythonReading acts as a wrapper on the Reading class to convert to and from Readings in C and Python.

**Parameters**

| | |
|---|---|
| *pyReading* | The Python DICT |

Set id, uuid, ts and user_ts of the original data

## 4.137.3 Member Function Documentation

### 4.137.3.1 errorMessage()

```
string PythonReading::errorMessage ( ) [static]
```

Retrieve the error message last raised in Python.

**Returns**

string The Python error message

### 4.137.3.2 isArray()

```
bool PythonReading::isArray (
            PyObject * obj ) [static]
```

Return true of the Python object is an array.

This is mostly for testing and overcomes an issue with including the numpy header files multiple times.

**Parameters**

| | |
|---|---|
| *obj* | The Pythin object to test |

**Returns**

true if the Python object is a numpy array

### 4.137.3.3 toPython()

```
PyObject * PythonReading::toPython (
            bool changeKeys = false,
            bool useBytesString = false )
```

Convert a PythonReading, which is just a Reading, into a PyObject structure that can be passed to embedded Python code.

**Parameters**

| | |
|---|---|
| *changeKeys* | Set DICT keys as reading/asset_code if true or readings/asset if false |
| *useBytesString* | Whether to use DICT keys as BytesString and string values as BytesString |

**Returns**

PyObject∗ The Python representation of the readings as a DICT

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

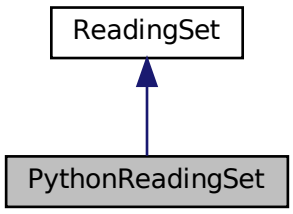
- C/common/include/pythonreading.h
- C/common/pythonreading.cpp

## 4.138 PythonReadingSet Class Reference

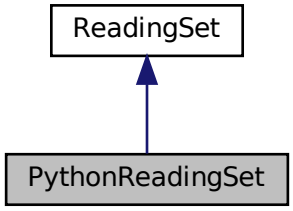
A wrapper class for the ReadingSet class that allows conversion to and from Python objects.

```
#include <pythonreadingset.h>
```

Inheritance diagram for PythonReadingSet:




Collaboration diagram for PythonReadingSet:



## Public Member Functions

- PythonReadingSet (PyObject *pySet)

  *Construct PythonReadingSet from a python list object that contains a list of readings.*
- PyObject * toPython (bool changeKeys=false)

  *Convert the ReadingSet to a Python List.*

## Additional Inherited Members

## 4.138.1 Detailed Description

A wrapper class for the ReadingSet class that allows conversion to and from Python objects.

## 4.138.2 Constructor & Destructor Documentation

### 4.138.2.1 PythonReadingSet()

```
PythonReadingSet::PythonReadingSet (
            PyObject * set )
```

Construct PythonReadingSet from a python list object that contains a list of readings.

**Parameters**

| | |
|---|---|
| *set* | A Python object pointer that contians a list of readings |

## 4.138.3 Member Function Documentation

### 4.138.3.1 toPython()

```
PyObject * PythonReadingSet::toPython (
            bool changeKeys = false )
```

Convert the ReadingSet to a Python List.

**Returns**

A Python object that contains the set of readings as a Python list

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

- C/common/include/pythonreadingset.h
- C/common/pythonreadingset.cpp

## 4.139 PythonRuntime Class Reference

### Public Member Functions

- void execute (const std::string &python)

  *Execute simple Python script passed as a string.*
- PyObject * **call** (const std::string &name, const std::string &fmt,...)
- PyObject * **call** (PyObject *module, const std::string &name, const std::string &fmt,...)
- PyObject * importModule (const std::string &name)

  *Import a Python module.*

### Static Public Member Functions

- static PythonRuntime * getPythonRuntime ()

  *Get PythonRuntime singleton instance for the process.*
- static bool **initialised** ()
- static void shutdown ()

  *Shutdown an instance of a Python runtime if one has been started.*

### 4.139.1 Member Function Documentation

#### 4.139.1.1 execute()

```
void PythonRuntime::execute (
            const std::string & python )
```

Execute simple Python script passed as a string.

**Parameters**

| | |
|---|---|
| *python* | The Python code to run |

#### 4.139.1.2 getPythonRuntime()

```
PythonRuntime * PythonRuntime::getPythonRuntime ( ) [static]
```

Get PythonRuntime singleton instance for the process.

**Returns**

Singleton PythonRuntime instance

#### 4.139.1.3 importModule()

```
PyObject * PythonRuntime::importModule (
            const std::string & name )
```

Import a Python module.

**Parameters**

| | |
|---|---|
| *name* | The name of the module to import |

**Returns**

PyObject∗ The Python module

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

- C/common/include/pyruntime.h
- C/common/pyexception.cpp
- C/common/pyruntime.cpp

## 4.140 Query Class Reference

Storage layer query container.

```
#include <query.h>
```

### Public Member Functions

- Query (Where ∗where)

  *Construct a query with a simple where clause.*
- Query (Aggregate ∗aggreate, Where ∗where)

  *Construct a query with a where clause and aggregate response.*
- Query (Timebucket ∗timebucket, Where ∗where)

  *Construct a timebucket query with a simple where clause.*
- Query (Timebucket ∗timebucket, Where ∗where, unsigned int limit)

  *Construct a timebucket query with a simple where clause and a limit on the rows to return.*
- Query (Returns ∗returns)

  *Construct a simple query to return a certain column from a table.*
- Query (std::vector< Returns ∗ > returns)

  *Construct a simple query to return certain columns from a table.*
- Query (std::vector< Returns ∗ > returns, Where ∗where)

  *Construct a query with a fixed set of returned values and a simple where clause.*
- Query (std::vector< Returns ∗ > returns, Where ∗where, unsigned int limit)

  *Construct a query with a fixed set of returned values and a simple where clause and return a limited set of rows.*
- ∼Query ()

  *Destructor for a query object.*
- void aggregate (Aggregate ∗aggegate)

*Add a aggregate operation to an existing query object.*

- void group (const std::string &column)

  *Add a group operation to a query.*
- void sort (Sort *sort)

  *Add a sort operation to an existing query.*
- void limit (unsigned int limit)

  *Limit the numebr of rows returned by the query.*
- void timebucket (Timebucket *)

  *Add a timebucket operation to an existing query.*
- void returns (Returns *)

  *Limit the query to return just a single column.*
- void returns (std::vector< Returns * >)

  *Limit the columns returned by the query.*
- void distinct ()

  *Add a distinct value modifier to the query.*
- void join (Join *join)

  *Add a join clause to a query.*
- const std::string toJSON () const

  *Return the JSON payload for a where clause.*

## 4.140.1 Detailed Description

Storage layer query container.

## 4.140.2 Constructor & Destructor Documentation

### 4.140.2.1 Query() [1/8]

```
Query::Query (
            Where * where )
```

Construct a query with a simple where clause.

**Parameters**

| | |
|---|---|
| *where* | A pointer to the where condition |

### 4.140.2.2 Query() [2/8]

```
Query::Query (
            Aggregate * aggregate,
            Where * where )
```

Construct a query with a where clause and aggregate response.

**Parameters**

| | |
|---|---|
| *aggregate* | A ppointer to the aggregate operation to perform |
| *where* | A pointer to the where condition |

#### 4.140.2.3 Query() [3/8]

```
Query::Query (
            Timebucket * timebucket,
            Where * where )
```

Construct a timebucket query with a simple where clause.

**Parameters**

| | |
|---|---|
| *timebuck* | A pointer to the timebucket definition |
| *where* | A pointer to the where condition |

#### 4.140.2.4 Query() [4/8]

```
Query::Query (
            Timebucket * timebucket,
            Where * where,
            unsigned int limit )
```

Construct a timebucket query with a simple where clause and a limit on the rows to return.

**Parameters**

| | |
|---|---|
| *timebuck* | A pointer to the timebucket definition |
| *where* | A pointer to the where condition |
| *limit* | The number of rows to return |

#### 4.140.2.5 Query() [5/8]

```
Query::Query (
            Returns * returns )
```

Construct a simple query to return a certain column from a table.

**Parameters**

| | |
|---|---|
| *returns* | The rows to return |

#### 4.140.2.6 Query() [6/8]

```
Query::Query (
            std::vector< Returns * > returns )
```

Construct a simple query to return certain columns from a table.

**Parameters**

| | |
|---|---|
| *returns* | The rows to return |

#### 4.140.2.7 Query() [7/8]

```
Query::Query (
            std::vector< Returns * > returns,
            Where * where )
```

Construct a query with a fixed set of returned values and a simple where clause.

@params returns The set of rows to return @params where The where clause

#### 4.140.2.8 Query() [8/8]

```
Query::Query (
            std::vector< Returns * > returns,
            Where * where,
            unsigned int limit )
```

Construct a query with a fixed set of returned values and a simple where clause and return a limited set of rows.

@params returns The set of rows to return @params where The where clause

**Parameters**

| | |
|---|---|
| *limit* | The numebr of rows to return |

### 4.140.3 Member Function Documentation

#### 4.140.3.1 aggregate()

```
void Query::aggregate (
            Aggregate * aggregate )
```

Add a aggregate operation to an existing query object.

**Parameters**

| | |
|---|---|
| *aggregate* | The aggregate operation to add |

#### 4.140.3.2 group()

```
void Query::group (
            const std::string & column )
```

Add a group operation to a query.

**Parameters**

| | |
|---|---|
| *column* | The column to group by |

#### 4.140.3.3 join()

```
void Query::join (
            Join * join )
```

Add a join clause to a query.

**Parameters**

| | |
|---|---|
| *join* | A pointer to a Join onject |

#### 4.140.3.4 limit()

```
void Query::limit (
            unsigned int limit )
```

Limit the numebr of rows returned by the query.

**Parameters**

| | |
|---|---|
| *limit* | The number of rows to limit the return to |

#### 4.140.3.5 returns() [1/2]

```
void Query::returns (
            Returns * returns )
```

Limit the query to return just a single column.

**Parameters**

| | |
|---|---|
| *returns* | The column to return |

#### 4.140.3.6 returns() [2/2]

```
void Query::returns (
            std::vector< Returns * > returns )
```

Limit the columns returned by the query.

**Parameters**

| | |
|---|---|
| *returns* | The columns to return |

#### 4.140.3.7 sort()

```
void Query::sort (
            Sort * sort )
```

Add a sort operation to an existing query.

**Parameters**

| | |
|---|---|
| *sort* | The sort operation to add |

#### 4.140.3.8 timebucket()

```
void Query::timebucket (
            Timebucket * timebucket )
```

Add a timebucket operation to an existing query.

**Parameters**

| | |
|---|---|
| *timebucket* | The timebucket operation to add to the query |

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

- C/common/include/query.h
- C/common/query.cpp

## 4.141 QueryProfile Class Reference

### Public Member Functions

- **QueryProfile** (int samples)
- void **insert** (ProfileItem *item)
- **QueryProfile** (int samples)
- void **insert** (ProfileItem *item)

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

- C/plugins/storage/sqlite/include/profile.h

## 4.142 RDSAcknowledge Struct Reference

### Public Attributes

- uint32_t **magic**
- uint32_t **block**

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

- C/common/include/reading_stream.h

## 4.143 RDSBlockHeader Struct Reference

### Public Attributes

- uint32_t **magic**
- uint32_t **blockNumber**
- uint32_t **count**

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

- C/common/include/reading_stream.h

## 4.144 RDSConnectHeader Struct Reference

### Public Attributes

- uint32_t **magic**
- uint32_t **token**

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

- C/common/include/reading_stream.h

## 4.145 RDSReadingHeader Struct Reference

### Public Attributes

- uint32_t **magic**
- uint32_t **readingNo**
- uint32_t **assetLength**
- uint32_t **payloadLength**

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

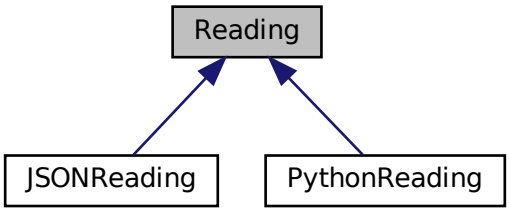
- C/common/include/reading_stream.h

## 4.146 Reading Class Reference

An asset reading represented as a class.

```
#include <reading.h>
```

Inheritance diagram for Reading:




### Public Types

- enum **dateTimeFormat** { **FMT_DEFAULT**, **FMT_STANDARD**, **FMT_ISO8601**, **FMT_ISO8601MS** }
- typedef enum Reading::dateTimeFormat **readingTimeFormat**

### Public Member Functions

- **Reading** (const std::string &asset, Datapoint *value)
- **Reading** (const std::string &asset, std::vector< Datapoint * > values)
- **Reading** (const std::string &asset, std::vector< Datapoint * > values, const std::string &ts)
- **Reading** (const std::string &asset, const std::string &datapoints)
- Reading (const Reading &orig)

  *Reading copy constructor.*
- virtual ~Reading ()

  *Destructor for Reading class.*
- void addDatapoint (Datapoint *value)

  *Add another data point to an asset reading.*
- Datapoint * removeDatapoint (const std::string &name)

  *Remove a datapoint from the reading.*
- Datapoint * getDatapoint (const std::string &name) const

  *Return a specific data point by name.*
- std::string toJSON (bool minimal=false) const

  *Return the asset reading as a JSON structure encoded in a C++ string.*
- std::string getDatapointsJSON () const

  *Return the asset reading as a JSON structure encoded in a C++ string.*
- const std::string & **getAssetName** () const
- void **setAssetName** (std::string assetName)
- unsigned int **getDatapointCount** ()
- void removeAllDatapoints ()

*Remove all data points for Reading class.*

- const std::vector< Datapoint * > **getReadingData** () const
- std::vector< Datapoint * > & **getReadingData** ()
- bool **hasId** () const
- unsigned long **getId** () const
- unsigned long **getTimestamp** () const
- unsigned long **getUserTimestamp** () const
- void **setId** (unsigned long id)
- void **setTimestamp** (unsigned long ts)
- void **setTimestamp** (struct timeval tm)
- void **setTimestamp** (const std::string &timestamp)
- void **getTimestamp** (struct timeval *tm)
- void **setUserTimestamp** (unsigned long uTs)
- void **setUserTimestamp** (struct timeval tm)
- void **setUserTimestamp** (const std::string &timestamp)
- void **getUserTimestamp** (struct timeval *tm)
- void getFormattedDateTimeStr (const time_t *tv_sec, char *date_time, readingTimeFormat dateFormat) const

  *Convert time since epoch to a formatted m_timestamp DataTime in UTC and use a cache to speed it up.*
- const std::string getAssetDateTime (readingTimeFormat datetimeFmt=FMT_DEFAULT, bool addMs=true) const

  *Return a formatted m_timestamp DataTime in UTC.*
- const std::string getAssetDateUserTime (readingTimeFormat datetimeFmt=FMT_DEFAULT, bool add↩Ms=true) const

  *Return a formatted m_userTimestamp DataTime in UTC.*
- std::string substitute (const std::string &str)

  *Substitute values from this reading into the string.*

## Protected Member Functions

- Reading & **operator=** (Reading const &)
- void stringToTimestamp (const std::string &timestamp, struct timeval *ts)

  *Convert a string timestamp, with milliseconds to a struct timeval.*
- const std::string escape (const std::string &str) const

  *Escape quotes etc to allow the string to be a property value within a JSON document.*
- std::vector< Datapoint * > * JSONtoDatapoints (const rapidjson::Value &json)

  *Convert a JSON Value object to a set of data points.*

## Protected Attributes

- unsigned long **m_id**
- bool **m_has_id**
- std::string **m_asset**
- struct timeval **m_timestamp**
- struct timeval **m_userTimestamp**
- std::vector< Datapoint * > **m_values**

## Static Protected Attributes

- static std::vector< std::string > **m_dateTypes**

### 4.146.1 Detailed Description

An asset reading represented as a class.

Each asset reading may have multiple datapoints to represent the multiple values that maybe held within a complex asset.

NB The timestamp data held for both the system timestamp and the user timestamp are always held internally as UTC times

### 4.146.2 Member Function Documentation

#### 4.146.2.1 escape()

```
const string Reading::escape (
            const std::string & str ) const  [protected]
```

Escape quotes etc to allow the string to be a property value within a JSON document.

**Parameters**

| | |
|---|---|
| *str* | The string to escape |

**Returns**

The escaped string

#### 4.146.2.2 getAssetDateTime()

```
const string Reading::getAssetDateTime (
            readingTimeFormat dateFormat = FMT_DEFAULT,
            bool addMS = true ) const
```

Return a formatted m_timestamp DataTime in UTC.

**Parameters**

| | |
|---|---|
| *dateFormat* | Format: FMT_DEFAULT or FMT_STANDARD |

**Returns**

The formatted datetime string

#### 4.146.2.3 getAssetDateUserTime()

```
const string Reading::getAssetDateUserTime (
            readingTimeFormat dateFormat = FMT_DEFAULT,
            bool addMS = true ) const
```

Return a formatted m_userTimestamp DataTime in UTC.

**Parameters**

| | |
|---|---|
| *dateFormat* | Format: FMT_DEFAULT or FMT_STANDARD |

**Returns**

The formatted datetime string

#### 4.146.2.4 getDatapoint()

```
Datapoint * Reading::getDatapoint (
            const std::string & name ) const
```

Return a specific data point by name.

**Parameters**

| | |
|---|---|
| *name* | The name of the datapoint to return |

**Returns**

Pointer a the named datapoint or NULL if it does not exist

#### 4.146.2.5 getFormattedDateTimeStr()

```
void Reading::getFormattedDateTimeStr (
            const time_t * tv_sec,
            char * date_time,
            readingTimeFormat dateFormat ) const
```

Convert time since epoch to a formatted m_timestamp DataTime in UTC and use a cache to speed it up.

**Parameters**

| | |
|---|---|
| *tv_sec* | Seconds since epoch |
| *date_time* | Buffer in which to return the formatted timestamp |
| *dateFormat* | Format: FMT_DEFAULT or FMT_STANDARD |

Build date_time with format YYYY-MM-DD HH24:MM:SS.MS+00:00 this is same as Python3: datetime.datetime.↩ now(tz=datetime.timezone.utc)

#### 4.146.2.6 JSONtoDatapoints()

```
vector< Datapoint * > * Reading::JSONtoDatapoints (
            const rapidjson::Value & json )  [protected]
```

Convert a JSON Value object to a set of data points.

**Parameters**

| | |
|---|---|
| *json* | The json object to convert |

#### 4.146.2.7 removeDatapoint()

```
Datapoint * Reading::removeDatapoint (
            const std::string & name )
```

Remove a datapoint from the reading.

**Parameters**

| | |
|---|---|
| *name* | Name of the datapoint to remove |

**Returns**

Pointer to the datapoint removed or NULL if it was not found

#### 4.146.2.8 stringToTimestamp()

```
void Reading::stringToTimestamp (
            const std::string & timestamp,
            struct timeval * ts )  [protected]
```

Convert a string timestamp, with milliseconds to a struct timeval.

Timezone handling The timezone in the string is extracted to get UTC values. Times within a reading are always stored as UTC

**Parameters**

| | |
|---|---|
| *timestamp* | String timestamp |
| *ts* | Struct timeval to populate |

#### 4.146.2.9 substitute()

```
string Reading::substitute (
            const std::string & str )
```

Substitute values from this reading into the string.

Macros are of the form $datapointname$, $ASSET$ or $datapointname|default$

**Parameters**

| | |
|---|---|
| *str* | The string to substitute into |

**Returns**

string The substituted string

### 4.146.3 Member Data Documentation

#### 4.146.3.1 m_dateTypes

```
std::vector< std::string > Reading::m_dateTypes  [static], [protected]
```

**Initial value:**

```
= {
    DEFAULT_DATE_TIME_FORMAT,
    COMBINED_DATE_STANDARD_FORMAT,
    ISO8601_DATE_TIME_FORMAT,
    ISO8601_DATE_TIME_FORMAT
}
```

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

- C/common/include/reading.h
- C/common/reading.cpp

## 4.147 ReadingCircularBuffer Class Reference

A circular buffer of readings.

```
#include <reading_circularbuffer.h>
```

## Public Member Functions

- ReadingCircularBuffer (unsigned int size)

  *Create a circular buffer of readings.*
- ~ReadingCircularBuffer ()

  *Destructor for the circular buffer.*
- void insert (Reading *)

  *Insert a single reading into the shared buffer.*
- void insert (const std::vector< Reading * > &readings)

  *Insert a list of readings into the circular buffer.*
- void insert (const std::vector< Reading * > *readings)

  *Insert a list of readings into the circular buffer.*
- int extract (std::vector< std::shared_ptr< Reading >> &vec)

  *Return the buffered data into a supplied vector.*

### 4.147.1 Detailed Description

A circular buffer of readings.

The buffer size is set in the constructor, when it fills the oldest reading will be overwritten by new readings being appended.

The user can extract the current state at any point in historic order.

### 4.147.2 Constructor & Destructor Documentation

#### 4.147.2.1 ReadingCircularBuffer()

```
ReadingCircularBuffer::ReadingCircularBuffer (
            unsigned int size )
```

Create a circular buffer of readings.

**Parameters**

| | |
|---|---|
| *size* | The number of items to retain in the circular buffer |

### 4.147.3 Member Function Documentation

#### 4.147.3.1 extract()

```
int ReadingCircularBuffer::extract (
            std::vector< std::shared_ptr< Reading >> & vec )
```

Return the buffered data into a supplied vector.

**Parameters**

| | |
|---|---|
| *vec* | The vector to populate with the shared pointers |

**Returns**

int The number of readings placed in the vector

#### 4.147.3.2 insert() [1/3]

```
void ReadingCircularBuffer::insert (
            const std::vector< Reading * > & readings )
```

Insert a list of readings into the circular buffer.

**Parameters**

| | |
|---|---|
| *readings* | The set of readings to ingest |

#### 4.147.3.3 insert() [2/3]

```
void ReadingCircularBuffer::insert (
            const std::vector< Reading * > * readings )
```

Insert a list of readings into the circular buffer.

**Parameters**

| | |
|---|---|
| *readings* | The set of readings to ingest |

#### 4.147.3.4 insert() [3/3]

```
void ReadingCircularBuffer::insert (
            Reading * reading )
```

Insert a single reading into the shared buffer.

**Parameters**

| | |
|---|---|
| *reading* | The reading to insert |

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

- C/common/include/reading_circularbuffer.h
- C/common/reading_circularbuffer.cpp

## 4.148 ReadingsCatalogue::ReadingReference Struct Reference

### Public Attributes

- int **dbId**
- int **tableId**

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

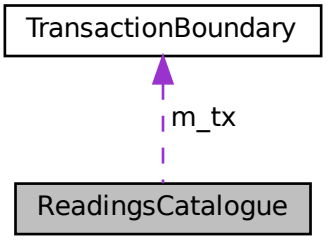
- C/plugins/storage/sqlite/common/include/readings_catalogue.h

## 4.149 ReadingsCatalogue Class Reference

Implements the handling of multiples readings tables stored among multiple SQLite databases.

```
#include <readings_catalogue.h>
```

Collaboration diagram for ReadingsCatalogue:




### Classes

- struct ReadingReference

### Public Types

- typedef struct ReadingsCatalogue::ReadingReference **tyReadingReference**

## Public Member Functions

- void multipleReadingsInit (STORAGE_CONFIGURATION &storageConfig)

  *Setup the multiple readings databases/tables feature.*
- std::string generateDbAlias (int dbId)

  *Generates a SQLite db alias from the database id.*
- std::string generateDbName (int tableId)

  *Generates a SQLIte database name from the database id.*
- std::string generateDbFileName (int dbId)

  *Generates a SQLITE database file name from the database id.*
- std::string generateDbNameFromTableId (int tableId)

  *Identifies SQLIte database name from the given table id.*
- std::string generateReadingsName (int dbId, int tableId)

  *Generates the name of the reading table from the given table id as: Prefix + db Id + reading Id.*
- void getAllDbs (std::vector< int > &dbIdList)

  *Generates a list of all the used databases.*
- void getNewDbs (std::vector< int > &dbIdList)

  *Retrieve the list of newly created databases.*
- int getMaxReadingsId (int dbId)

  *Retrieve the maximum table id for the provided database id.*
- int getReadingsCount ()

  *Returns the number of readings in use.*
- int getReadingPosition (int dbId, int tableId)

  *Returns the position in the array of the specific readings Id considering the database id and the table id.*
- int **getNReadingsAvailable** () const
- long **getIncGlobalId** ()
- long getMinGlobalId (sqlite3 *dbHandle)

  *Calculates the minimum id from the readings tables executing a min(id) on each table.*
- long **getGlobalId** ()
- bool evaluateGlobalId ()

  *Retrieves the global id stored in SQLite and if it is not possible it calculates the value from the readings tables executing a max(id) on each table.*
- bool storeGlobalId ()

  *Stores the global id into SQlite.*
- void preallocateReadingsTables (int dbId)

  *Creates all the required readings tables considering the tables already defined in the database and the number of tables to have on each database.*
- bool loadAssetReadingCatalogue ()

  *Loads the reading catalogue stored in SQLite into an in memory structure.*
- bool loadEmptyAssetReadingCatalogue (bool clean=true)

  *Loads the empty reading table catalogue.*
- bool latestDbUpdate (sqlite3 *dbHandle, int newDbId)

  *Stores on the persistent storage the id of the last created database.*
- int preallocateNewDbsRange (int dbIdStart, int dbIdEnd)

  *Create a set of databases.*
- tyReadingReference getEmptyReadingTableReference (std::string &asset)

  *Get Empty Reading Table.*
- tyReadingReference getReadingReference (Connection *connection, const char *asset_code)

  *Allocates a reading table to the given asset_code.*
- bool attachDbsToAllConnections ()

  *Attaches all the defined SQlite database to all the connections and enable the WAL.*

- std::string sqlConstructMultiDb (std::string &sqlCmdBase, std::vector< std::string > &assetCodes, bool considerExclusion=false)

  *Constructs a sql command from the given one consisting of a set of UNION ALL commands considering all the readings tables in use.*
- std::string sqlConstructOverflow (std::string &sqlCmdBase, std::vector< std::string > &assetCodes, bool considerExclusion=false, bool groupBy=false)

  *Add union all clauses for all the overflow tables based on tempalted SQL that is passed in and a set of assets codes.*
- int purgeAllReadings (sqlite3 *dbHandle, const char *sqlCmdBase, char **errMsg=NULL, unsigned long *rowsAffected=NULL)

  *Delete the content of all the active readings tables using the provided sql command sqlCmdBase.*
- bool connectionAttachAllDbs (sqlite3 *dbHandle)

  *Attaches all the defined SQlite database to all the connections and enable the WAL.*
- bool connectionAttachDbList (sqlite3 *dbHandle, std::vector< int > &dbIdList)

  *Attaches all the defined SQlite database to all the connections and enable the WAL.*
- bool attachDb (sqlite3 *dbHandle, std::string &path, std::string &alias, int dbId)

  *Attach a database to the database connection passed to the call.*
- void detachDb (sqlite3 *dbHandle, std::string &alias)

  *Detach a database from a connection.*
- void setUsedDbId (int dbId)

  *Add the newly create db to the list.*
- int extractReadingsIdFromName (std::string tableName)

  *Extracts the readings id from the table name.*
- int extractDbIdFromName (std::string tableName)

  *Extract the database id from the table name.*
- int SQLExec (sqlite3 *dbHandle, const char *sqlCmd, char **errMsg=NULL)

  *SQLIte wrapper to retry statements when the database is locked.*
- bool createReadingsOverflowTable (sqlite3 *dbHandle, int dbId)

  *Create the overflow reading tables in the given database id.*
- int **getMaxAttached** ()

## Static Public Member Functions

- static ReadingsCatalogue * **getInstance** ()

## Public Attributes

- TransactionBoundary **m_tx**

### 4.149.1 Detailed Description

Implements the handling of multiples readings tables stored among multiple SQLite databases.

The databases are named using the format readings_<dbid>, like for example readings_1.db and each database contains multiples readings named as readings_<dbid>_<id> like readings_1_1, readings_1_2

The table asset_reading_catalogue is used as a catalogue in order map a particular asset_code to a table that holds readings for that asset_code.

readings_1.asset_reading_catalogue:

- table_id INTEGER NOT NULL,
- db_id INTEGER NOT NULL,
- asset_code character varying(50) NOT NULL

The first reading table readings_1_1 is created by the script init_readings.sql executed during the storage init all the other readings tables are created by the code when Fledge starts.

The table configuration_readings created by the script init_readings.sql keeps track of the information:

- global_id – Stores the last global Id used +1, Updated at -1 when Fledge starts, Updated at the proper value when Fledge stops
- db_id_Last – Latest database available
- n_readings_per_db – Number of readings table per database
- n_db_preallocate – Number of databases to allocate in advance

The readings tables are allocated in sequence starting from the readings_1_1 and proceeding with the other tables available in the first database. The tables in the 2nd database (readings_2.db) will be used when all the tables in the first db are allocated.

Implementation notes:

1) Many functions receive the database connection as an input parameter:

- sqlite3 ∗dbHandle

and they will use that connection for the sql operations instead of allocating a new one each time. This approach allows to:

- allocate a connection once using it for all the following operations
- avoid to receive in use a connection having a different configuration (attached databases) as the connections are handled in pool and it is not defined which one will be allocated moreover all the operations are executed in parallel in multi threads

## 4.149.2 Member Function Documentation

### 4.149.2.1 attachDb()

```
bool ReadingsCatalogue::attachDb (
            sqlite3 * dbHandle,
            std::string & path,
            std::string & alias,
            int id )
```

Attach a database to the database connection passed to the call.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |
| *path* | path of the database to attach |
| *alias* | alias to be assigned to the attached database |
| *id* | the database ID |

#### 4.149.2.2 attachDbsToAllConnections()

```
bool ReadingsCatalogue::attachDbsToAllConnections ( )
```

Attaches all the defined SQlite database to all the connections and enable the WAL.

**Returns**

True of success, false on any error

#### 4.149.2.3 connectionAttachAllDbs()

```
bool ReadingsCatalogue::connectionAttachAllDbs (
            sqlite3 * dbHandle )
```

Attaches all the defined SQlite database to all the connections and enable the WAL.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |

**Returns**

True of success, false on any error

#### 4.149.2.4 connectionAttachDbList()

```
bool ReadingsCatalogue::connectionAttachDbList (
            sqlite3 * dbHandle,
            std::vector< int > & dbIdList )
```

Attaches all the defined SQlite database to all the connections and enable the WAL.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |
| *dbIdList* | List of database to attach |

**Returns**

True of success, false on any error

#### 4.149.2.5 createReadingsOverflowTable()

```
bool ReadingsCatalogue::createReadingsOverflowTable (
            sqlite3 * dbHandle,
            int dbId )
```

Create the overflow reading tables in the given database id.

We should only do this once when we upgrade to the version with an overflow table. Although this should ideally be done in the schema update script we can't do this as we can not loop over all the databases in that script.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operation |

#### 4.149.2.6 detachDb()

```
void ReadingsCatalogue::detachDb (
            sqlite3 * dbHandle,
            std::string & alias )
```

Detach a database from a connection.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations* |
| *alias* | alias of the database to detach |

#### 4.149.2.7 evaluateGlobalId()

```
bool ReadingsCatalogue::evaluateGlobalId ( )
```

Retrieves the global id stored in SQLite and if it is not possible it calculates the value from the readings tables executing a max(id) on each table.

Once retrieved or calculated, It updates the value into SQlite to -1 to force a calculation at the next plugin init (Fledge starts) in the case the proper value was not stored as the plugin shutdown (when Fledge is stopped) was not called.

#### 4.149.2.8 extractDbIdFromName()

```
int ReadingsCatalogue::extractDbIdFromName (
            std::string tableName )
```

Extract the database id from the table name.

**Parameters**

| | |
|---|---|
| *tableName* | Table name from which the database id must be extracted |

**Returns**

Extracted database id or -1 on error

#### 4.149.2.9 extractReadingsIdFromName()

```
int ReadingsCatalogue::extractReadingsIdFromName (
            std::string tableName )
```

Extracts the readings id from the table name.

**Parameters**

| | |
|---|---|
| *tableName* | Table name from which the id must be extracted |

**Returns**

Extracted reading id or -1 on error

#### 4.149.2.10 generateDbAlias()

```
string ReadingsCatalogue::generateDbAlias (
            int dbId )
```

Generates a SQLite db alias from the database id.

**Parameters**

| | |
|---|---|
| *db↩ Id* | Database id for which the alias must be generated |

**Returns**

Generated alias

#### 4.149.2.11 generateDbFileName()

```
string ReadingsCatalogue::generateDbFileName (
            int dbId )
```

Generates a SQLITE database file name from the database id.

**Parameters**

| | |
|---|---|
| *db↩ Id* | Database id for which the database file name must be generated |

**Returns**

Generated database file name

#### 4.149.2.12 generateDbName()

```
string ReadingsCatalogue::generateDbName (
            int dbId )
```

Generates a SQLIte database name from the database id.

**Parameters**

| | |
|---|---|
| *db↩ Id* | Database id for which the database name must be generated |

**Returns**

Generated database name

#### 4.149.2.13 generateDbNameFromTableId()

```
string ReadingsCatalogue::generateDbNameFromTableId (
            int tableId )
```

Identifies SQLIte database name from the given table id.

**Parameters**

| | |
|---|---|
| *table↩ Id* | Table id for which the database name must be retrieved |

**Returns**

Retrieved database name for the requested reading id

#### 4.149.2.14 generateReadingsName()

```
string ReadingsCatalogue::generateReadingsName (
            int dbId,
            int tableId )
```

Generates the name of the reading table from the given table id as: Prefix + db Id + reading Id.

If the tableId is 0 then this is a reference to the overflow table

**Parameters**

| | |
|---|---|
| *dbId* | Database id to use for the generation of the table name |
| *table↩ Id* | Table id to use for the generation of the table name |

**Returns**

Generated reading table name

#### 4.149.2.15 getAllDbs()

```
void ReadingsCatalogue::getAllDbs (
            std::vector< int > & dbIdList )
```

Generates a list of all the used databases.

Note this list does not include the first database, readings_1, onl the ohtes that have been added.

**Parameters**

| | |
|---|---|
| *dbIdList* | returned by reference, the list databases in use |

#### 4.149.2.16 getEmptyReadingTableReference()

```
ReadingsCatalogue::tyReadingReference ReadingsCatalogue::getEmptyReadingTableReference (
            std::string & asset )
```

Get Empty Reading Table.

**Parameters**

| | |
|---|---|
| *asset* | emptyAsset, copies value of asset for which empty table is found |

**Returns**

the reading id associated to the provided empty table

#### 4.149.2.17 getMaxReadingsId()

```
int ReadingsCatalogue::getMaxReadingsId (
            int dbId )
```

Retrieve the maximum table id for the provided database id.

**Parameters**

| | |
|---|---|
| *db↩ Id* | Database id for which the maximum reading id must be retrieved |

**Returns**

Maximum readings for the requested database id

#### 4.149.2.18 getMinGlobalId()

```
long ReadingsCatalogue::getMinGlobalId (
            sqlite3 * dbHandle )
```

Calculates the minimum id from the readings tables executing a min(id) on each table.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |

#### 4.149.2.19 getNewDbs()

```
void ReadingsCatalogue::getNewDbs (
            std::vector< int > & dbIdList )
```

Retrieve the list of newly created databases.

**Parameters**

| | |
|---|---|
| *dbIdList* | returned by reference, the list of new databases |

#### 4.149.2.20 getReadingPosition()

```
int ReadingsCatalogue::getReadingPosition (
            int dbId,
            int tableId )
```

Returns the position in the array of the specific readings Id considering the database id and the table id.

**Parameters**

| | |
|---|---|
| *dbId* | Database id for which calculation must be evaluated |
| *tableId* | table id for which calculation must be evaluated |

**Returns**

Position in the array of the specific readings Id

#### 4.149.2.21 getReadingReference()

```
ReadingsCatalogue::tyReadingReference ReadingsCatalogue::getReadingReference (
            Connection * connection,
            const char * asset_code )
```

Allocates a reading table to the given asset_code.

**Parameters**

| | |
|---|---|
| *connection* | Db connection to be used for the operations |
| *asset_code* | for which the referenced readings table should be idenfied |

**Returns**

the reading id associated to the provided asset_code

#### 4.149.2.22 getReadingsCount()

```
int ReadingsCatalogue::getReadingsCount ( )
```

Returns the number of readings in use.

**Returns**

number of readings in use

#### 4.149.2.23 latestDbUpdate()

```
bool ReadingsCatalogue::latestDbUpdate (
            sqlite3 * dbHandle,
            int newDbId )
```

Stores on the persistent storage the id of the last created database.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |
| *newDbId* | Id of the created database |

**Returns**

True of success, false on any error

#### 4.149.2.24 loadAssetReadingCatalogue()

```
bool ReadingsCatalogue::loadAssetReadingCatalogue ( )
```

Loads the reading catalogue stored in SQLite into an in memory structure.

#### 4.149.2.25 loadEmptyAssetReadingCatalogue()

```
bool ReadingsCatalogue::loadEmptyAssetReadingCatalogue (
            bool clean = true )
```

Loads the empty reading table catalogue.

#### 4.149.2.26 multipleReadingsInit()

```
void ReadingsCatalogue::multipleReadingsInit (
            STORAGE_CONFIGURATION & storageConfig )
```

Setup the multiple readings databases/tables feature.

**Parameters**

| | |
|---|---|
| *storageConfig* | Configuration to apply |

#### 4.149.2.27 preallocateNewDbsRange()

```
int ReadingsCatalogue::preallocateNewDbsRange (
            int dbIdStart,
            int dbIdEnd )
```

Create a set of databases.

**Parameters**

| | |
|---|---|
| *dbIdStart* | Range of the database to create |
| *dbIdEnd* | Range of the database to create |

**Returns**

int The number of datbases created

#### 4.149.2.28 preallocateReadingsTables()

```
void ReadingsCatalogue::preallocateReadingsTables (
            int dbId )
```

Creates all the required readings tables considering the tables already defined in the database and the number of tables to have on each database.

**Parameters**

| | |
|---|---|
| *db↩ Id* | Database Id in which the table must be created |

#### 4.149.2.29 purgeAllReadings()

```
int ReadingsCatalogue::purgeAllReadings (
            sqlite3 * dbHandle,
            const char * sqlCmdBase,
            char ** zErrMsg = NULL,
            unsigned long * rowsAffected = NULL )
```

Delete the content of all the active readings tables using the provided sql command sqlCmdBase.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |
| *sqlCmdBase* | Sql command to execute |
| *zErrMsg* | value returned by reference, Error message |
| *rowsAffected* | value returned by reference if != 0, Number of affected rows |

**Returns**

returns SQLITE_OK if all the sql commands are properly executed

#### 4.149.2.30 setUsedDbId()

```
void ReadingsCatalogue::setUsedDbId (
            int dbId )
```

Add the newly create db to the list.

#### 4.149.2.31 sqlConstructMultiDb()

```
string ReadingsCatalogue::sqlConstructMultiDb (
            std::string & sqlCmdBase,
            std::vector< std::string > & assetCodes,
            bool considerExclusion = false )
```

Constructs a sql command from the given one consisting of a set of UNION ALL commands considering all the readings tables in use.

**Parameters**

| | |
|---|---|
| *sqlCmdBase* | Base Sql command |
| *assetCodes* | Asset codes to evaluate for the operation |
| *considerExclusion* | If True the asset code in the excluded list must not be considered |

**Returns**

Full sql command

#### 4.149.2.32 sqlConstructOverflow()

```
string ReadingsCatalogue::sqlConstructOverflow (
            std::string & sqlCmdBase,
            std::vector< std::string > & assetCodes,
            bool considerExclusion = false,
            bool groupBy = false )
```

Add union all clauses for all the overflow tables based on tempalted SQL that is passed in and a set of assets codes.

**Parameters**

| | |
|---|---|
| *sqlCmdBase* | Base Sql command |
| *assetCodes* | Asset codes to evaluate for the operation |
| *considerExclusion* | If True the asset code in the excluded list must not be considered |
| *groupBy* | Include a group by asset_code in each sub query |

**Returns**

Full sql command

#### 4.149.2.33 SQLExec()

```
int ReadingsCatalogue::SQLExec (
            sqlite3 * dbHandle,
            const char * sqlCmd,
            char ** errMsg = NULL )
```

SQLIte wrapper to retry statements when the database is locked.

**Parameters**

| | |
|---|---|
| *dbHandle* | Database connection to use for the operations |
| *sqlCmdsql* | The SQL to execute |
| *errmsg* | Returned by reference, error message |

**Returns**

SQLite constant indicating the outcome of the requested operation, like for example SQLITE_LOCKED, SQ↩LITE_BUSY...

#### 4.149.2.34 storeGlobalId()

```
bool ReadingsCatalogue::storeGlobalId ( )
```

Stores the global id into SQlite.

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

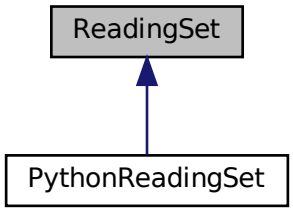
- C/plugins/storage/sqlite/common/include/readings_catalogue.h
- C/plugins/storage/sqlite/common/readings_catalogue.cpp

## 4.150 ReadingSet Class Reference

Reading set class.

```
#include <reading_set.h>
```

Inheritance diagram for ReadingSet:




### Public Member Functions

- ReadingSet ()

  *Construct an empty reading set.*
- ReadingSet (const std::string &json)

  *Construct a reading set from a JSON document returned from the Fledge storage service query or notification.*
- ReadingSet (const std::vector< Reading * > *readings)

  *Construct a reading set from a vector<Reading *> pointer NOTE: readings are copied into m_readings.*
- virtual ~ReadingSet ()

  *Destructor for a result set.*

- unsigned long **getCount** () const
- const Reading ∗ **operator[ ]** (const unsigned int idx)
- const std::vector< Reading ∗ > & **getAllReadings** () const
- std::vector< Reading ∗ > ∗ **getAllReadingsPtr** ()
- std::vector< Reading ∗ > ∗ moveAllReadings ()

  *Remove readings from the vector and return a reference to new vector containing readings∗.*
- Reading ∗ removeReading (unsigned long id)

  *Remove reading from vector based on index and return its pointer.*
- unsigned long **getLastId** () const
- unsigned long getReadingId (uint32_t pos)

  *Return the ID of the nth reading in the reading set.*
- void append (ReadingSet ∗)

  *Append the readings in a second reading set to this reading set.*
- void append (ReadingSet &)

  *Append the readings in a second reading set to this reading set.*
- void append (std::vector< Reading ∗ > &)

  *Append a set of readings to this reading set.*
- void merge (std::vector< Reading ∗ > ∗readings)

  *merge the readings in a vector with the set of readings in the reading set.*
- void removeAll ()

  *Remove all readings from the reading set and delete the memory After this call the reading set exists but contains no readings.*
- void clear ()

  *Remove the readings from the vector without deleting them.*
- bool copy (const ReadingSet &src)

  *Deep copy a set of readings to this reading set.*

## Protected Member Functions

- **ReadingSet** (const ReadingSet &)
- ReadingSet & **operator=** (ReadingSet const &)

## Protected Attributes

- unsigned long **m_count**
- std::vector< Reading ∗ > **m_readings**
- unsigned long **m_last_id**

### 4.150.1 Detailed Description

Reading set class.

A specialised container for a set of readings that allows creation from a JSON document.

### 4.150.2 Constructor & Destructor Documentation

#### 4.150.2.1 ReadingSet() [1/2]

```
ReadingSet::ReadingSet (
            const std::string & json )
```

Construct a reading set from a JSON document returned from the Fledge storage service query or notification.

The JSON is parsed using the in-situ RapidJSON parser in order to reduce overhead on what is most likely a large JSON document.

WARNING: Although the string passed in is defiend as const this call is destructive to this string and the conntents of the string should not be used after making this call.

**Parameters**

| | |
|---|---|
| *json* | The JSON document (as string) with readings data |

#### 4.150.2.2 ReadingSet() [2/2]

```
ReadingSet::ReadingSet (
            const std::vector< Reading * > * readings )
```

Construct a reading set from a vector<Reading *> pointer NOTE: readings are copied into m_readings.

**Parameters**

| | |
|---|---|
| *readings* | The vector<Reading *> pointer of readings to be copied into m_readings vector |

### 4.150.3 Member Function Documentation

#### 4.150.3.1 append() [1/3]

```
void ReadingSet::append (
            ReadingSet & readings )
```

Append the readings in a second reading set to this reading set.

The readings are removed from the original reading set

**Parameters**

| | |
|---|---|
| *readings* | A ReadingSet to append to the current ReadingSet |

#### 4.150.3.2 append() [2/3]

```
void ReadingSet::append (
            ReadingSet * readings )
```

Append the readings in a second reading set to this reading set.

The readings are removed from the original reading set

**Parameters**

| | |
|---|---|
| *readings* | A ReadingSet to append to the current ReadingSet |

#### 4.150.3.3 append() [3/3]

```
void ReadingSet::append (
            std::vector< Reading * > & readings )
```

Append a set of readings to this reading set.

The readings are not copied, but rather moved from the vector, with the resulting vector havign the values removed on return.

It is assumed the readings in the vector have been created with the new operator.

**Parameters**

| | |
|---|---|
| *readings* | A vector of Reading pointers to append to the ReadingSet |

#### 4.150.3.4 copy()

```
bool ReadingSet::copy (
            const ReadingSet & src )
```

Deep copy a set of readings to this reading set.

**Parameters**

| | |
|---|---|
| *src* | The reading set to copy |

**Returns**

bool True if the reading set was copied

#### 4.150.3.5 getReadingId()

```
unsigned long ReadingSet::getReadingId (
            uint32_t pos )
```

Return the ID of the nth reading in the reading set.

**Parameters**

| | |
|---|---|
| *pos* | The position of the reading to return the ID for |

#### 4.150.3.6 merge()

```
void ReadingSet::merge (
            std::vector< Reading * > * readings )
```

merge the readings in a vector with the set of readings in the reading set.

The input reading vector must be ordered as per timestamp and cleared at the end of the operation.

**Parameters**

| | |
|---|---|
| *readings* | A vector of Reading pointers to merge with the ReadingSet |

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

- C/common/include/reading_set.h
- C/common/reading_set.cpp

## 4.151 ReadingSetCircularBuffer Class Reference

Reading set circular buffer class.

```
#include <readingset_circularbuffer.h>
```

### Public Member Functions

- ReadingSetCircularBuffer (unsigned long maxBufferSize=10)

  *Construct an empty reading set circular buffer.*
- ~ReadingSetCircularBuffer ()

  *Destructor for a result set.*
- void insert (ReadingSet *)

  *Insert a ReadingSet into circular buffer.*
- void insert (ReadingSet &)

  *Insert a ReadingSet into circular buffer.*
- std::vector< std::shared_ptr< ReadingSet > > extract (bool isExtractSingleElement=true)

  *Fetch the vector of ReadingSet from circular buffer.*

### 4.151.1 Detailed Description

Reading set circular buffer class.

Reading set circular buffer is a data structure to hold ReadingSet passed to a plugin.

### 4.151.2 Constructor & Destructor Documentation

#### 4.151.2.1 ReadingSetCircularBuffer()

```
ReadingSetCircularBuffer::ReadingSetCircularBuffer (
            unsigned long maxBufferSize = 10 )
```

Construct an empty reading set circular buffer.

**Parameters**

| | |
|---|---|
| *maxBufferSize* | Maximum size of the ReadingSet circular buffer. It should be atleast one. |

### 4.151.3 Member Function Documentation

#### 4.151.3.1 extract()

```
std::vector< std::shared_ptr< ReadingSet > > ReadingSetCircularBuffer::extract (
            bool isExtractSingleElement = true )
```

Fetch the vector of ReadingSet from circular buffer.

**Parameters**

| | |
|---|---|
| *isExtractSingleElement* | True to extract single ReadingSet otherwise for extract entire buffer |

**Returns**

Return a vector of shared pointer to ReadingSet

#### 4.151.3.2 insert() [1/2]

```
void ReadingSetCircularBuffer::insert (
            ReadingSet & readings )
```

Insert a ReadingSet into circular buffer.

**Parameters**

| | |
|---|---|
| *readings* | Reference for ReadingSet to be inserted into circular buffer |

#### 4.151.3.3 insert() [2/2]

```
void ReadingSetCircularBuffer::insert (
            ReadingSet * readings )
```

Insert a ReadingSet into circular buffer.

**Parameters**

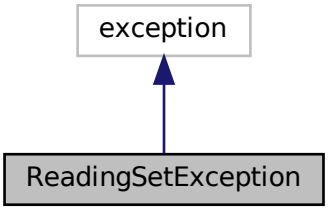
| | |
|---|---|
| *readings* | Pointer for ReadingSet to be inserted into circular buffer |

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

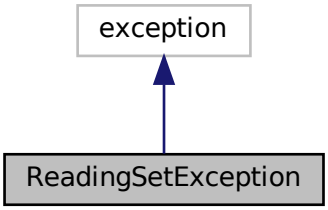
- C/common/include/readingset_circularbuffer.h
- C/common/readingset_circularbuffer.cpp

## 4.152 ReadingSetException Class Reference

Inheritance diagram for ReadingSetException:



Collaboration diagram for ReadingSetException:




### Public Member Functions

- **ReadingSetException** (const char ∗what)
- virtual const char ∗ **what** () const throw ()

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

- C/common/include/reading_set.h

## 4.153 ReadingStream Struct Reference

### Public Attributes

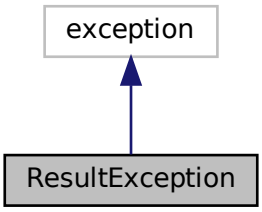
- uint32_t **assetCodeLength**
- uint32_t **payloadLength**
- struct timeval **userTs**
- char **assetCode** [1]

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

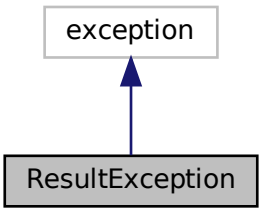
- C/common/include/reading_stream.h

## 4.154 ResultException Class Reference

Inheritance diagram for ResultException:




Collaboration diagram for ResultException:




### Public Member Functions

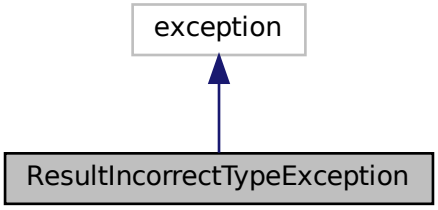
- **ResultException** (const char ∗what)
- virtual const char ∗ **what** () const throw ()

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

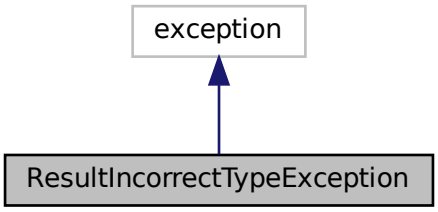
- C/common/include/resultset.h

## 4.155 ResultIncorrectTypeException Class Reference

Inheritance diagram for ResultIncorrectTypeException:




Collaboration diagram for ResultIncorrectTypeException:




### Public Member Functions

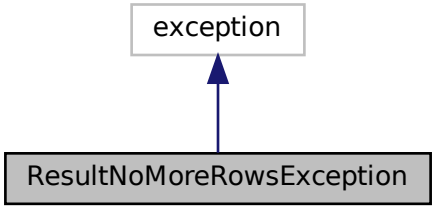
- virtual const char ∗ **what** () const throw ()

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

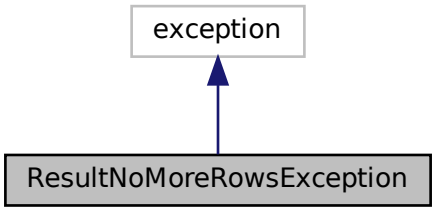
- C/common/include/resultset.h

## 4.156 ResultNoMoreRowsException Class Reference

Inheritance diagram for ResultNoMoreRowsException:




Collaboration diagram for ResultNoMoreRowsException:




### Public Member Functions

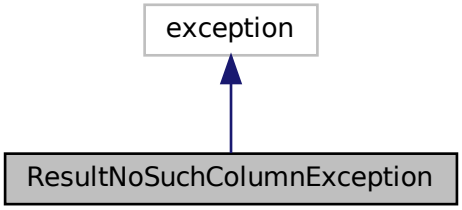
- virtual const char ∗ **what** () const throw ()

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

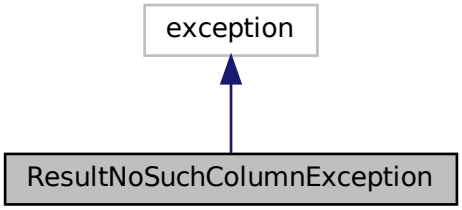
- C/common/include/resultset.h

## 4.157 ResultNoSuchColumnException Class Reference

Inheritance diagram for ResultNoSuchColumnException:




Collaboration diagram for ResultNoSuchColumnException:




### Public Member Functions

- virtual const char ∗ **what** () const throw ()

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

- C/common/include/resultset.h

## 4.158 ResultSet Class Reference

Result set.

```
#include <resultset.h>
```

## Classes

- class ColumnValue
- class Row

## Public Types

- typedef std::vector< Row * >::iterator **RowIterator**

## Public Member Functions

- ResultSet (const std::string &json)

  *Construct a result set from a JSON document returned from the Fledge storage service.*
- ~ResultSet ()

  *Destructor for a result set.*
- unsigned int **rowCount** () const
- unsigned int **columnCount** () const
- const std::string & columnName (unsigned int column) const

  *Return the name of a specific column.*
- ColumnType columnType (unsigned int column) const

  *Return the type of a specific column.*
- ColumnType **columnType** (const std::string &name) const
- RowIterator firstRow ()

  *Fetch an iterator for the rows in a result set.*
- RowIterator nextRow (RowIterator it)

  *Given an iterator over the rows in a result set move to the next row in the result set.*
- bool isLastRow (RowIterator it) const

  *Given an iterator over the rows in a result set return if there this is the last row in the result set.*
- bool hasNextRow (RowIterator it) const

  *Given an iterator over the rows in a result set return if there are any more rows in the result set.*
- unsigned int findColumn (const std::string &name) const

  *Find the named column in the result set and return the column index.*
- const Row * **operator[]** (unsigned long rowNo)

### 4.158.1 Detailed Description

Result set.

### 4.158.2 Constructor & Destructor Documentation

#### 4.158.2.1 ResultSet()

```
ResultSet::ResultSet (
            const std::string & json )
```

Construct a result set from a JSON document returned from the Fledge storage service.

**Parameters**

| | |
|---|---|
| *json* | The JSON document to construct the result set from |

## 4.158.3 Member Function Documentation

### 4.158.3.1 columnName()

```
const string & ResultSet::columnName (
            unsigned int column ) const
```

Return the name of a specific column.

**Parameters**

| | |
|---|---|
| *column* | - the column number of the column to return. Columns are numbered from 0 |

**Returns**

string& The name of the column

**Exceptions**

| | |
|---|---|
| *ResultNoSuchColumnException* | The specified column does not exist in the result set |

### 4.158.3.2 columnType()

```
ColumnType ResultSet::columnType (
            unsigned int column ) const
```

Return the type of a specific column.

**Parameters**

| | |
|---|---|
| *column* | - the column number of the column to return. Columns are numbered from 0 |

**Returns**

ColumnType The type of the specified column

**Exceptions**

| | |
|---|---|
| *ResultNoSuchColumnException* | The specified column does not exist in the result set |

#### 4.158.3.3 findColumn()

```
unsigned int ResultSet::findColumn (
            const std::string & name ) const
```

Find the named column in the result set and return the column index.

**Parameters**

| | |
|---|---|
| *name* | The name of the column to return |

**Returns**

unsigned int The index of the named column

**Exceptions**

| | |
|---|---|
| *ResultNoSuchColumnException* | The named column does not exist in the result set |

#### 4.158.3.4 firstRow()

```
ResultSet::RowIterator ResultSet::firstRow ( )
```

Fetch an iterator for the rows in a result set.

The iterator is positioned at the first row in the result set.

#### 4.158.3.5 hasNextRow()

```
bool ResultSet::hasNextRow (
            RowIterator it ) const
```

Given an iterator over the rows in a result set return if there are any more rows in the result set.

**Parameters**

| | |
|---|---|
| *it* | Iterator returned by the firstRow() method |

**Returns**

bool True if there are more rows in the result set

#### 4.158.3.6 isLastRow()

```
bool ResultSet::isLastRow (
            RowIterator it ) const
```

Given an iterator over the rows in a result set return if there this is the last row in the result set.

**Parameters**

| | |
|---|---|
| *it* | Iterator returned by the firstRow() method |

**Returns**

bool True if there are no more rows in the result set

#### 4.158.3.7 nextRow()

```
ResultSet::RowIterator ResultSet::nextRow (
            RowIterator it )
```

Given an iterator over the rows in a result set move to the next row in the result set.

**Parameters**

| | |
|---|---|
| *it* | Iterator returned by the firstRow() method |

**Returns**

RowIterator New value of the iterator

**Exceptions**

| | |
|---|---|
| *ResultNoMoreRowsException* | There are no more rows in the result set |

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

- C/common/include/resultset.h
- C/common/result_set.cpp

## 4.159 Returns Class Reference

Control a returned column.

```
#include <returns.h>
```

### Public Member Functions

- **Returns** (const std::string &column)
- **Returns** (const std::string &column, const std::string &alias)
- **Returns** (const std::string &column, const std::string &alias, const std::string &format)
- void **format** (const std::string format)
- void **timezone** (const std::string timezone)
- std::string **toJSON** ()

### 4.159.1 Detailed Description

Control a returned column.

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

- C/common/include/returns.h

## 4.160 ResultSet::Row Class Reference

### Public Member Functions

- **Row** (ResultSet *resultSet)
- void **append** (ColumnValue *value)
- ColumnType getType (unsigned int column)

  *Return the type of the given column in this row.*
- ColumnType **getType** (const std::string &name)
- ColumnValue * getColumn (unsigned int column) const

  *Return the column value of the given column in this row.*
- ColumnValue * **getColumn** (const std::string &name) const
- ColumnValue * **operator[]** (unsigned long colNo) const

### 4.160.1 Member Function Documentation

#### 4.160.1.1 getColumn()

```
ResultSet::ColumnValue * ResultSet::Row::getColumn (
            unsigned int column ) const
```

Return the column value of the given column in this row.

**Parameters**

| | |
|---|---|
| *column* | The column number in the row, columns are numbered from 0 |

**Returns**

ColumnValue The column value of the specified column

**Exceptions**

| | |
|---|---|
| *ResultNoSuchColumnException* | The specified column does not exist in the row |

#### 4.160.1.2 getType()

```
ColumnType ResultSet::Row::getType (
            unsigned int column )
```

Return the type of the given column in this row.

**Parameters**

| | |
|---|---|
| *column* | The column number in the row, columns are numbered from 0 |

**Returns**

ColumnType The column type of the specified column

**Exceptions**

| | |
|---|---|
| *ResultNoSuchColumnException* | The specified column does not exist in the row |

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

- C/common/include/resultset.h
- C/common/result_set.cpp

## 4.161 Schema Class Reference

Representation of an extension schema.

```
#include <schema.h>
```

## Public Member Functions

- **Schema** (const std::string &name, const std::string &service, int version, const std::string &definition)
- Schema (sqlite3 ∗db, const rapidjson::Document &doc)

  *Constructor for a schema.*
- int **getVersion** ()
- std::string **getService** ()
- bool upgrade (sqlite3 ∗db, const rapidjson::Document &doc, const std::string &definition)

  *Upgrade an existing schema.*
- bool attach (sqlite3 ∗db)

  *Attach the schema to the database handle if not already attached.*

## 4.161.1 Detailed Description

Representation of an extension schema.

Each active schema has an instance of the class that is managed by the SchemaManager class. This is either created when Fledge restarts and reads the schema definition from the database or when a service requests an extension schema to be created.

The class is responsible for the creation and update of the extension schemas. The name, service, version and definition of each schema is written to a control table in the Fledge schema to allow for the versions to be tracked and also to allow the extension schemas to be attached.

## 4.161.2 Constructor & Destructor Documentation

### 4.161.2.1 Schema()

```
Schema::Schema (
            sqlite3 * db,
            const rapidjson::Document & doc )
```

Constructor for a schema.

This is the case when a service has requested a schema that does not already exist. We must create a new schema from scratch.

**Parameters**

| | |
|---|---|
| *db* | SQLite3 database handle |
| *doc* | JSON definition of the schema to create |

## 4.161.3 Member Function Documentation

#### 4.161.3.1 attach()

```
bool Schema::attach (
            sqlite3 * db )
```

Attach the schema to the database handle if not already attached.

**Parameters**

| | |
|---|---|
| *db* | The database handle to attach the schema to |

**Returns**

bool True if the schema was attached

#### 4.161.3.2 upgrade()

```
bool Schema::upgrade (
            sqlite3 * db,
            const rapidjson::Document & doc,
            const std::string & definition )
```

Upgrade an existing schema.

The upgrade process is limited and will only do the following operations; add a new table, drop a table, add a new column to a table, drop a column from a table, add a new index or drop an index.

**Parameters**

| | |
|---|---|
| *db* | The SQLite3 database connection |
| *doc* | The pre-parsed version of the schema definition |
| *definition* | The schema defintion for the new version of the schema as JSON |
| *bool* | True if the upgrade suceeded. |

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

- C/plugins/storage/sqlite/schema/include/schema.h
- C/plugins/storage/sqlite/schema/schema.cpp

## 4.162 SchemaManager Class Reference

The singleton SchemaManager class used to interact with the extension schemas created by various extension services.

```
#include <schema.h>
```

Collaboration diagram for SchemaManager:

## Public Member Functions

- void load (sqlite3 ∗db)

  *Load the existing Schema from the table of supported schemas.*
- bool create (sqlite3 ∗db, const std::string &definition)

  *Create a schema.*
- bool exists (sqlite3 ∗db, const std::string &schema)

  *Check if a named schema exists, loading the schemas if need be.*

## Static Public Member Functions

- static SchemaManager ∗ getInstance ()

  *Fetch the singleton instance of the SchemaManager.*

## Static Public Attributes

- static SchemaManager ∗ **instance** = 0

## 4.162.1 Detailed Description

The singleton SchemaManager class used to interact with the extension schemas created by various extension services.

## 4.162.2 Member Function Documentation

### 4.162.2.1 create()

```
bool SchemaManager::create (
            sqlite3 * db,
            const std::string & definition )
```

Create a schema.

This may create a completely new schema if it does not already exist, update an existing schema if the version is different from the one that already exists or no nothing if the schema exists and the version of the schema is the same.

**Parameters**

| | |
|---|---|
| *db* | A database connection to use for sqlite3 interactions |
| *definition* | The schema definition |

**Returns**

bool Returns true if the schema was created, updated or no action is required

#### 4.162.2.2 exists()

```
bool SchemaManager::exists (
            sqlite3 * db,
            const std::string & schema )
```

Check if a named schema exists, loading the schemas if need be.

As a side effect the schema will be attached to the database connection.

**Parameters**

| | |
|---|---|
| *schema* | The schema to check the existance of |

**Returns**

bool True if the schema exists

#### 4.162.2.3 getInstance()

```
SchemaManager * SchemaManager::getInstance ( )  [static]
```

Fetch the singleton instance of the SchemaManager.

**Returns**

SchemaManager∗ The singleton SchemaManager instance

#### 4.162.2.4 load()

```
void SchemaManager::load (
            sqlite3 * db )
```

Load the existing Schema from the table of supported schemas.

**Parameters**

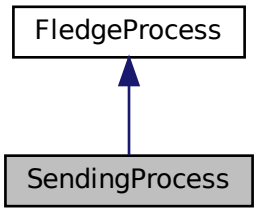
| | |
|---|---|
| *db* | The database connection to use to load the schema information |

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

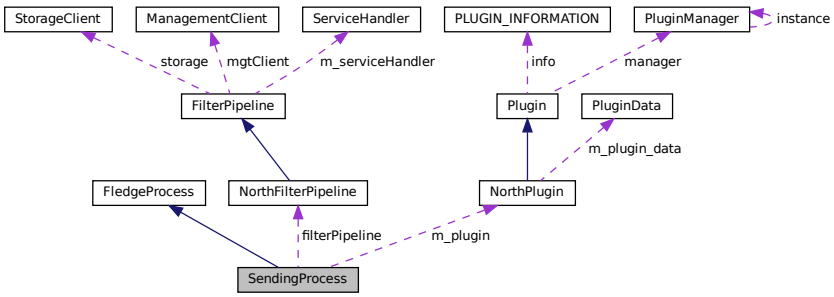
- C/plugins/storage/sqlite/schema/include/schema.h
- C/plugins/storage/sqlite/schema/schema.cpp

## 4.163 SendingProcess Class Reference

Inheritance diagram for SendingProcess:




Collaboration diagram for SendingProcess:




### Public Member Functions

- SendingProcess (int argc, char **argv)
- ~SendingProcess ()

  *SendingProcess class methods.*
- void **run** () const
- void **stop** ()

- int **getStreamId** () const
- bool **isRunning** () const
- void **stopRunning** ()
- void **setLastFetchId** (unsigned long id)
- unsigned long **getLastFetchId** () const
- void **setLastSentId** (unsigned long id)
- unsigned long **getLastSentId** () const
- unsigned long **getSentReadings** () const
- bool **updateSentReadings** (unsigned long num)
- void **resetSentReadings** ()
- void updateDatabaseCounters ()

  *Update database tables statistics and streams setting last_object id in streams.*
- bool getLastSentReadingId ()

  *Get last_object id sent for current stream_id Access foglam.streams table.*
- bool createStream (int)

  *Creates a new stream, it adds a new row into the streams table allocating specific stream id.*
- int createNewStream ()

  *Creates a new stream, it adds a new row into the streams table allocating a new stream id.*
- unsigned int **getDuration** () const
- unsigned int **getSleepTime** () const
- bool **getUpdateDb** () const
- bool **setUpdateDb** (bool val)
- unsigned long **getReadBlockSize** () const
- const std::string & **getDataSourceType** () const
- const std::string & **getPluginName** () const
- void setLoadBufferIndex (unsigned long loadBufferIdx)

  *Set the current buffer load index.*
- unsigned long getLoadBufferIndex () const

  *Get the current buffer load index.*
- const unsigned long ∗ getLoadBufferIndexPtr () const

  *Get the current buffer load index pointer.*
- unsigned long **getMemoryBufferSize** () const
- void createConfigCategories (DefaultConfigCategory configCategory, std::string parent_name, std::string current_name, std::string current_description)

  *Creates config categories and sub categories recursively, along with their parent-child relations.*

## Static Public Member Functions

- static void **setLoadBufferData** (unsigned long index, ReadingSet ∗readings)
- static std::vector< ReadingSet ∗ > ∗ **getDataBuffers** ()
- static void useFilteredData (OUTPUT_HANDLE ∗outHandle, READINGSET ∗readings)

  *Use the current input readings (they have been filtered by all filters)*
- static void passToOnwardFilter (OUTPUT_HANDLE ∗outHandle, READINGSET ∗readings)

  *Pass the current readings set to the next filter in the pipeline.*

## Public Attributes

- std::vector< ReadingSet ∗ > **m_buffer**
- std::thread ∗ **m_thread_load**
- std::thread ∗ **m_thread_send**
- NorthPlugin ∗ **m_plugin**
- std::vector< unsigned long > **m_last_read_id**
- NorthFilterPipeline ∗ **filterPipeline**

## Additional Inherited Members

## 4.163.1 Constructor & Destructor Documentation

### 4.163.1.1 SendingProcess()

```
SendingProcess::SendingProcess (
            int argc,
            char ** argv )
```

Get Configuration from sending process and loaded plugin Create or update configuration via Fledge API

## 4.163.2 Member Function Documentation

### 4.163.2.1 createNewStream()

```
int SendingProcess::createNewStream ( )
```

Creates a new stream, it adds a new row into the streams table allocating a new stream id.

**Returns**

newly created stream, 0 otherwise

### 4.163.2.2 createStream()

```
bool SendingProcess::createStream (
            int streamId )
```

Creates a new stream, it adds a new row into the streams table allocating specific stream id.

**Returns**

true if successful created, false otherwise

#### 4.163.2.3 getLastSentReadingId()

```
bool SendingProcess::getLastSentReadingId ( )
```

Get last_object id sent for current stream_id Access foglam.streams table.

**Returns**

true if last_object is found, false otherwise

#### 4.163.2.4 getLoadBufferIndex()

```
unsigned long SendingProcess::getLoadBufferIndex ( ) const
```

Get the current buffer load index.

**Returns**

The buffer load index the load thread is using

#### 4.163.2.5 getLoadBufferIndexPtr()

```
const unsigned long * SendingProcess::getLoadBufferIndexPtr ( ) const
```

Get the current buffer load index pointer.

NOTE: this routine must be called only to pass the index pointer to the last filter in the pipeline for the readings set.

**Returns**

The pointer to the buffer load index being used by the load thread

#### 4.163.2.6 passToOnwardFilter()

```
void SendingProcess::passToOnwardFilter (
            OUTPUT_HANDLE * outHandle,
            READINGSET * readings )  [static]
```

Pass the current readings set to the next filter in the pipeline.

Note: This routine must be passed to all filters "plugin_init" except the last one

Static method

**Parameters**

| | |
|---|---|
| *outHandle* | Pointer to next filter |
| *readings* | Current readings set |

#### 4.163.2.7 setLoadBufferIndex()

```
void SendingProcess::setLoadBufferIndex (
            unsigned long loadBufferIndex )
```

Set the current buffer load index.

**Parameters**

| | |
|---|---|
| *loadBufferIndex* | The buffer load index the load thread is using |

#### 4.163.2.8 useFilteredData()

```
void SendingProcess::useFilteredData (
            OUTPUT_HANDLE * outHandle,
            READINGSET * readings )  [static]
```

Use the current input readings (they have been filtered by all filters)

Note: This routine must passed to last filter "plugin_init" only

Static method

**Parameters**

| | |
|---|---|
| *outHandle* | Pointer to current buffer index where to add the readings |
| *readings* | Filtered readings to add to buffer[index] |

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

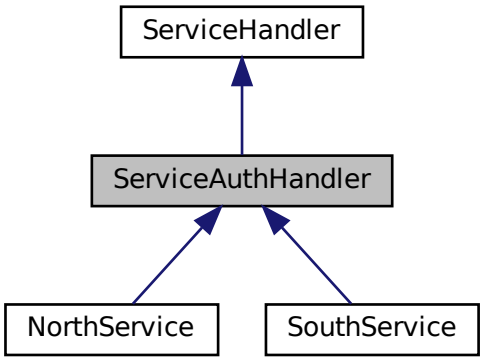
- C/tasks/north/sending_process/include/sending.h
- C/tasks/north/sending_process/sending.cpp

## 4.164 ServiceAuthHandler Class Reference

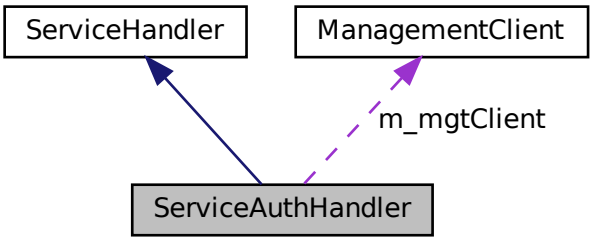
ServiceAuthHandler adds security to the base class ServiceHandler.

```
#include <service_handler.h>
```

Inheritance diagram for ServiceAuthHandler:




Collaboration diagram for ServiceAuthHandler:




## Public Member Functions

- std::string & **getName** ()
- std::string & **getType** ()
- bool createSecurityCategories (ManagementClient *mgtClient, bool dryRun)

  *Create "${service}Security" category with empty content.*
- bool updateSecurityCategory (const std::string &newCategory)

  *Update the class objects from security category content update.*
- void setInitialAuthenticatedCaller ()

  *Set initial value of enabled authentication.*
- void setAuthenticatedCaller (bool enabled)

  *Set enabled authentication value.*
- bool getAuthenticatedCaller ()

  *Return enabled authentication value.*

- bool AuthenticationMiddlewareACL (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::Request > request, const std::string &serviceName, const std::string &serviceType)

  *Authentication Middleware ACL check.*
- bool AuthenticationMiddlewareCommon (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::Request > request, std::string &callerName, std::string &callerType)

  *Authentication Middleware for Dispatcher service.*
- void AuthenticationMiddlewarePUT (std::shared_ptr< HttpServer::Response > response, std::shared_ptr< HttpServer::Request > request, std::function< void(std::shared_ptr< HttpServer::Response >, std::shared_ptr< HttpServer::Request >)> funcPUT)

  *Authentication Middleware for PUT methods.*
- void refreshBearerToken ()

  *Refresh the bearer token of the runnign service This routine is run by a thread started in createSecurityCategories.*
- void **respond** (std::shared_ptr< HttpServer::Response > response, const std::string &payload)
- void **respond** (std::shared_ptr< HttpServer::Response > response, SimpleWeb::StatusCode code, const std::string &payload)
- bool securityChange (const std::string &payload)

  *Request security change action:*

## Static Public Member Functions

- static ManagementClient * **getMgmtClient** ()

## Protected Attributes

- std::string **m_name**
- std::string **m_type**

## Static Protected Attributes

- static ManagementClient * m_mgtClient = NULL

  *Initialise m_mgtClient object to NULL.*

## 4.164.1 Detailed Description

ServiceAuthHandler adds security to the base class ServiceHandler.

## 4.164.2 Member Function Documentation

### 4.164.2.1 AuthenticationMiddlewareACL()

```
bool ServiceAuthHandler::AuthenticationMiddlewareACL (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request,
            const std::string & serviceName,
            const std::string & serviceType )
```

Authentication Middleware ACL check.

serviceName, serviceType and url (request->path) are cheked with verifyURL and verifyService routines

If access is granted return true otherwise error response is sent to the client and return is false

**Parameters**

| | |
|---|---|
| *response* | The HTTP Response to send |
| *request* | The HTTP Request |
| *callerName* | The caller service name to check |
| *callerType* | The caller service type to check |

**Returns**

True on success False otherwise with server reply error

#### 4.164.2.2 AuthenticationMiddlewareCommon()

```
bool ServiceAuthHandler::AuthenticationMiddlewareCommon (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request,
            std::string & callerName,
            std::string & callerType )
```

Authentication Middleware for Dispatcher service.

Routine first check whether the service is configured with authentication

Access bearer token is then verified against FogLAMP core API endpoint token claims 'sub' and 'aud' along with request are passed to verifyURL and verifyService routines

If access is granted then return map with token claims otherwise error response is sent to the client and empty map is returned.

**Parameters**

| | |
|---|---|
| *response* | The HTTP Response to send |
| *request* | The HTTP Request |

**Returns**

True on success False on errors

#### 4.164.2.3 AuthenticationMiddlewarePUT()

```
void ServiceAuthHandler::AuthenticationMiddlewarePUT (
            std::shared_ptr< HttpServer::Response > response,
            std::shared_ptr< HttpServer::Request > request,
            std::function< void(std::shared_ptr< HttpServer::Response >, std::shared_ptr<
HttpServer::Request >)> funcPUT )
```

Authentication Middleware for PUT methods.

Routine first check whether the service is configured with authentication

Access bearer token is then verified against FogLAMP core API endpoint JWT token claims are passed to verifyURL and verifyService routines

If access is granted the input funcPUT funcion is called otherwise error response is sent to the client

**Parameters**

| | |
|---|---|
| *response* | The HTTP Response to send |
| *request* | The HTTP Request |
| *funcPUT* | The function to call in case of access granted |

#### 4.164.2.4 createSecurityCategories()

```
bool ServiceAuthHandler::createSecurityCategories (
            ManagementClient * mgtClient,
            bool dryRun )
```

Create "${service}Security" category with empty content.

**Parameters**

| | |
|---|---|
| *mgtClient* | The management client object |
| *dryRun* | Dryrun so do not register interest in the category |

**Returns**

True on success, False otherwise

#### 4.164.2.5 getAuthenticatedCaller()

```
bool ServiceAuthHandler::getAuthenticatedCaller ( )
```

Return enabled authentication value.

**Returns**

True on success, False otherwise

#### 4.164.2.6 refreshBearerToken()

```
void ServiceAuthHandler::refreshBearerToken ( )
```

Refresh the bearer token of the runnign service This routine is run by a thread started in createSecurityCategories.

After sleep time got in 'exp' of curren token a new one is requested to the core via token_refresh API endpoint

#### 4.164.2.7 securityChange()

```
bool ServiceAuthHandler::securityChange (
            const std::string & payload )  [virtual]
```

Request security change action:

Given a reason code, “attachACL”, “detachACL”, “reloadACL”, “updateACL” in 'reason' atribute, the ACL name in 'argument' could be attached, detached or reloaded

**Parameters**

| | |
|---|---|
| *payload* | The JSON document with 'reason' and 'argument' @retun True on success |

Reimplemented from ServiceHandler.

#### 4.164.2.8 setAuthenticatedCaller()

```
void ServiceAuthHandler::setAuthenticatedCaller (
            bool enabled )
```

Set enabled authentication value.

**Parameters**

| | |
|---|---|
| *enabled* | The enable/disable flag to set |

#### 4.164.2.9 updateSecurityCategory()

```
bool ServiceAuthHandler::updateSecurityCategory (
            const std::string & newCategory )
```

Update the class objects from security category content update.

**Parameters**

| | |
|---|---|
| *category* | The service category name |

**Returns**

True on success, False otherwise

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

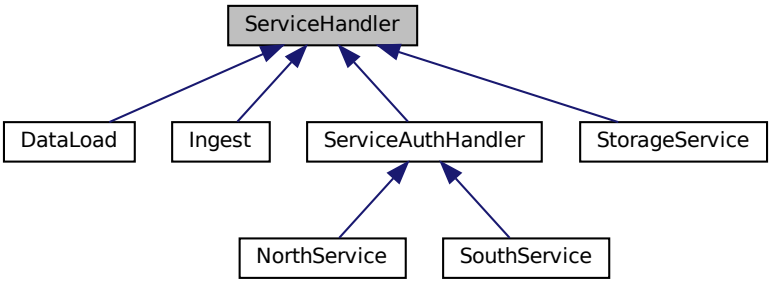
- C/services/common/include/service_handler.h
- C/services/common/service_security.cpp

## 4.165 ServiceHandler Class Reference

ServiceHandler abstract class - the interface that services using the management API must provide.

```
#include <service_handler.h>
```

Inheritance diagram for ServiceHandler:




### Public Member Functions

- virtual void **shutdown** ()=0
- virtual void **restart** ()=0
- virtual void **configChange** (const std::string &category, const std::string &config)=0
- virtual void **configChildCreate** (const std::string &parent_category, const std::string &category, const std↩ ::string &config)=0
- virtual void **configChildDelete** (const std::string &parent_category, const std::string &category)=0
- virtual bool **isRunning** ()=0
- virtual bool **securityChange** (const std::string &payload)

### 4.165.1 Detailed Description

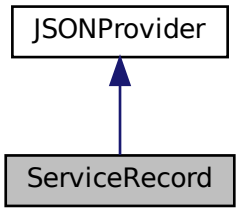
ServiceHandler abstract class - the interface that services using the management API must provide.

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

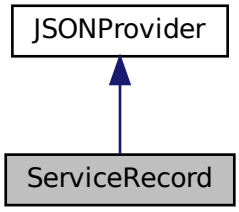
- C/services/common/include/service_handler.h

## 4.166 ServiceRecord Class Reference

Inheritance diagram for ServiceRecord:




Collaboration diagram for ServiceRecord:




### Public Member Functions

- **ServiceRecord** (const std::string &name)
- **ServiceRecord** (const std::string &name, const std::string &type)
- **ServiceRecord** (const std::string &name, const std::string &type, const std::string &protocol, const std::string &address, const unsigned short port, const unsigned short managementPort, const std::string &token="")
- void asJSON (std::string &) const

  *Serialise the service record to json.*
- const std::string & **getName** () const
- const std::string & **getType** () const
- void **setAddress** (const std::string &address)
- void **setPort** (const unsigned short port)
- void **setProtocol** (const std::string &protocol)
- const std::string & **getProtocol** () const
- void **setManagementPort** (const unsigned short managementPort)
- const std::string & **getAddress** ()
- unsigned short **getPort** ()
- bool **operator==** (const ServiceRecord &b) const

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

- C/common/include/service_record.h
- C/common/service_record.cpp

## 4.167 ServiceRegistry Class Reference

ServiceRegistry Singleton class.

```
#include <service_registry.h>
```

### Public Member Functions

- bool registerService (ServiceRecord *service)

  *Register a service with the service registry.*
- bool unRegisterService (ServiceRecord *service)

  *Unregister a service with the service registry.*
- bool **unRegisterService** (const std::string &uuid)
- ServiceRecord * findService (const std::string &name)

  *Find a service that is registered with the service registry.*
- std::string getUUID (ServiceRecord *service)

  *Return the uuid of the registration record for a given service.*

### Static Public Member Functions

- static ServiceRegistry * getInstance ()

  *Return the singleton instance of the service registry.*

### 4.167.1 Detailed Description

ServiceRegistry Singleton class.

### 4.167.2 Member Function Documentation

#### 4.167.2.1 findService()

```
ServiceRecord * ServiceRegistry::findService (
            const std::string & name )
```

Find a service that is registered with the service registry.

**Parameters**

| | |
|---|---|
| *name* | The name of the service to find |

**Returns**

ServiceRecord∗ The service record or null if not found

#### 4.167.2.2 getUUID()

```
string ServiceRegistry::getUUID (
            ServiceRecord * service )
```

Return the uuid of the registration record for a given service.

**Parameters**

| | |
|---|---|
| *service* | The service to return the uuid for |

**Returns**

string The uud of the service registration

**Exceptions**

| | |
|---|---|
| *eception* | If the service could not be found |

#### 4.167.2.3 registerService()

```
bool ServiceRegistry::registerService (
            ServiceRecord * service )
```

Register a service with the service registry.

**Parameters**

| | |
|---|---|
| *service* | The service to register |

**Returns**

bool True if the service was registered

#### 4.167.2.4 unRegisterService()

```
bool ServiceRegistry::unRegisterService (
            ServiceRecord * service )
```

Unregister a service with the service registry.

**Parameters**

| | |
|---|---|
| *service* | The service to unregister |

**Returns**

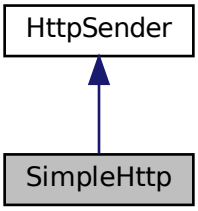
bool True if the service was unregistered

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

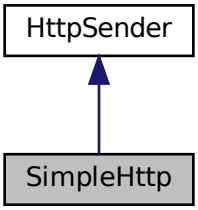
- C/services/core/include/service_registry.h
- C/services/core/service_registry.cpp

## 4.168 SimpleHttp Class Reference

Inheritance diagram for SimpleHttp:




Collaboration diagram for SimpleHttp:



## Public Member Functions

- SimpleHttp (const std::string &host_port, unsigned int connect_timeout=0, unsigned int request_timeout=0, unsigned int retry_sleep_Time=1, unsigned int max_retry=4)

  *Constructor: pass host:port & connect & request timeouts.*
- ~SimpleHttp ()

  *Destructor.*
- void setProxy (const std::string &proxy)

  *Set the host and port of the proxy server.*
- int sendRequest (const std::string &method=std::string(HTTP_SENDER_DEFAULT_METHOD), const std::string &path=std::string(HTTP_SENDER_DEFAULT_PATH), const std::vector< std::pair< std::string, std::string >> &headers={}, const std::string &payload=std::string())

  *HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.*
- void **setAuthMethod** (std::string &authMethod)
- void **setAuthBasicCredentials** (std::string &authBasicCredentials)
- std::string **getHostPort** ()
- std::string **getHTTPResponse** ()
- unsigned int **getMaxRetries** ()
- void **setOCSNamespace** (std::string &OCSNamespace)
- void **setOCSTenantId** (std::string &OCSTenantId)
- void **setOCSClientId** (std::string &OCSClientId)
- void **setOCSClientSecret** (std::string &OCSClientSecret)
- void **setOCSToken** (std::string &OCSToken)
- void **setMaxRetries** (unsigned int retries)

## Additional Inherited Members

## 4.168.1 Member Function Documentation

### 4.168.1.1 sendRequest()

```
int SimpleHttp::sendRequest (
            const std::string & method = std::string(HTTP_SENDER_DEFAULT_METHOD),
            const std::string & path = std::string(HTTP_SENDER_DEFAULT_PATH),
            const std::vector< std::pair< std::string, std::string >> & headers = {},
            const std::string & payload = std::string() )  [virtual]
```

HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.

Send data.

**Parameters**

| | |
|---|---|
| *method* | The HTTP method (GET, POST, ...) |
| *path* | The URL path |
| *headers* | The optional headers to send |
| *payload* | The optional data payload (for POST, PUT) |

**Returns**

The HTTP code on success or 0 on execptions

Implements HttpSender.

#### 4.168.1.2 setProxy()

```
void SimpleHttp::setProxy (
            const std::string & proxy )  [virtual]
```

Set the host and port of the proxy server.

Set a proxy server.

**Parameters**

| | |
|---|---|
| *proxy* | The name and port of the proxy server |

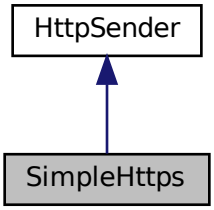
Implements HttpSender.

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

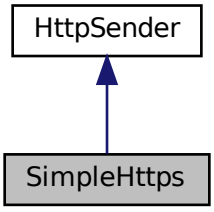
- C/plugins/common/include/simple_http.h
- C/plugins/common/simple_http.cpp

## 4.169 SimpleHttps Class Reference

Inheritance diagram for SimpleHttps:



Collaboration diagram for SimpleHttps:




## Public Member Functions

- SimpleHttps (const std::string &host_port, unsigned int connect_timeout=0, unsigned int request_timeout=0, unsigned int retry_sleep_Time=1, unsigned int max_retry=4)

  *Constructor: pass host:port, connect & request timeouts.*
- ~SimpleHttps ()

  *Destructor.*
- void setProxy (const std::string &proxy)

  *Set the host and port of the proxy server.*
- int sendRequest (const std::string &method=std::string(HTTP_SENDER_DEFAULT_METHOD), const std::string &path=std::string(HTTP_SENDER_DEFAULT_PATH), const std::vector< std::pair< std::string, std::string >> &headers={}, const std::string &payload=std::string())

  *HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.*
- void **setAuthMethod** (std::string &authMethod)
- void **setAuthBasicCredentials** (std::string &authBasicCredentials)
- void **setMaxRetries** (unsigned int retries)
- void **setOCSNamespace** (std::string &OCSNamespace)
- void **setOCSTenantId** (std::string &OCSTenantId)
- void **setOCSClientId** (std::string &OCSClientId)
- void **setOCSClientSecret** (std::string &OCSClientSecret)
- void **setOCSToken** (std::string &OCSToken)
- std::string **getHTTPResponse** ()
- std::string **getHostPort** ()
- unsigned int **getMaxRetries** ()

## Additional Inherited Members

### 4.169.1 Member Function Documentation

#### 4.169.1.1 sendRequest()

```
int SimpleHttps::sendRequest (
            const std::string & method = std::string(HTTP_SENDER_DEFAULT_METHOD),
            const std::string & path = std::string(HTTP_SENDER_DEFAULT_PATH),
            const std::vector< std::pair< std::string, std::string >> & headers = {},
            const std::string & payload = std::string() )  [virtual]
```

HTTP(S) request: pass method and path, HTTP headers and POST/PUT payload.

Send data, it retries the operation m_max_retry times waiting m_retry_sleep_time∗2 at each attempt.

**Parameters**

| | |
|---|---|
| *method* | The HTTP method (GET, POST, ...) |
| *path* | The URL path |
| *headers* | The optional headers to send |
| *payload* | The optional data payload (for POST, PUT) |

**Returns**

The HTTP code for the cases : 1xx Informational / 2xx Success / 3xx Redirection

**Exceptions**

| | |
|---|---|
| *BadRequest* | for HTTP 400 error std::exception as generic exception for all the cases >= 401 Client errors / 5xx Server errors |

Implements HttpSender.

#### 4.169.1.2 setProxy()

```
void SimpleHttps::setProxy (
            const std::string & proxy )  [virtual]
```

Set the host and port of the proxy server.

Set a proxy server.

**Parameters**

| | |
|---|---|
| *proxy* | The name and port of the proxy server |

Implements HttpSender.

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

- C/plugins/common/include/simple_https.h
- C/plugins/common/simple_https.cpp

## 4.170 Sort Class Reference

Sort clause in a selection of records.

```
#include <sort.h>
```

### Public Member Functions

- **Sort** (const std::string &column)
- **Sort** (const std::string &column, bool reverse)
- std::string **toJSON** ()

### 4.170.1 Detailed Description

Sort clause in a selection of records.

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

- C/common/include/sort.h

## 4.171 SouthApi Class Reference

### Public Member Functions

- SouthApi (SouthService *)

  *South API class constructor.*
- ~SouthApi ()

  *Destroy the API.*
- unsigned short getListenerPort ()

  *Return the port the service is listening on.*
- void setPoint (std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Response > response, std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Request > request)

  *Implement the setPoint PUT request.*
- void operation (std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Response > response, std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Request > request)

  *Invoke an operation on the south plugin.*
- void startServer ()

  *Called on the API service thread.*
- void attachDebugger (Response response, Request request)

  *Invoke debugger attach on the south plugin.*
- void detachDebugger (Response response, Request request)

  *Invoke debugger detach on the south plugin.*
- void setDebuggerBuffer (Response response, Request request)

  *Invoke set debugger buffer size on the south plugin.*
- void getDebuggerBuffer (Response response, Request request)

  *Invoke get debugger buffer size on the south plugin.*
- void isolateDebugger (Response response, Request request)

*Invoke isolate debugger handler on the south plugin.*

- void suspendDebugger (Response response, Request request)

  *Invoke suspend debugger handler on the south plugin.*
- void stepDebugger (Response response, Request request)

  *Invoke set debugger step command on the south plugin.*
- void replayDebugger (Response response, Request request)

  *Invoke debugger replay on the south plugin.*
- void stateDebugger (Response response, Request request)

  *Invoke debugger state on the south plugin.*

## 4.171.1 Constructor & Destructor Documentation

### 4.171.1.1 SouthApi()

```
SouthApi::SouthApi (
            SouthService * service )
```

South API class constructor.

**Parameters**

| | |
|---|---|
| *service* | The SouthService class this is the API for |

### 4.171.1.2 ~SouthApi()

```
SouthApi::~SouthApi ( )
```

Destroy the API.

Stop the service and wait fo rthe thread to terminate.

## 4.171.2 Member Function Documentation

### 4.171.2.1 attachDebugger()

```
void SouthApi::attachDebugger (
            Response response,
            Request request )
```

Invoke debugger attach on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.171.2.2 detachDebugger()

```
void SouthApi::detachDebugger (
            Response response,
            Request request )
```

Invoke debugger detach on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.171.2.3 getDebuggerBuffer()

```
void SouthApi::getDebuggerBuffer (
            Response response,
            Request request )
```

Invoke get debugger buffer size on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.171.2.4 isolateDebugger()

```
void SouthApi::isolateDebugger (
            Response response,
            Request request )
```

Invoke isolate debugger handler on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.171.2.5 operation()

```
void SouthApi::operation (
            std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Response > response,
            std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Request > request )
```

Invoke an operation on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.171.2.6 replayDebugger()

```
void SouthApi::replayDebugger (
            Response response,
            Request request )
```

Invoke debugger replay on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.171.2.7 setDebuggerBuffer()

```
void SouthApi::setDebuggerBuffer (
            Response response,
            Request request )
```

Invoke set debugger buffer size on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.171.2.8 setPoint()

```
void SouthApi::setPoint (
            std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Response > response,
            std::shared_ptr< SimpleWeb::Server< SimpleWeb::HTTP >::Request > request )
```

Implement the setPoint PUT request.

Caues the write operation on the south plugin to be called with each of the set point parameters

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.171.2.9 startServer()

```
void SouthApi::startServer ( )
```

Called on the API service thread.

Start the listener for HTTP requests

#### 4.171.2.10 stateDebugger()

```
void SouthApi::stateDebugger (
            Response response,
            Request request )
```

Invoke debugger state on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request - unused |

#### 4.171.2.11 stepDebugger()

```
void SouthApi::stepDebugger (
            Response response,
            Request request )
```

Invoke set debugger step command on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

#### 4.171.2.12 suspendDebugger()

```
void SouthApi::suspendDebugger (
            Response response,
            Request request )
```

Invoke suspend debugger handler on the south plugin.

**Parameters**

| | |
|---|---|
| *response* | The HTTP response |
| *request* | The HTTP request |

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

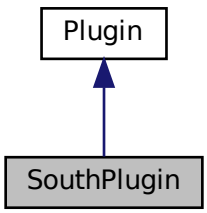
- C/services/south/include/south_api.h
- C/services/south/south_api.cpp

## 4.172 SouthPlugin Class Reference

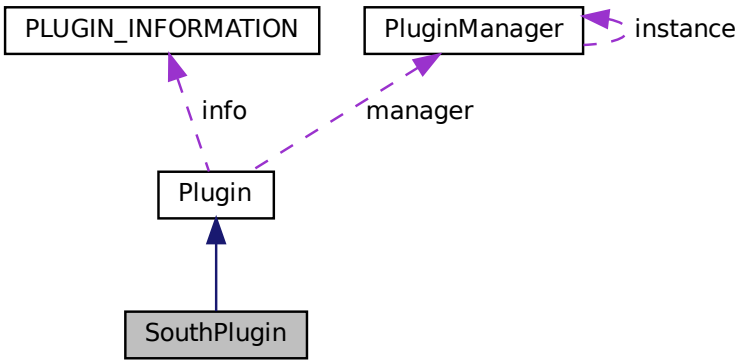
Class that represents a south plugin.

```
#include <south_plugin.h>
```

Inheritance diagram for SouthPlugin:



Collaboration diagram for SouthPlugin:




## Public Member Functions

- SouthPlugin (PLUGIN_HANDLE handle, const ConfigCategory &category)

  *Constructor for the class that wraps the south plugin.*
- ~SouthPlugin ()

  *South plugin destructor.*
- Reading poll ()

  *Call the poll method in the plugin.*
- ReadingSet * pollV2 ()

  *Call the poll method in the plugin supporting interface ver 2.x.*
- void start ()

  *Call the start method in the plugin.*
- void reconfigure (const std::string &)

  *Call the reconfigure method in the plugin.*
- void shutdown ()

  *Call the shutdown method in the plugin.*
- void **registerIngest** (INGEST_CB, void *)
- void **registerIngestV2** (INGEST_CB2, void *)
- bool **isAsync** ()
- bool **hasControl** ()
- bool **persistData** ()
- void startData (const std::string &pluginData)

  *Call the start method in the plugin.*
- std::string shutdownSaveData ()

  *Call the shutdown method in the plugin.*
- bool write (const std::string &name, const std::string &value)

  *Call the write entry point of the plugin.*
- bool operation (const std::string &name, std::vector< PLUGIN_PARAMETER * > &)

  *Call the plugin operation entry point with the operation to execute.*

## Additional Inherited Members

### 4.172.1 Detailed Description

Class that represents a south plugin.

The purpose of this class is to hide the use of the pointers into the dynamically loaded plugin and wrap the interface into a class that can be used directly in the south subsystem.

This is achieved by having a set of private member variables which are the pointers to the functions in the plugin, and a set of public methods that will call these functions via the function pointers.

### 4.172.2 Constructor & Destructor Documentation

#### 4.172.2.1 SouthPlugin()

```
SouthPlugin::SouthPlugin (
            PLUGIN_HANDLE handle,
            const ConfigCategory & category )
```

Constructor for the class that wraps the south plugin.

Create a set of function points that resolve to the loaded plugin and enclose in the class.

### 4.172.3 Member Function Documentation

#### 4.172.3.1 operation()

```
bool SouthPlugin::operation (
            const std::string & name,
            std::vector< PLUGIN_PARAMETER * > &  )
```

Call the plugin operation entry point with the operation to execute.

**Parameters**

| | |
|---|---|
| *name* | The name of the operation |
| *parameters* | The paramters for the operation. |

**Returns**

bool Status of the operation

#### 4.172.3.2 write()

```
bool SouthPlugin::write (
            const std::string & name,
            const std::string & value )
```

Call the write entry point of the plugin.

**Parameters**

| | |
|---|---|
| *name* | The name of the parameter to change |
| *value* | The value to set the parameter |

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

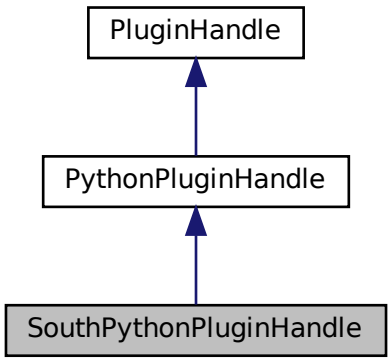
- C/services/south/include/south_plugin.h
- C/services/south/south_plugin.cpp

## 4.173 SouthPythonPluginHandle Class Reference

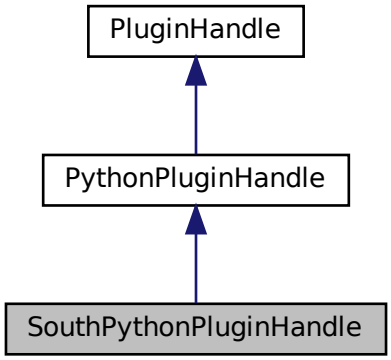
The SouthPythonPluginHandle class is used to represent an interface to a South plugin that is available as a python script.

```
#include <south_python_plugin_handle.h>
```

Inheritance diagram for SouthPythonPluginHandle:



Collaboration diagram for SouthPythonPluginHandle:




## Public Member Functions

- SouthPythonPluginHandle (const char ∗name, const char ∗path)

  *Constructor for PythonPluginHandle.*

## Additional Inherited Members

### 4.173.1 Detailed Description

The SouthPythonPluginHandle class is used to represent an interface to a South plugin that is available as a python script.

### 4.173.2 Constructor & Destructor Documentation

#### 4.173.2.1 SouthPythonPluginHandle()

```
SouthPythonPluginHandle::SouthPythonPluginHandle (
            const char * pluginName,
            const char * pluginPathName )
```

Constructor for PythonPluginHandle.

- Load python interface library and initialize the interface

**Parameters**

| | |
|---|---|
| *pluginName* | The Python plugin name to load |
| *pluginPathName* | The Python plugin path |

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

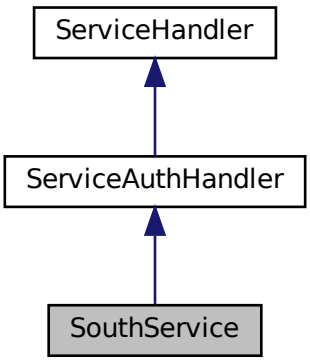
- C/services/common/include/south_python_plugin_handle.h
- C/services/common/south_python_plugin_handle.cpp

## 4.174 SouthService Class Reference

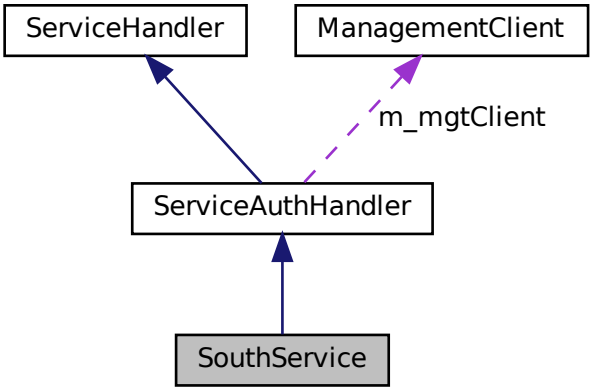
The SouthService class.

```
#include <south_service.h>
```

Inheritance diagram for SouthService:



Collaboration diagram for SouthService:




## Public Member Functions

- SouthService (const std::string &name, const std::string &token="")

  *Constructor for the south service.*
- virtual ~SouthService ()

  *Destructor for south service.*
- void start (std::string &coreAddress, unsigned short corePort)

  *Start the south service.*
- void stop ()

  *Stop the storage service/.*
- void shutdown ()

  *Shutdown request.*
- void restart ()

  *Restart request.*
- void configChange (const std::string &, const std::string &)

  *Configuration change notification using a separate thread.*
- void processConfigChange (const std::string &, const std::string &)

  *Configuration change notification.*
- void **configChildCreate** (const std::string &, const std::string &, const std::string &)
- void **configChildDelete** (const std::string &, const std::string &)
- bool **isRunning** ()
- bool setPoint (const std::string &name, const std::string &value)

  *Perform a setPoint operation on the south plugin.*
- bool operation (const std::string &name, std::vector< PLUGIN_PARAMETER * > &)

  *Perform an operation on the south plugin.*
- void **setDryRun** ()
- void handlePendingReconf ()

  *Handle configuration change notification; called by reconf thread Waits for some reconf operation(s) to get queued up, then works thru' them.*
- bool **attachDebugger** ()
- void **detachDebugger** ()

- void **setDebuggerBuffer** (unsigned int size)
- std::string **getDebuggerBuffer** ()
- void **suspendDebugger** (bool suspend)
- void **isolateDebugger** (bool isolate)
- void **stepDebugger** (unsigned int steps)
- void **replayDebugger** ()
- std::string debugState ()

  *Return the state of the pipeline debugger.*
- bool **debuggerAttached** ()
- bool **allowControl** ()
- bool **allowDebugger** ()

## Additional Inherited Members

### 4.174.1 Detailed Description

The SouthService class.

This class is the core of the service that provides south side services to Fledge.

### 4.174.2 Member Function Documentation

#### 4.174.2.1 configChange()

```
void SouthService::configChange (
            const std::string & ,
            const std::string &  )  [virtual]
```

Configuration change notification using a separate thread.

**Parameters**

| | |
|---|---|
| *categoryName* | Category name |
| *category* | Category value |

Implements ServiceHandler.

#### 4.174.2.2 debugState()

```
string SouthService::debugState ( )
```

Return the state of the pipeline debugger.

**Returns**

string JSON document reporting the state of the pipeline debugger

#### 4.174.2.3 operation()

```
bool SouthService::operation (
            const std::string & name,
            std::vector< PLUGIN_PARAMETER * > &  )
```

Perform an operation on the south plugin.

**Parameters**

| | |
|---|---|
| *name* | Name of the operation |
| *params* | The parameters for the operaiton, if any |

**Returns**

Success or failure of the operation

#### 4.174.2.4 processConfigChange()

```
void SouthService::processConfigChange (
            const std::string & ,
            const std::string &  )
```

Configuration change notification.

**Parameters**

| | |
|---|---|
| *categoryName* | Category name |
| *category* | Category value |

#### 4.174.2.5 setPoint()

```
bool SouthService::setPoint (
            const std::string & name,
            const std::string & value )
```

Perform a setPoint operation on the south plugin.

**Parameters**

| | |
|---|---|
| *name* | Name of the point to set |
| *value* | The value to set |

**Returns**

Success or failure of the SetPoint operation

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

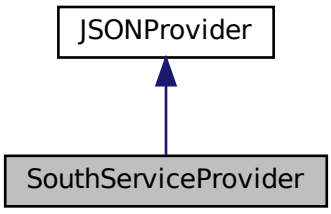
- C/services/south/include/south_service.h
- C/services/south/south.cpp

## 4.175 SouthServiceProvider Class Reference

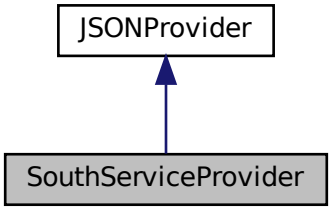
A data provider class to return data in the south service ping response.

```
#include <south_service.h>
```

Inheritance diagram for SouthServiceProvider:




Collaboration diagram for SouthServiceProvider:



### Public Member Functions

- **SouthServiceProvider** (SouthService *south)
- void **asJSON** (std::string &json) const

### 4.175.1 Detailed Description

A data provider class to return data in the south service ping response.

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

- C/services/south/include/south_service.h

## 4.176 SQLBuffer Class Reference

Buffer class designed to hold SQL statement that can as required but have minimal copy semantics.

```
#include <sql_buffer.h>
```

### Public Member Functions

- SQLBuffer ()

  *Buffer class designed to hold SQL statement that can as required but have minimal copy semantics.*
- ~SQLBuffer ()

  *SQLBuffer destructor.*
- bool **isEmpty** ()
- void append (const char)

  *Append a character to a buffer.*
- void append (const char *)

  *Append a character string to a buffer.*
- void append (const int)

  *Append an integer to a buffer.*
- void append (const unsigned int)

  *Append an unsigned integer to a buffer.*
- void append (const long)

  *Append a long to a buffer.*
- void append (const unsigned long)

  *Append an unsigned long to a buffer.*
- void append (const double)

  *Append a double to a buffer.*
- void **append** (const std::string &)
- void quote (const std::string &)

  *Quote and append a string to a buffer.*
- const char * coalesce ()

  *Create a coalesced buffer from the buffer chain.*
- void clear ()

  *Clear all the buffers from the SQLBuffer and allow it to be reused.*

### 4.176.1 Detailed Description

Buffer class designed to hold SQL statement that can as required but have minimal copy semantics.

### 4.176.2 Constructor & Destructor Documentation

#### 4.176.2.1 SQLBuffer()

```
SQLBuffer::SQLBuffer ( )
```

Buffer class designed to hold SQL statement that can as required but have minimal copy semantics.

SQLBuffer constructor

### 4.176.3 Member Function Documentation

#### 4.176.3.1 append() [1/7]

```
void SQLBuffer::append (
            const char * data )
```

Append a character string to a buffer.

@para data The string to append to the buffer

#### 4.176.3.2 append() [2/7]

```
void SQLBuffer::append (
            const char data )
```

Append a character to a buffer.

**Parameters**

| | |
|---|---|
| *data* | The character to append to the buffer |

#### 4.176.3.3 append() [3/7]

```
void SQLBuffer::append (
            const double value )
```

Append a double to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The double value to append to the buffer |

#### 4.176.3.4 append() [4/7]

```
void SQLBuffer::append (
            const int value )
```

Append an integer to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The value to append to the buffer |

#### 4.176.3.5 append() [5/7]

```
void SQLBuffer::append (
            const long value )
```

Append a long to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The long value to append to the buffer |

#### 4.176.3.6 append() [6/7]

```
void SQLBuffer::append (
            const unsigned int value )
```

Append an unsigned integer to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The unsigned long value to append to the buffer |

#### 4.176.3.7 append() [7/7]

```
void SQLBuffer::append (
            const unsigned long value )
```

Append an unsigned long to a buffer.

**Parameters**

| | |
|---|---|
| *value* | The value to append to the buffer |

#### 4.176.3.8 coalesce()

```
const char * SQLBuffer::coalesce ( )
```

Create a coalesced buffer from the buffer chain.

The buffer returned has been created usign the new[] operator and must be deleted by the caller.

**Returns**

char∗ The SQL statement in a single buffer

#### 4.176.3.9 quote()

```
void SQLBuffer::quote (
            const std::string &  )
```

Quote and append a string to a buffer.

**Parameters**

| | |
|---|---|
| *str* | The string to quote and append to the buffer |

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

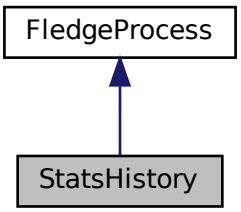
- C/plugins/storage/common/include/sql_buffer.h
- C/plugins/storage/common/sql_buffer.cpp

## 4.177 StatsHistory Class Reference

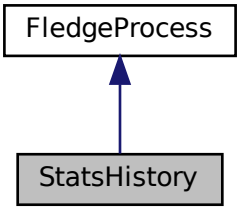
StatisticsHisotry class.

```
#include <stats_history.h>
```

Inheritance diagram for StatsHistory:




Collaboration diagram for StatsHistory:




## Public Member Functions

- StatsHistory (int argc, char **argv)

  *Constructor for Statistics history task.*
- ~StatsHistory ()

  *StatsHistory class methods.*
- void run () const

  *Statisitics History run method, called by the base class to start the process and do the actual work.*

## Additional Inherited Members

### 4.177.1 Detailed Description

StatisticsHisotry class.

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

- C/tasks/statistics_history/include/stats_history.h
- C/tasks/statistics_history/stats_history.cpp

## 4.178 STORAGE_CONFIGURATION Struct Reference

```
#include <readings_catalogue.h>
```

### Public Attributes

- int **poolSize** = 5
- int **nReadingsPerDb** = 14
- int **nDbPreallocate** = 3
- int **nDbLeftFreeBeforeAllocate** = 1
- int **nDbToAllocate** = 2

### 4.178.1 Detailed Description

- poolSize = Number of connections to allocate
- nReadingsPerDb = Number of readings tables per database
- nDbPreallocate = Number of databases to allocate in advance
- nDbLeftFreeBeforeAllocate = Number of free databases before a new allocation is executed
- nDbToAllocate = Number of database to allocate each time

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

- C/plugins/storage/sqlite/common/include/readings_catalogue.h

## 4.179 StorageApi Class Reference

The Storage API class - this class is responsible for the registration of all API entry points in the storage API and the dispatch of those API calls to the internals of the storage service and the storage plugin itself.

```
#include <storage_api.h>
```

### Public Member Functions

- StorageApi (const unsigned short port, const unsigned int threads, const unsigned int workerPoolSize)

  *Construct the singleton Storage API.*
- ~StorageApi ()

  *Destructor for the storage API class.*
- void initResources ()

  *Initialise the API entry points for the common data resource and the readings resource.*
- void setPlugin (StoragePlugin *)

  *Connect with the storage plugin.*
- void setReadingPlugin (StoragePlugin *)

  *Connect with the storage plugin.*
- void start ()

  *Start the HTTP server.*

- void **startServer** ()
- void wait ()

  *Wait for the HTTP server to shutdown.*
- void **stopServer** ()
- unsigned short getListenerPort ()

  *Return the current listener port.*
- void commonInsert (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform an insert into a table of the data provided in the payload.*
- void commonSimpleQuery (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::↩ Request > request)

  *Perform a simple query on the table using the query parameters as conditions TODO make this work for multiple column queries.*
- void commonQuery (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform query on a table using the JSON encoded query in the payload.*
- void commonUpdate (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform an update on a table of the data provided in the payload.*
- void commonDelete (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform a delete on a table using the condition encoded in the JSON payload.*
- void defaultResource (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Handle a bad URL endpoint call.*
- void readingAppend (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform an append operation on the readings.*
- void readingFetch (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Fetch a block of readings.*
- void readingQuery (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform a query on a set of readings.*
- void readingPurge (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Purge the readings.*
- void readingRegister (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Register interest in readings for an asset.*
- void readingUnregister (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Unregister interest in readings for an asset.*
- void tableRegister (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Register interest in readings for an asset.*
- void tableUnregister (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Unregister interest in readings for an asset.*
- void createTableSnapshot (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::↩ Request > request)

  *Create a table snapshot.*

- void loadTableSnapshot (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Load a table snapshot.*
- void deleteTableSnapshot (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Delete a table snapshot.*
- void getTableSnapshots (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Get list of a table snapshots.*
- void createStorageStream (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Create a stream for high speed storage ingestion.*
- bool readingStream (ReadingStream **readings, bool commit)

  *Append the readings that have arrived via a stream to the storage plugin.*
- void createStorageSchema (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform an create table and create index for schema provided in the payload.*
- void storageTableInsert (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform an insert table operation.*
- void storageTableUpdate (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform an update on a table of the data provided in the payload.*
- void storageTableDelete (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform a delete on a table using the condition encoded in the JSON payload.*
- void storageTableSimpleQuery (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform a simple query on the table using the query parameters as conditions TODO make this work for multiple column queries.*
- void storageTableQuery (shared_ptr< HttpServer::Response > response, shared_ptr< HttpServer::Request > request)

  *Perform query on a table using the JSON encoded query in the payload.*
- void **printList** ()
- bool **createSchema** (const std::string &schema)
- void **setTimeout** (long timeout)
- StoragePlugin * **getStoragePlugin** ()
- StoragePerformanceMonitor * **getPerformanceMonitor** ()
- void worker ()

  *The worker thread.*
- void queue (StorageOperation::Operations op, shared_ptr< HttpServer::Request > request, shared_ptr< HttpServer::Response > response)

  *Append a request to the readings request queue.*

## Static Public Member Functions

- static StorageApi * getInstance ()

  *Return the singleton instance of the StorageAPI class.*

## Public Attributes

- std::atomic< int > **m_workers_count**

## 4.179.1 Detailed Description

The Storage API class - this class is responsible for the registration of all API entry points in the storage API and the dispatch of those API calls to the internals of the storage service and the storage plugin itself.

## 4.179.2 Constructor & Destructor Documentation

### 4.179.2.1 ~StorageApi()

```
StorageApi::~StorageApi ( )
```

Destructor for the storage API class.

There is only ever one StorageApi class in existance and it lives for the entire duration of the storage service, so this is really for completeness rather than any pracitical use.

## 4.179.3 Member Function Documentation

### 4.179.3.1 commonDelete()

```
void StorageApi::commonDelete (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform a delete on a table using the condition encoded in the JSON payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

### 4.179.3.2 commonInsert()

```
void StorageApi::commonInsert (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform an insert into a table of the data provided in the payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.3 commonQuery()

```
void StorageApi::commonQuery (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform query on a table using the JSON encoded query in the payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.4 commonSimpleQuery()

```
void StorageApi::commonSimpleQuery (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform a simple query on the table using the query parameters as conditions TODO make this work for multiple column queries.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.5 commonUpdate()

```
void StorageApi::commonUpdate (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform an update on a table of the data provided in the payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.6 createStorageSchema()

```
void StorageApi::createStorageSchema (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform an create table and create index for schema provided in the payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.7 createStorageStream()

```
void StorageApi::createStorageStream (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Create a stream for high speed storage ingestion.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.8 queue()

```
void StorageApi::queue (
            StorageOperation::Operations op,
            shared_ptr< HttpServer::Request > request,
            shared_ptr< HttpServer::Response > response )
```

Append a request to the readings request queue.

If the queue is starting to get long delay the return as a primitive way to throttle incoming requests

**Parameters**

| | |
|---|---|
| *op* | The operation to perform |
| *request* | The HTTP request |
| *response* | The HTTP response |

#### 4.179.3.9 readingAppend()

```
void StorageApi::readingAppend (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform an append operation on the readings.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.10 readingFetch()

```
void StorageApi::readingFetch (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Fetch a block of readings.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.11 readingPurge()

```
void StorageApi::readingPurge (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Purge the readings.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

Handle PluginNotImplementedException exception here

Return HTTP code 400 with message from storage plugin

Handle general exception

#### 4.179.3.12 readingQuery()

```
void StorageApi::readingQuery (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform a query on a set of readings.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.13 readingStream()

```
bool StorageApi::readingStream (
            ReadingStream ** readings,
            bool commit )
```

Append the readings that have arrived via a stream to the storage plugin.

**Parameters**

| | |
|---|---|
| *readings* | A Null terminated array of points to ReadingStream structures |
| *commit* | A flag to commit the readings block |

#### 4.179.3.14 storageTableDelete()

```
void StorageApi::storageTableDelete (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform a delete on a table using the condition encoded in the JSON payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.15 storageTableInsert()

```
void StorageApi::storageTableInsert (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform an insert table operation.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.16 storageTableQuery()

```
void StorageApi::storageTableQuery (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform query on a table using the JSON encoded query in the payload.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.17 storageTableSimpleQuery()

```
void StorageApi::storageTableSimpleQuery (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform a simple query on the table using the query parameters as conditions TODO make this work for multiple column queries.

**Parameters**

| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

#### 4.179.3.18 storageTableUpdate()

```
void StorageApi::storageTableUpdate (
            shared_ptr< HttpServer::Response > response,
            shared_ptr< HttpServer::Request > request )
```

Perform an update on a table of the data provided in the payload.

**Parameters**

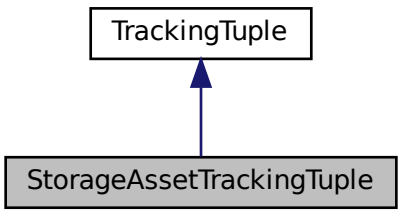
| | |
|---|---|
| *response* | The response stream to send the response on |
| *request* | The HTTP request |

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

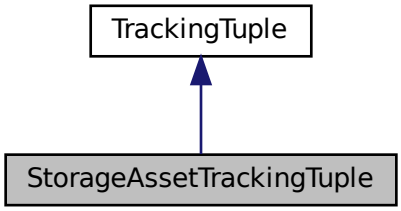
- C/services/storage/include/storage_api.h
- C/services/storage/storage_api.cpp

## 4.180 StorageAssetTrackingTuple Class Reference

Inheritance diagram for StorageAssetTrackingTuple:



Collaboration diagram for StorageAssetTrackingTuple:




## Public Member Functions

- **StorageAssetTrackingTuple** (const std::string &service, const std::string &plugin, const std::string &asset, const std::string &event, const bool &deprecated=false, const std::string &datapoints="", unsigned int c=0)
- bool **operator==** (const StorageAssetTrackingTuple &x) const
- std::string **assetToString** ()
- bool **isDeprecated** ()
- unsigned int **getMaxCount** ()
- std::string **getDataPoints** ()
- void **unDeprecate** ()
- void **setDeprecate** ()
- InsertValues processData (bool storage, ManagementClient *mgtClient, bool &warned, std::string &instanceName)

  *Insert StorageAssetTrackingTuple data via Fledge core API or prepare InsertValues object for direct DB operation.*

## Public Attributes

- std::string **m_datapoints**
- unsigned int **m_maxCount**
- std::string **m_serviceName**
- std::string **m_pluginName**
- std::string **m_assetName**
- std::string **m_eventName**

## 4.180.1 Member Function Documentation

### 4.180.1.1 processData()

```
InsertValues StorageAssetTrackingTuple::processData (
            bool storage,
            ManagementClient * mgtClient,
            bool & warned,
            std::string & instanceName )  [virtual]
```

Insert StorageAssetTrackingTuple data via Fledge core API or prepare InsertValues object for direct DB operation.

**Parameters**

| | |
|---|---|
| *storage* | Boolean for storage being available |
| *mgtClient* | ManagementClient object pointer |
| *warned* | Boolean ireference updated for logging operation |
| *instanceName* | Fledge instance name |

**Returns**

InsertValues object

Implements TrackingTuple.

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

- C/common/include/asset_tracking.h
- C/common/asset_tracking.cpp

## 4.181 StorageAssetTrackingTuplePtrEqual Struct Reference

### Public Member Functions

- bool **operator()** (StorageAssetTrackingTuple const ∗a, StorageAssetTrackingTuple const ∗b) const

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

- C/common/include/asset_tracking.h

## 4.182 StorageClient Class Reference

Client for accessing the storage service.

```
#include <storage_client.h>
```

## Public Member Functions

- StorageClient (HttpClient ∗client)

  *Storage Client constructor stores the provided HttpClient into the map.*
- **StorageClient** (const std::string &hostname, const unsigned short port)
- ∼StorageClient ()

  *Destructor for storage client.*
- ResultSet ∗ queryTable (const std::string &schema, const std::string &tablename, const Query &query)

  *Query a table.*
- ResultSet ∗ queryTable (const std::string &tablename, const Query &query)

  *Query a table.*
- ReadingSet ∗ queryTableToReadings (const std::string &tableName, const Query &query)

  *Query a table and return a ReadingSet pointer.*
- int **insertTable** (const std::string &schema, const std::string &tableName, const InsertValues &values)
- int **insertTable** (const std::string &schema, const std::string &tableName, const std::vector< InsertValues > &values)
- int **insertTable** (const std::string &tableName, const std::vector< InsertValues > &values)
- int **updateTable** (const std::string &schema, const std::string &tableName, const InsertValues &values, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &schema, const std::string &tableName, const JSONProperties &json, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &schema, const std::string &tableName, const InsertValues &values, const JSONProperties &json, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &schema, const std::string &tableName, const ExpressionValues &values, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &schema, const std::string &tableName, std::vector< std::pair< ExpressionValues ∗, Where ∗ >> &updates, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &schema, const std::string &tableName, const InsertValues &values, const ExpressionValues &expressoins, const Where &where, const UpdateModifier ∗modifier=NULL)
- int deleteTable (const std::string &schema, const std::string &tableName, const Query &query)

  *Delete from a table.*
- int **insertTable** (const std::string &tableName, const InsertValues &values)
- int **updateTable** (const std::string &tableName, const InsertValues &values, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &tableName, const JSONProperties &json, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &tableName, const InsertValues &values, const JSONProperties &json, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &tableName, const ExpressionValues &values, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &tableName, std::vector< std::pair< ExpressionValues ∗, Where ∗ >> &updates, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &tableName, const InsertValues &values, const ExpressionValues &expressions, const Where &where, const UpdateModifier ∗modifier=NULL)
- int **updateTable** (const std::string &schema, const std::string &tableName, std::vector< std::pair< InsertValue ∗, Where ∗ > > &updates, const UpdateModifier ∗modifier)
- int **updateTable** (const std::string &tableName, std::vector< std::pair< InsertValue ∗, Where ∗ > > &updates, const UpdateModifier ∗modifier=NULL)
- int deleteTable (const std::string &tableName, const Query &query)

  *Delete from a table.*
- bool readingAppend (Reading &reading)

  *Append a single reading.*
- bool readingAppend (const std::vector< Reading ∗ > &readings)

  *Append multiple readings.*

- ResultSet * readingQuery (const Query &query)

  *Perform a generic query against the readings data.*
- ReadingSet * readingQueryToReadings (const Query &query)

  *Perform a generic query against the readings data, returning ReadingSet object.*
- ReadingSet * readingFetch (const unsigned long readingId, const unsigned long count)

  *Retrieve a set of readings for sending on the northbound interface of Fledge.*
- PurgeResult readingPurgeByAge (unsigned long age, unsigned long sent, bool purgeUnsent)

  *Purge the readings by age.*
- PurgeResult readingPurgeBySize (unsigned long size, unsigned long sent, bool purgeUnsent)

  *Purge the readings by size.*
- PurgeResult readingPurgeByAsset (const std::string &asset)

  *Purge the readings by asset name.*
- bool registerAssetNotification (const std::string &assetName, const std::string &callbackUrl)

  *Register interest for a Reading asset name.*
- bool unregisterAssetNotification (const std::string &assetName, const std::string &callbackUrl)

  *Unregister interest for a Reading asset name.*
- bool registerTableNotification (const std::string &tableName, const std::string &key, std::vector< std::string > keyValues, const std::string &operation, const std::string &callbackUrl)

  *Register interest for a table.*
- bool unregisterTableNotification (const std::string &tableName, const std::string &key, std::vector< std::string > keyValues, const std::string &operation, const std::string &callbackUrl)

  *Unregister interest for a table name.*
- void **registerManagement** (ManagementClient *mgmnt)
- bool createSchema (const std::string &)

  *Function to create Storage Schema.*
- bool deleteHttpClient ()

  *Delete HttpClient object for current thread.*

## 4.182.1 Detailed Description

Client for accessing the storage service.

## 4.182.2 Member Function Documentation

### 4.182.2.1 deleteTable() [1/2]

```
int StorageClient::deleteTable (
            const std::string & schema,
            const std::string & tableName,
            const Query & query )
```

Delete from a table.

**Parameters**

| | |
|---|---|
| *schema* | The name of the schema to delete from |
| *tablename* | The name of the table to delete from |
| *query* | The query payload to match rows to delete |

**Returns**

int The number of rows deleted

#### 4.182.2.2 deleteTable() [2/2]

```
int StorageClient::deleteTable (
            const std::string & tableName,
            const Query & query )
```

Delete from a table.

**Parameters**

| | |
|---|---|
| *tablename* | The name of the table to delete from |
| *query* | The query payload to match rows to delete |

**Returns**

int The number of rows deleted

#### 4.182.2.3 queryTable() [1/2]

```
ResultSet * StorageClient::queryTable (
            const std::string & schema,
            const std::string & tableName,
            const Query & query )
```

Query a table.

**Parameters**

| | |
|---|---|
| *schema* | The name of the schema to query |
| *tablename* | The name of the table to query |
| *query* | The query payload |

**Returns**

ResultSet∗ The resultset of the query

#### 4.182.2.4 queryTable() [2/2]

```
ResultSet * StorageClient::queryTable (
            const std::string & tableName,
            const Query & query )
```

Query a table.

**Parameters**

| | |
|---|---|
| *tablename* | The name of the table to query |
| *query* | The query payload |

**Returns**

ResultSet∗ The resultset of the query

#### 4.182.2.5 queryTableToReadings()

```
ReadingSet * StorageClient::queryTableToReadings (
            const std::string & tableName,
            const Query & query )
```

Query a table and return a ReadingSet pointer.

**Parameters**

| | |
|---|---|
| *tablename* | The name of the table to query |
| *query* | The query payload |

**Returns**

ReadingSet∗ The resultset of the query as ReadingSet class pointer

#### 4.182.2.6 readingAppend()

```
bool StorageClient::readingAppend (
            const std::vector< Reading * > & readings )
```

Append multiple readings.

TODO implement a mechanism to force streamed or non-streamed mode

#### 4.182.2.7 readingFetch()

```
ReadingSet * StorageClient::readingFetch (
            const unsigned long readingId,
            const unsigned long count )
```

Retrieve a set of readings for sending on the northbound interface of Fledge.

**Parameters**

| | |
|---|---|
| *reading↩ Id* | The ID of the reading which should be the first one to send |
| *count* | Maximum number if readings to return |

**Returns**

ReadingSet The set of readings

#### 4.182.2.8 readingPurgeByAge()

```
PurgeResult StorageClient::readingPurgeByAge (
            unsigned long age,
            unsigned long sent,
            bool purgeUnsent )
```

Purge the readings by age.

**Parameters**

| | |
|---|---|
| *age* | Number of hours old a reading has to be to be considered for purging |
| *sent* | The ID of the last reading that was sent |
| *purgeUnsent* | Flag to control if unsent readings should be purged |

**Returns**

PurgeResult Data on the readings hat were purged

#### 4.182.2.9 readingPurgeByAsset()

```
PurgeResult StorageClient::readingPurgeByAsset (
            const std::string & asset )
```

Purge the readings by asset name.

**Parameters**

| | |
|---|---|
| *asset* | The name of the asset to purge |

**Returns**

PurgeResult Data on the readings that were purged

#### 4.182.2.10 readingPurgeBySize()

```
PurgeResult StorageClient::readingPurgeBySize (
            unsigned long size,
            unsigned long sent,
            bool purgeUnsent )
```

Purge the readings by size.

**Parameters**

| | |
|---|---|
| *size* | Desired maximum size of readings table |
| *sent* | The ID of the last reading that was sent |
| *purgeUnsent* | Flag to control if unsent readings should be purged |

**Returns**

PurgeResult Data on the readings hat were purged

#### 4.182.2.11 readingQuery()

```
ResultSet * StorageClient::readingQuery (
            const Query & query )
```

Perform a generic query against the readings data.

**Parameters**

| | |
|---|---|
| *query* | The query to execute |

**Returns**

ResultSet The result of the query

#### 4.182.2.12 readingQueryToReadings()

```
ReadingSet * StorageClient::readingQueryToReadings (
            const Query & query )
```

Perform a generic query against the readings data, returning ReadingSet object.

**Parameters**

| | |
|---|---|
| *query* | The query to execute |

**Returns**

ReadingSet The result of the query

#### 4.182.2.13 registerAssetNotification()

```
bool StorageClient::registerAssetNotification (
            const std::string & assetName,
            const std::string & callbackUrl )
```

Register interest for a Reading asset name.

**Parameters**

| | |
|---|---|
| *assetName* | The asset name to register for readings data notification |
| *callbackUrl* | The callback URL to send readings data. |

**Returns**

True on success, false otherwise.

#### 4.182.2.14 registerTableNotification()

```
bool StorageClient::registerTableNotification (
            const std::string & tableName,
            const std::string & key,
            std::vector< std::string > keyValues,
            const std::string & operation,
            const std::string & callbackUrl )
```

Register interest for a table.

**Parameters**

| | |
|---|---|
| *tableName* | The table name to register for notification |
| *tableKey* | The key of interest in the table |
| *tableKeyValues* | The key values of interest |
| *tableOperation* | The table operation of interest (insert/update/delete) |
| *callbackUrl* | The callback URL to send change data |

**Returns**

True on success, false otherwise.

#### 4.182.2.15 unregisterAssetNotification()

```
bool StorageClient::unregisterAssetNotification (
            const std::string & assetName,
            const std::string & callbackUrl )
```

Unregister interest for a Reading asset name.

**Parameters**

| | |
|---|---|
| *assetName* | The asset name to unregister for readings data notification |
| *callbackUrl* | The callback URL provided in registration. |

**Returns**

True on success, false otherwise.

#### 4.182.2.16 unregisterTableNotification()

```
bool StorageClient::unregisterTableNotification (
            const std::string & tableName,
            const std::string & key,
            std::vector< std::string > keyValues,
            const std::string & operation,
            const std::string & callbackUrl )
```

Unregister interest for a table name.

**Parameters**

| | |
|---|---|
| *tableName* | The table name to unregister interest in |
| *tableKey* | The key of interest in the table |
| *tableKeyValues* | The key values of interest |
| *tableOperation* | The table operation of interest (insert/update/delete) |
| *callbackUrl* | The callback URL to send change data |

**Returns**

True on success, false otherwise.

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

- C/common/include/storage_client.h
- C/common/storage_client.cpp

## 4.183 StorageConfiguration Class Reference

The storage service must handle its own configuration differently to other services as it is unable to read the configuration from the database.

```
#include <configuration.h>
```

### Public Member Functions

- StorageConfiguration ()

  *Constructor for storage service configuration class.*
- ~StorageConfiguration ()

  *Storage configuration destructor.*
- const char * getValue (const std::string &key)

  *Return a value from the cached configuration category.*
- bool hasValue (const std::string &key)

  *Return if a value exsits for the cached configuration category.*
- bool setValue (const std::string &key, const std::string &value)

  *Set the value of a configuration item.*
- void updateCategory (const std::string &json)

  *Called when the configuration category is updated.*
- DefaultConfigCategory * getDefaultCategory ()

  *Return the default category to register with the core.*

### 4.183.1 Detailed Description

The storage service must handle its own configuration differently to other services as it is unable to read the configuration from the database.

The configuration is required in order to connnect to the database. Therefore it keeps a shadow copy in a local file and it keeps this local, cached copy up to date by registering interest in the category and whenever a chaneg is made writing the category to the local cache file.

### 4.183.2 Constructor & Destructor Documentation

#### 4.183.2.1 StorageConfiguration()

```
StorageConfiguration::StorageConfiguration ( )
```

Constructor for storage service configuration class.

Update options in deafult configuration for items 'plugin' and 'readingPlugin' with installed plugins

### 4.183.3 Member Function Documentation

#### 4.183.3.1 getDefaultCategory()

```
DefaultConfigCategory * StorageConfiguration::getDefaultCategory ( )
```

Return the default category to register with the core.

This allows the storage configuration to appear in the UI

**Returns**

DefaultConfigCategory∗ The default configuration category

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

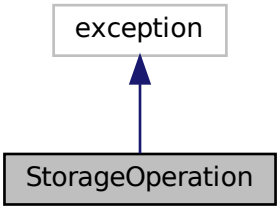
- C/services/storage/include/configuration.h
- C/services/storage/configuration.cpp

## 4.184 StorageOperation Class Reference

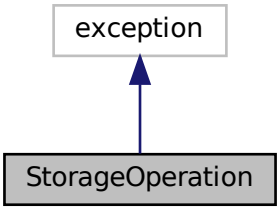
StorageOperation.

```
#include <configuration_manager.h>
```

Inheritance diagram for StorageOperation:




Collaboration diagram for StorageOperation:



### Public Types

- enum **Operations** { **ReadingAppend**, **ReadingPurge**, **ReadingFetch**, **ReadingQuery** }

### Public Member Functions

- virtual const char ∗ **what** () const throw ()
- **StorageOperation** (StorageOperation::Operations operation, shared_ptr< HttpServer::Request > request, shared_ptr< HttpServer::Response > response)

### Public Attributes

- StorageOperation::Operations **m_operation**
- shared_ptr< HttpServer::Request > **m_request**
- shared_ptr< HttpServer::Response > **m_response**

### 4.184.1 Detailed Description

StorageOperation.

Class used to queue the operations to be executed by the worker thread pool.

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

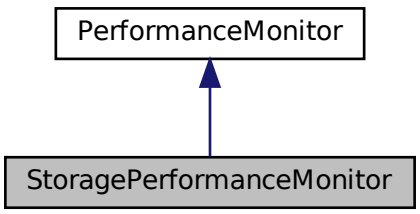
- C/services/core/include/configuration_manager.h
- C/services/storage/include/storage_api.h

## 4.185 StoragePerformanceMonitor Class Reference

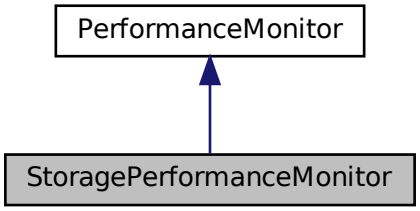
StoragePerformanceMonitor is a derived class from PerformanceMonitor It allows direct writing of monitoring data to database.

```
#include <storage_api.h>
```

Inheritance diagram for StoragePerformanceMonitor:



Collaboration diagram for StoragePerformanceMonitor:




### Public Member Functions

- **StoragePerformanceMonitor** (const std::string &name, StorageApi *api)
- void **writeData** (const std::string &table, const InsertValues &values)

### 4.185.1 Detailed Description

StoragePerformanceMonitor is a derived class from PerformanceMonitor It allows direct writing of monitoring data to database.

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

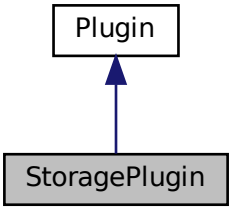
- C/services/storage/include/storage_api.h

## 4.186 StoragePlugin Class Reference

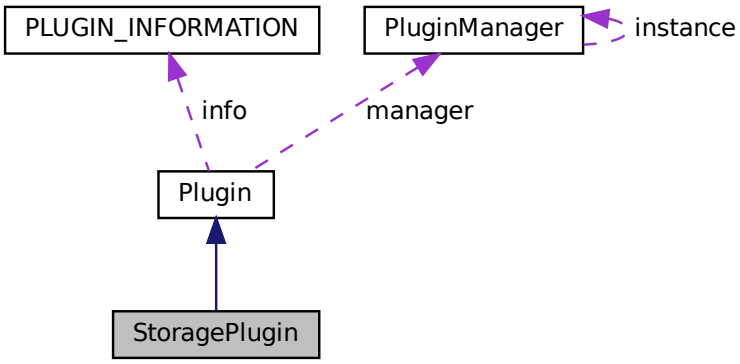
Class that represents a storage plugin.

```
#include <storage_plugin.h>
```

Inheritance diagram for StoragePlugin:



Collaboration diagram for StoragePlugin:




## Public Member Functions

- StoragePlugin (const std::string &name, PLUGIN_HANDLE handle)

  *Constructor for the class that wraps the storage plugin.*
- int commonInsert (const std::string &table, const std::string &payload, const char ∗schema=nullptr)

  *Call the insert method in the plugin.*
- char ∗ commonRetrieve (const std::string &table, const std::string &payload, const char ∗schema=nullptr)

  *Call the retrieve method in the plugin.*
- int commonUpdate (const std::string &table, const std::string &payload, const char ∗schema=nullptr)

  *Call the update method in the plugin.*
- int commonDelete (const std::string &table, const std::string &payload, const char ∗schema=nullptr)

  *Call the delete method in the plugin.*
- int readingsAppend (const std::string &payload)

  *Call the readings append method in the plugin.*
- char ∗ readingsFetch (unsigned long id, unsigned int blksize)

  *Call the readings fetch method in the plugin.*
- char ∗ readingsRetrieve (const std::string &payload)

  *Call the readings retrieve method in the plugin.*
- char ∗ readingsPurge (unsigned long age, unsigned int flags, unsigned long sent)

  *Call the readings purge method in the plugin.*
- long ∗ **readingsPurge** ()
- char ∗ readingsPurgeAsset (const std::string &asset)

  *Call the readings purge asset method in the plugin.*
- void release (const char ∗response)

  *Release a result from a retrieve.*
- int createTableSnapshot (const std::string &table, const std::string &id)

  *Call the create table snaphot method in the plugin.*
- int loadTableSnapshot (const std::string &table, const std::string &id)

  *Call the load table snaphot method in the plugin.*
- int deleteTableSnapshot (const std::string &table, const std::string &id)

  *Call the delete table snaphot method in the plugin.*

- char * getTableSnapshots (const std::string &table)

  *Call the get table snaphot method in the plugin.*
- PLUGIN_ERROR * lastError ()

  *Get the last error from the plugin.*
- bool **hasStreamSupport** ()
- int readingStream (ReadingStream **stream, bool commit)

  *Call the reading stream method in the plugin.*
- bool pluginShutdown ()

  *Call the shutdown entry point of the plugin.*
- int createSchema (const std::string &payload)

  *Call the schema create method in the plugin.*
- StoragePluginConfiguration * **getConfig** ()
- const std::string & **getName** ()

## Additional Inherited Members

### 4.186.1 Detailed Description

Class that represents a storage plugin.

The purpose of this class is to hide the use of the pointers into the dynamically loaded plugin and wrap the interface into a class that can be used directly in the storage subsystem.

This is achieved by having a set of private member variables which are the pointers to the functions in the plugin, and a set of public methods that will call these functions via the function pointers.

### 4.186.2 Constructor & Destructor Documentation

#### 4.186.2.1 StoragePlugin()

```
StoragePlugin::StoragePlugin (
            const std::string & name,
            PLUGIN_HANDLE handle )
```

Constructor for the class that wraps the storage plugin.

Create a set of function points that resolve to the loaded plugin and enclose in the class.

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

- C/services/storage/include/storage_plugin.h
- C/services/storage/storage_plugin.cpp

## 4.187 StoragePluginConfiguration Class Reference

The storage service must handle its own configuration differently to other services as it is unable to read the configuration from the database.

```
#include <plugin_configuration.h>
```

### Public Member Functions

- StoragePluginConfiguration (const std::string &name, StoragePlugin *plugin)

  *Constructor for storage service configuration class.*
- const char * getValue (const std::string &key)

  *Return a value from the cached configuration category.*
- bool hasValue (const std::string &key)

  *Return if a value exsits for the cached configuration category.*
- bool setValue (const std::string &key, const std::string &value)

  *Set the value of a configuration item.*
- void updateCategory (const std::string &json)

  *Called when the configuration category is updated.*
- void **registerCategory** (ManagementClient *client)
- DefaultConfigCategory * getDefaultCategory ()

  *Return the default category to register with the core.*
- ConfigCategory * getConfiguration ()

  *Return the category to register with the core.*

### 4.187.1 Detailed Description

The storage service must handle its own configuration differently to other services as it is unable to read the configuration from the database.

This class deals with the configuration from the storage plugins, maintaining a cache for the plugin

### 4.187.2 Member Function Documentation

#### 4.187.2.1 getConfiguration()

```
ConfigCategory * StoragePluginConfiguration::getConfiguration ( )
```

Return the category to register with the core.

This allows the storage configuration to appear in the UI

**Returns**

ConfigCategory* The default configuration category

#### 4.187.2.2 getDefaultCategory()

```
DefaultConfigCategory * StoragePluginConfiguration::getDefaultCategory ( )
```

Return the default category to register with the core.

This allows the storage configuration to appear in the UI

**Returns**

DefaultConfigCategory∗ The default configuration category

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

- C/services/storage/include/plugin_configuration.h
- C/services/storage/pluginconfiguration.cpp

## 4.188 StorageRegistry Class Reference

StorageRegistry - a class that manages requests from other microservices to register interest in new readings being inserted into the storage layer that match a given asset code, or any asset code "∗".

```
#include <storage_registry.h>
```

### Public Member Functions

- StorageRegistry ()

  *StorageRegistry constructor.*
- ∼StorageRegistry ()

  *StorageRegistry destructor.*
- void registerAsset (const std::string &asset, const std::string &url)

  *Handle a registration request from a client of the storage layer.*
- void unregisterAsset (const std::string &asset, const std::string &url)

  *Handle a request to remove a registration of interest.*
- void process (const std::string &payload)

  *Process a reading append payload and determine if any microservice has registered an interest in this asset.*
- void processTableInsert (const std::string &tableName, const std::string &payload)

  *Process a table insert payload and determine if any microservice has registered an interest in this table.*
- void processTableUpdate (const std::string &tableName, const std::string &payload)

  *Process a table update payload and determine if any microservice has registered an interest in this table.*
- void processTableDelete (const std::string &tableName, const std::string &payload)

  *Process a table delete payload and determine if any microservice has registered an interest in this table.*
- void registerTable (const std::string &table, const std::string &url)

  *Handle a registration request for a table from a client of the storage layer.*
- void unregisterTable (const std::string &table, const std::string &url)

  *Handle a request to remove a registration of interest in a table.*
- void run ()

  *The worker function that processes the queue of payloads that may need to be sent to subscribers.*

### 4.188.1 Detailed Description

StorageRegistry - a class that manages requests from other microservices to register interest in new readings being inserted into the storage layer that match a given asset code, or any asset code "∗".

### 4.188.2 Constructor & Destructor Documentation

#### 4.188.2.1 StorageRegistry()

```
StorageRegistry::StorageRegistry ( )
```

StorageRegistry constructor.

The storage registry holds registrations for other micro services that wish to receive notifications when new data is avialable for a given asset. The interested service registers a URL and an asset code, or ∗ for all assets, that URL will then be called when new data arrives for the particular asset.

The service registry maintians a worker thread that is responsible for sending these notifications such that the main flow of data into the storage layer is minimally impacted by the registration and delivery of these messages to interested microservices.

### 4.188.3 Member Function Documentation

#### 4.188.3.1 process()

```
void StorageRegistry::process (
            const std::string & payload )
```

Process a reading append payload and determine if any microservice has registered an interest in this asset.

**Parameters**

| | |
|---|---|
| *payload* | The reading append payload |

#### 4.188.3.2 processTableDelete()

```
void StorageRegistry::processTableDelete (
            const std::string & tableName,
            const std::string & payload )
```

Process a table delete payload and determine if any microservice has registered an interest in this table.

Called from StorageApi::commonDelete()

**Parameters**

| | |
|---|---|
| *payload* | The table delete payload |

#### 4.188.3.3 processTableInsert()

```
void StorageRegistry::processTableInsert (
            const std::string & tableName,
            const std::string & payload )
```

Process a table insert payload and determine if any microservice has registered an interest in this table.

Called from StorageApi::commonInsert()

**Parameters**

| | |
|---|---|
| *payload* | The table insert payload |

#### 4.188.3.4 processTableUpdate()

```
void StorageRegistry::processTableUpdate (
            const std::string & tableName,
            const std::string & payload )
```

Process a table update payload and determine if any microservice has registered an interest in this table.

Called from StorageApi::commonUpdate()

**Parameters**

| | |
|---|---|
| *payload* | The table update payload |

#### 4.188.3.5 registerAsset()

```
void StorageRegistry::registerAsset (
            const std::string & asset,
            const std::string & url )
```

Handle a registration request from a client of the storage layer.

**Parameters**

| | |
|---|---|
| *asset* | The asset of interest |
| *url* | The URL to call |

#### 4.188.3.6 registerTable()

```
void StorageRegistry::registerTable (
            const std::string & table,
            const std::string & url )
```

Handle a registration request for a table from a client of the storage layer.

**Parameters**

| | |
|---|---|
| *table* | The table of interest |
| *payload* | JSON payload describing the interest |

#### 4.188.3.7 unregisterAsset()

```
void StorageRegistry::unregisterAsset (
            const std::string & asset,
            const std::string & url )
```

Handle a request to remove a registration of interest.

**Parameters**

| | |
|---|---|
| *asset* | The asset of interest |
| *url* | The URL to call |

#### 4.188.3.8 unregisterTable()

```
void StorageRegistry::unregisterTable (
            const std::string & table,
            const std::string & url )
```

Handle a request to remove a registration of interest in a table.

**Parameters**

| | |
|---|---|
| *table* | The table of interest |
| *payload* | JSON payload describing the interest |

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

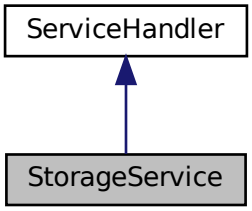
- C/services/storage/include/storage_registry.h
- C/services/storage/storage_registry.cpp

## 4.189 StorageService Class Reference

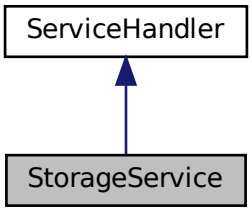
The StorageService class.

```
#include <storage_service.h>
```

Inheritance diagram for StorageService:




Collaboration diagram for StorageService:




### Public Member Functions

- StorageService (const string &name)

  *Constructor for the storage service.*
- ~StorageService ()

  *Storage Service destructor.*
- void start (std::string &coreAddress, unsigned short corePort)

  *Start the storage service.*
- void stop ()

  *Stop the storage service/.*
- void shutdown ()

  *Shutdown request.*
- void restart ()

*Restart request.*

- bool **isRunning** ()
- void configChange (const std::string &, const std::string &)

  *Configuration change notification.*
- void **configChildCreate** (const std::string &, const std::string &, const std::string &)
- void **configChildDelete** (const std::string &, const std::string &)
- string getPluginName ()

  *Return the name of the configured storage service.*
- string getPluginManagedStatus ()

  *Return the managed status of the storage plugin.*
- string getReadingPluginName ()

  *Return the name of the configured reading plugin.*
- void **setLogLevel** (std::string level)

### 4.189.1 Detailed Description

The StorageService class.

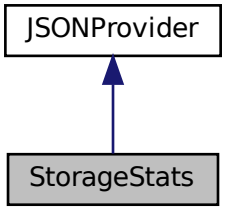
This class is the core of the service that offers access to the Fledge storage layer. It maintains the API and provides the hooks for incoming management API requests.

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

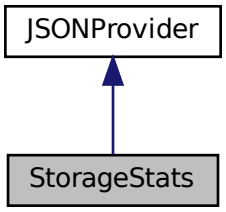
- C/services/storage/include/storage_service.h
- C/services/storage/storage.cpp

## 4.190 StorageStats Class Reference

Inheritance diagram for StorageStats:



Collaboration diagram for StorageStats:




### Public Member Functions

- StorageStats ()

  *Construct the statistics class for the storage service.*
- void asJSON (std::string &) const

  *Serialise the statistics as JSON.*

### Public Attributes

- unsigned int **commonInsert**
- unsigned int **commonSimpleQuery**
- unsigned int **commonQuery**
- unsigned int **commonUpdate**
- unsigned int **commonDelete**
- unsigned int **readingAppend**
- unsigned int **readingFetch**
- unsigned int **readingQuery**
- unsigned int **readingPurge**

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

- C/services/storage/include/storage_stats.h
- C/services/storage/storage_stats.cpp

## 4.191 StreamHandler Class Reference

### Public Member Functions

- StreamHandler (StorageApi *)

  *Constructor for the StreamHandler class.*
- ~StreamHandler ()

  *Destructor for the StreamHandler.*
- void handler ()

  *The handler method for the stream handler.*
- uint32_t createStream (uint32_t *token)

  *Create a new stream and add it to the epoll mechanism for the stream handler.*

### 4.191.1 Constructor & Destructor Documentation

#### 4.191.1.1 ~StreamHandler()

```
StreamHandler::~StreamHandler ( )
```

Destructor for the StreamHandler.

Close down the epoll system and wait for the handler thread to terminate.

### 4.191.2 Member Function Documentation

#### 4.191.2.1 createStream()

```
uint32_t StreamHandler::createStream (
            uint32_t * token )
```

Create a new stream and add it to the epoll mechanism for the stream handler.

**Parameters**

| | |
|---|---|
| *token* | The single use connection token the client should send |
| *The* | port on which this stream is listening |

#### 4.191.2.2 handler()

```
void StreamHandler::handler ( )
```

The handler method for the stream handler.

This is run in its own thread and is responsible for using epoll to gather events on the descriptors and to dispatch them to the individual streams

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

- C/services/storage/include/stream_handler.h
- C/services/storage/stream_handler.cpp

## 4.192 TableReference Class Reference

Class used to store table references.

```
#include <readings_catalogue.h>
```

### Public Member Functions

- **TableReference** (int dbId, int tableId)
- time_t **lastIssued** ()
- int **getTable** ()
- int **getDatabase** ()
- void **issue** ()

### 4.192.1 Detailed Description

Class used to store table references.

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

- C/plugins/storage/sqlite/common/include/readings_catalogue.h

## 4.193 TableRegistration Struct Reference

### Public Attributes

- std::string **url**
- std::string **key**
- std::vector< std::string > **keyValues**
- std::string **operation**

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

- C/services/storage/include/storage_registry.h

## 4.194 Timebucket Class Reference

Timebucket clause in a selection of records.

```
#include <timebucket.h>
```

### Public Member Functions

- **Timebucket** (const std::string &column, unsigned int size, const std::string &format, const std::string &alias)
- **Timebucket** (const std::string &column, unsigned int size, const std::string &format)
- std::string **toJSON** ()

### 4.194.1 Detailed Description

Timebucket clause in a selection of records.

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

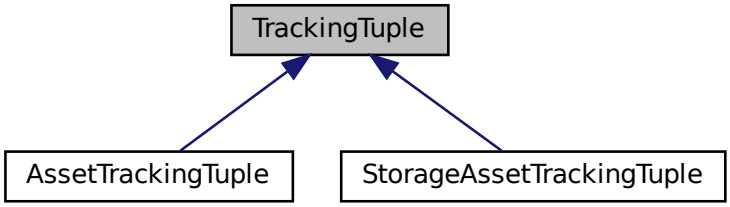
- C/common/include/timebucket.h

## 4.195 TrackingTuple Class Reference

Tracking abstract base class to be passed in the process data queue.

```
#include <asset_tracking.h>
```

Inheritance diagram for TrackingTuple:




### Public Member Functions

- virtual InsertValues **processData** (bool storage_connected, ManagementClient *mgtClient, bool &warned, std::string &instanceName)=0
- virtual std::string **assetToString** ()=0

### 4.195.1 Detailed Description

Tracking abstract base class to be passed in the process data queue.

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

- C/common/include/asset_tracking.h

## 4.196 TransactionBoundary Class Reference

This class handles per thread started transaction boundaries:

```
#include <readings_catalogue.h>
```

### Public Member Functions

- unsigned long GetMinReadingId ()

  *Fetch the minimum safe global reading id among all UNCOMMITTED per thread transactions.*
- void SetThreadTransactionStart (std::thread::id tid, unsigned long id)

  *Set BEGIN of a transaction for a given thread, reading id.*
- void ClearThreadTransaction (std::thread::id)

  *Remove committed TRANSACTION for the given thread.*

### 4.196.1 Detailed Description

This class handles per thread started transaction boundaries:

### 4.196.2 Member Function Documentation

#### 4.196.2.1 ClearThreadTransaction()

```
void TransactionBoundary::ClearThreadTransaction (
            std::thread::id tid )
```

Remove committed TRANSACTION for the given thread.

**Parameters**

| | |
|---|---|
| *tid* | The thread id that just committed a transaction |

#### 4.196.2.2 GetMinReadingId()

```
unsigned long TransactionBoundary::GetMinReadingId ( )
```

Fetch the minimum safe global reading id among all UNCOMMITTED per thread transactions.

**Returns**

The safe global reading id to use in UNION ALL queries as boundary limit

#### 4.196.2.3 SetThreadTransactionStart()

```
void TransactionBoundary::SetThreadTransactionStart (
            std::thread::id tid,
            unsigned long id )
```

Set BEGIN of a transaction for a given thread, reading id.

**Parameters**

| | |
|---|---|
| *tid* | The thread id that just started a transaction |
| *id* | The global reading id that starts the transaction |

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

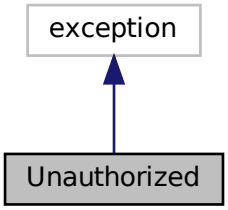
- C/plugins/storage/sqlite/common/include/readings_catalogue.h
- C/plugins/storage/sqlite/common/readings_catalogue.cpp

## 4.197 Unauthorized Class Reference

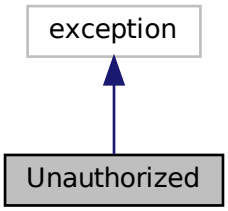
Unauthorized exception.

```
#include <http_sender.h>
```

Inheritance diagram for Unauthorized:




Collaboration diagram for Unauthorized:




### Public Member Functions

- **Unauthorized** (const std::string &serverReply)
- virtual const char ∗ **what** () const throw ()

### 4.197.1 Detailed Description

Unauthorized exception.

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

- C/plugins/common/include/http_sender.h

## 4.198 UpdateModifier Class Reference

Update modifier.

```
#include <update_modifier.h>
```

### Public Member Functions

- **UpdateModifier** (const std::string &modifier)
- const std::string **toJSON** () const

### 4.198.1 Detailed Description

Update modifier.

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

- C/common/include/update_modifier.h

## 4.199 UpdateValue Class Reference

A value in an update statement.

```
#include <value.h>
```

### Public Types

- enum **UpdateType** { **StringType**, **IntType**, **DoubleType**, **JSONType** }

### Public Member Functions

- **UpdateValue** (const std::string &column, const std::string &value)
- **UpdateValue** (const std::string &column, const int value)
- **UpdateValue** (const std::string &column, const double value)
- std::string **toJSON** ()

### 4.199.1 Detailed Description

A value in an update statement.

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

- C/common/include/value.h

## 4.200 ACL::UrlItem Class Reference

### Public Member Functions

- **UrlItem** (const std::string &url, const std::vector< KeyValueItem > &acl)

### Public Attributes

- std::string **url**
- std::vector< KeyValueItem > **acl**

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

- C/common/include/acl.h

## 4.201 Where Class Reference

Where clause in a selection of records.

```
#include <where.h>
```

### Public Member Functions

- **Where** (const std::string &column, const Condition condition, const std::string &value)
- **Where** (const std::string &column, const Condition condition, const std::string &value, Where *andCondition)
- **Where** (const std::string &column, const Condition condition)
- **Where** (const std::string &column, const Condition condition, Where *andCondition)
- ~Where ()

  *Where clause destructor.*
- void **andWhere** (Where *condition)
- void **orWhere** (Where *condition)
- void **addIn** (const std::string &value)
- const std::string toJSON () const

  *Return the JSON payload for a where clause.*

### 4.201.1 Detailed Description

Where clause in a selection of records.

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

- C/common/include/where.h
- C/common/where.cpp

# Chapter 5

# Example Documentation

## 5.1 /home/ubuntu/fledge/C/common/include/utils.h

Converts a string representation of a boolean value to a boolean type.This function takes a string input and checks if it represents a boolean value. It recognizes "true", "1", and their case-insensitive variants as true. Any other string will be interpreted as false.

**Parameters**

| | |
|---|---|
| *str* | The string to convert to a boolean. Can be "true", "false", "1", "0", etc. |

**Returns**

true if the input string represents a true value; false otherwise.

**Note**

This function is case-insensitive and will convert the input string to lowercase before comparison.

bool result1 = stringToBool("True"); // result1 is true bool result2 = stringToBool("false"); // result2 is false bool result3 = stringToBool("1"); // result3 is true bool result4 = stringToBool("0"); // result4 is false

```
#ifndef _FLEDGE_UTILS_H
#define _FLEDGE_UTILS_H
/*
 * Fledge general utilities
 *
 * Copyright (c) 2018 Dianomic Systems
 *
 * Released under the Apache 2.0 Licence
 *
 * Author: Massimiliano Pinto
 */
#include <string>
#include <algorithm>
#define _FLEDGE_ROOT_PATH    "/usr/local/fledge"
using namespace std;
static const string getRootDir()
{
    const char* rootDir = getenv("FLEDGE_ROOT");
    return (rootDir ? string(rootDir) : string(_FLEDGE_ROOT_PATH));
}
static const string getDataDir()
{
    const char* dataDir = getenv("FLEDGE_DATA");
    return (dataDir ? string(dataDir) : string(getRootDir() + "/data"));
```

```
}
static std::string getDebugTracePath()
{
    return getDataDir() + "/logs/debug-trace";
}
static bool stringToBool(const std::string& str)
{
    std::string lowerStr = str;
    std::transform(lowerStr.begin(), lowerStr.end(), lowerStr.begin(), ::tolower);
    return (lowerStr == "true" || lowerStr == "1");
}
#endif
```

# Index


~AssetTracker
    AssetTracker, 22
~Connection
    Connection, 87
~DataLoad
    DataLoad, 105
~DatapointValue
    DatapointValue, 111
~DefaultConfigCategory
    DefaultConfigCategory, 115
~Ingest
    Ingest, 143
~JSONPath
    JSONPath, 154
~NorthApi
    NorthApi, 189
~PipelineBranch
    PipelineBranch, 258
~PythonPluginHandle
    PythonPluginHandle, 286
~SouthApi
    SouthApi, 367
~StorageApi
    StorageApi, 389
~StreamHandler
    StreamHandler, 421

ACL, 15
ACL::ACLReason, 17
    ACLReason, 17
ACL::KeyValueItem, 158
ACL::UrlItem, 427
ACLMalformed, 16
ACLReason
    ACL::ACLReason, 17
ACLReasonMalformed, 18
add
    AssetTrackingTable, 26
addAssetTrackingTuple
    AssetTracker, 22
    ManagementClient, 174
addAuditEntry
    ManagementClient, 175
addCategory
    ManagementClient, 175
addCategoryDescription
    ConfigCategories, 44
addChildCategories
    ManagementClient, 176
addChildCategory
    ConfigurationManager, 76
    CoreManagementApi, 99
addProxy
    ManagementClient, 176
addStorageAssetTrackingTuple
    AssetTracker, 22
    ManagementClient, 177
addValue
    PerfMon, 253
Aggregate, 19
aggregate
    Query, 296
alertFailures
    NorthService, 201
AllCategoriesEx, 19
allocate
    ConnectionManager, 94
allowMoreDatabases
    ConnectionManager, 94
append
    OMFBuffer, 225, 226
    ReadingSet, 326, 327
    SQLBuffer, 382, 383
ApplyPIServerNamingRulesInvalidChars
    OMF, 212
ApplyPIServerNamingRulesObj
    OMF, 213
ApplyPIServerNamingRulesPath
    OMF, 213
AssetTracker, 20
    ~AssetTracker, 22
    addAssetTrackingTuple, 22
    addStorageAssetTrackingTuple, 22
    AssetTracker, 21
    checkAssetTrackingCache, 23
    findAssetTrackingCache, 23
    findStorageAssetTrackingCache, 23
    getAssetTracker, 24
    getDeprecated, 24
    getStorageAssetTrackingCacheData, 24
    populateAssetTrackingCache, 25
    populateStorageAssetTrackingCache, 25
    tune, 25
    updateCache, 25
AssetTrackingTable, 26
    add, 26
    find, 27
AssetTrackingTuple, 27
    processData, 28

AssetTrackingTuplePtrEqual, 29
attach
Schema, 343
attachDb
ReadingsCatalogue, 312
attachDbsToAllConnections
ReadingsCatalogue, 313
AttachDbSync, 29
attachDebugger
FilterPipeline, 124
NorthApi, 189
PipelineElement, 264
SouthApi, 367
attachNewDb
ConnectionManager, 94
attachRequestNewDb
ConnectionManager, 95
audit
AuditLogger, 31
AuditLogger, 30
audit, 31
AuditLogger, 30
AuthenticationMiddlewareACL
ServiceAuthHandler, 353
AuthenticationMiddlewareCommon
ServiceAuthHandler, 354
AuthenticationMiddlewarePUT
ServiceAuthHandler, 354

background
ConnectionManager, 95
BadRequest, 31
Base64DataBuffer, 32
encode, 33
Base64DPImage, 34
encode, 35
BearerToken, 35
BearerToken, 35
verify, 36
BinaryPluginHandle, 36
buildLookup
OMFLinkedData, 242

CategoryDetailsEx, 37
checkAssetTrackingCache
AssetTracker, 23
checkDefaultValuesOnly
ConfigCategory, 48
CheckUpdates, 40
ChildCategoriesEx, 41
cleanupFilters
FilterPipeline, 124
clearAlert
ManagementClient, 177
clearFailures
NorthService, 202
clearLALookup
OMFLinkedData, 242
ClearThreadTransaction
TransactionBoundary, 424
coalesce
OMFBuffer, 227
SQLBuffer, 384
collect
PerformanceMonitor, 256
columnName
ResultSet, 338
columnType
ResultSet, 338
commonDelete
StorageApi, 389
commonInsert
StorageApi, 389
commonQuery
StorageApi, 390
commonSimpleQuery
StorageApi, 390
commonUpdate
StorageApi, 390
compress_string
OMF, 214
ConfigCategories, 44
addCategoryDescription, 44
ConfigCategories, 44
ConfigCategory, 45
checkDefaultValuesOnly, 48
ConfigCategory, 48
extractSubcategory, 49
getBoolValue, 49
getDefault, 49
getItemAttribute, 50
getLength, 50
getMaximum, 51
getMinimum, 51
getOptions, 52
getPermissions, 52
getType, 53
getValue, 53
getValueKVList, 53
getValueList, 54
hasPermission, 54
isBool, 55
isDeprecated, 55
isDouble, 56
isEnumeration, 56
isJSON, 57
isKVList, 57
isList, 58
isNumber, 58
isString, 59
itemExists, 59
itemsToJSON, 59
itemToJSON, 60
keepItemsType, 60
parseBucketItemValue, 60
removeItems, 61
removeItemsType, 61

setDefault, 61
setDescription, 61
setItemAttribute, 62
setItemDisplayName, 62
setValue, 63
to_string, 63
toJSON, 63
ConfigCategory::CategoryItem, 38
defaultToJSON, 39
toJSON, 39
ConfigCategoryChange, 65
ConfigCategoryChange, 66
ConfigCategoryDefaultWithValue, 66
ConfigCategoryDescription, 67
ConfigCategoryEx, 68
configChange
ConfigHandler, 70
DataLoad, 105
DataSender, 113
FilterPipeline, 124
Ingest, 143
ManagementApi, 169
SouthService, 378
configChildCreate
ConfigHandler, 70
ManagementApi, 169
configChildDelete
ConfigHandler, 70
ManagementApi, 170
ConfigHandler, 69
configChange, 70
configChildCreate, 70
configChildDelete, 70
registerCategory, 71
registerCategoryChild, 71
unregisterCategory, 71
ConfigItemAttributeNotFound, 72
ConfigItemNotAList, 73
ConfigItemNotFound, 74
ConfigMalformed, 75
ConfigurationManager, 75
addChildCategory, 76
createCategory, 77
deleteCategory, 77
deleteCategoryItemValue, 79
deleteChildCategory, 79
getAllCategoryNames, 80
getCategoryAllItems, 80
getCategoryItem, 81
getCategoryItemValue, 81
getChildCategories, 82
getInstance, 82
setCategoryItemValue, 82
configureRateMonitor
Ingest, 144
ConfigValueFoundWithDefault, 83
Conflict, 84
Connection, 85
~Connection, 87
Connection, 87, 88
create_schema, 88
createSchema, 88
fetchReadings, 89
findSchemaFromDB, 89
formatDate, 89
parseDatabaseStorageSchema, 90
purgeReadingsByRows, 91
readingStream, 91
setTrace, 91
vacuum, 92
connectionAttachAllDbs
ReadingsCatalogue, 313
connectionAttachDbList
ReadingsCatalogue, 313
ConnectionManager, 92
allocate, 94
allowMoreDatabases, 94
attachNewDb, 94
attachRequestNewDb, 95
background, 95
detachNewDb, 95
getInstance, 95
growPool, 95
release, 96
setError, 96
setPurgeBlockSize, 96
shrinkPool, 97
shutdown, 97
containerSent
LALookup, 159
copy
ReadingSet, 327
CoreManagementApi, 97
addChildCategory, 99
createCategory, 99
getInstance, 100
registerMicroService, 100
create
SchemaManager, 345
create_schema
Connection, 88
createAssetAFH
OMF, 214
createCategory
ConfigurationManager, 77
CoreManagementApi, 99
createDebugTraceDirectory
HttpSender, 139
createNewStream
SendingProcess, 349
createReadingsOverflowTable
ReadingsCatalogue, 314
createSchema
Connection, 88
createSecurityCategories
ServiceAuthHandler, 355

createStatsDbEntry
Ingest, 144
createStorageSchema
StorageApi, 391
createStorageStream
StorageApi, 391
createStream
SendingProcess, 349
StreamHandler, 421

DataBuffer, 100
DataBuffer, 101, 102
populate, 102
DataLoad, 102
~DataLoad, 105
configChange, 105
DataLoad, 104
fetchReadings, 105
passToOnwardFilter, 106
pipelineEnd, 106
setDataSource, 107
triggerRead, 107
Datapoint, 107
parseJson, 108
recursiveJson, 108
DatapointValue, 109
~DatapointValue, 111
setValue, 111, 112
toString, 112
DataSender, 112
configChange, 113
isDryRun, 113
pause, 113
release, 113
debug
Logger, 164
debugState
NorthService, 202
SouthService, 378
DefaultConfigCategory, 114
~DefaultConfigCategory, 115
defaultToJSON
ConfigCategory::CategoryItem, 39
deleteAssetAFH
OMF, 214
deleteCategory
ConfigurationManager, 77
deleteCategoryItemValue
ConfigurationManager, 79
deleteChildCategory
ConfigurationManager, 79
deleteProxy
ManagementClient, 177
deleteTable
StorageClient, 399, 400
detachDb
ReadingsCatalogue, 314
detachDebugger
NorthApi, 189
SouthApi, 368
detachNewDb
ConnectionManager, 95
DiskSpaceMonitor, 115
DiskSpaceMonitor, 116
periodic, 116
DPImage, 117
DPImage, 118
operator=, 119

encode
Base64DataBuffer, 33
Base64DPImage, 35
error
Logger, 164
errorMessage
PythonReading, 289
errorMessageHandler
PIWebAPI, 270
escape
Reading, 303
evaluateGlobalId
ReadingsCatalogue, 314
exclude
PurgeConfiguration, 280
execute
FilterPipeline, 125
PythonRuntime, 292
ExistingChildCategories, 119
exists
SchemaManager, 346
Expression, 120
ExpressionValues, 121
extract
ReadingCircularBuffer, 307
ReadingSetCircularBuffer, 329
extractDbIdFromName
ReadingsCatalogue, 315
extractReadingsIdFromName
ReadingsCatalogue, 315
extractSubcategory
ConfigCategory, 49
extractToken
OCS, 209
extractVariable
OMF, 215
ExtractVersion
PIWebAPI, 271

fatal
Logger, 164
fetchBuffer
PipelineDebugger, 261
fetchReadings
Connection, 89
DataLoad, 105
FilterPipeline, 122
attachDebugger, 124
cleanupFilters, 124

configChange, 124
execute, 125
FilterPipeline, 123
getDebuggerBuffer, 125
loadFilters, 125
setDebuggerBuffer, 126
setupFiltersPipeline, 126
FilterPlugin, 127
FilterPlugin, 128
ingest, 128
init, 129
reconfigure, 129
shutdownSaveData, 129
startData, 130
FilterPythonPluginHandle, 130
FilterPythonPluginHandle, 131
find
AssetTrackingTable, 27
findAssetTrackingCache
AssetTracker, 23
findColumn
ResultSet, 339
findNode
JSONPath, 154
findSchemaFromDB
Connection, 89
findService
ServiceRegistry, 359
findStorageAssetTrackingCache
AssetTracker, 23
firstRow
ResultSet, 339
FledgeFilter, 132
FledgeFilter, 132
setConfig, 134
FledgeProcess, 134
getArgValue, 135
flowControl
Ingest, 144
flushContainers
OMFLinkedData, 243
formatDate
Connection, 89
FormData, 136
getUploadedData, 136
getUploadedFile, 137
saveFile, 137
FormData::FieldValue, 121

generateAFHierarchyPrefixLevel
OMF, 215
generateDbAlias
ReadingsCatalogue, 315
generateDbFileName
ReadingsCatalogue, 316
generateDbName
ReadingsCatalogue, 316
generateDbNameFromTableId
ReadingsCatalogue, 316
generateMeasurementId
OMF, 215
generateReadingsName
ReadingsCatalogue, 317
generateSuffixType
OMF, 216
getACL
ManagementClient, 178
getAlertByKey
ManagementClient, 178
getAllCategoryNames
ConfigurationManager, 80
getAllDbs
ReadingsCatalogue, 317
getArgValue
FledgeProcess, 135
getAssetDateTime
Reading, 303
getAssetDateUserTime
Reading, 303
getAssetTracker
AssetTracker, 24
getAssetTrackingTuple
ManagementClient, 179
getAssetTrackingTuples
ManagementClient, 179
getAuthenticatedCaller
ServiceAuthHandler, 355
getBoolValue
ConfigCategory, 49
getCategories
ManagementClient, 179
getCategory
ManagementClient, 180
getCategoryAllItems
ConfigurationManager, 80
getCategoryItem
ConfigurationManager, 81
getCategoryItemValue
ConfigurationManager, 81
getChildCategories
ConfigurationManager, 82
ManagementClient, 180
getColumn
ResultSet::Row, 341
getConfiguration
StoragePluginConfiguration, 412
getDatapoint
Reading, 304
getDebuggerBuffer
FilterPipeline, 125
NorthApi, 190
PipelineElement, 264
SouthApi, 368
getDefault
ConfigCategory, 49
getDefaultCategory
StorageConfiguration, 406

StoragePluginConfiguration, 412
getDeprecated
AssetTracker, 24
getDeprecatedAssetTrackingTuples
ManagementClient, 181
getEmptyReadingTableReference
ReadingsCatalogue, 318
getEventExceptionMessage
OMFError, 230
getEventExceptionType
OMFError, 231
getEventReason
OMFError, 231
getEventSeverity
OMFError, 231
getFormattedDateTimeStr
Reading, 304
getFormatType
OMF, 216
getHashStored
OMF, 217
getHintForChecksum
OMFHints, 236
getHttpClient
ManagementClient, 181
getHttpCode
OMFError, 232
getInstalledPlugins
PluginManager, 277
getInstance
ConfigurationManager, 82
ConnectionManager, 95
CoreManagementApi, 100
ManagementApi, 170
PurgeConfiguration, 280
SchemaManager, 346
getInteger
ResultSet::ColumnValue, 42
getItemAttribute
ConfigCategory, 50
getLastSentReadingId
SendingProcess, 349
getLength
ConfigCategory, 50
getLoadBufferIndex
SendingProcess, 350
getLoadBufferIndexPtr
SendingProcess, 350
getMaximum
ConfigCategory, 51
getMaxReadingsId
ReadingsCatalogue, 318
getMessage
OMFError, 232
getMinGlobalId
ReadingsCatalogue, 318
getMinimum
ConfigCategory, 51
GetMinReadingId
TransactionBoundary, 424
getNamingScheme
OMF, 217
getNewDbs
ReadingsCatalogue, 319
getNumber
ResultSet::ColumnValue, 43
getOMFTracePath
HttpSender, 140
getOptions
ConfigCategory, 52
getPathOrigStored
OMF, 217
getPathStored
OMF, 218
getPermissions
ConfigCategory, 52
getPluginsByFlags
PluginManager, 278
getPythonRuntime
PythonRuntime, 292
getReadingId
ReadingSet, 327
getReadingPosition
ReadingsCatalogue, 319
getReadingReference
ReadingsCatalogue, 319
getReadingsCount
ReadingsCatalogue, 320
getService
ManagementClient, 181
getServices
ManagementClient, 182
getStorageAssetTrackingCacheData
AssetTracker, 24
getStorageAssetTrackingTuple
ManagementClient, 182
getStorageAssetTrackingTuples
ManagementClient, 182
getString
ResultSet::ColumnValue, 43
getType
ConfigCategory, 53
ResultSet::Row, 342
getUploadedData
FormData, 136
getUploadedFile
FormData, 137
getUUID
ServiceRegistry, 360
getValue
ConfigCategory, 53
getValueKVList
ConfigCategory, 53
getValueList
ConfigCategory, 54
getValues

PerfMon, 255
GetVersion
PIWebAPI, 271
group
Query, 297
growPool
ConnectionManager, 95

handler
StreamHandler, 421
hasNextRow
ResultSet, 339
hasPermission
ConfigCategory, 54
HttpSender, 138
createDebugTraceDirectory, 139
getOMFTracePath, 140
HttpSender, 139

importModule
PythonRuntime, 292
info
Logger, 165
Ingest, 140
~Ingest, 143
configChange, 143
configureRateMonitor, 144
createStatsDbEntry, 144
flowControl, 144
Ingest, 142
ingest, 144, 145
loadFilters, 145
passToOnwardFilter, 145
processQueue, 146
running, 146
setResourceLimit, 146
setStatistics, 147
start, 147
unDeprecateAssetTrackingRecord, 147
unDeprecateStorageAssetTrackingRecord, 148
updateStats, 148
useFilteredData, 148
ingest
FilterPlugin, 128
Ingest, 144, 145
IngestRate, 150
PipelineBranch, 258
IngestRate, 149
ingest, 150
IngestRate, 150
periodic, 150
updateConfig, 150
init
FilterPlugin, 129
NorthPlugin, 197
PipelineBranch, 259
PipelineFilter, 266
insert
ReadingCircularBuffer, 308
ReadingSetCircularBuffer, 329, 331
InsertValue, 151
InsertValues, 152
isArray
PythonReading, 289
isBool
ConfigCategory, 55
isDataTypeError
OMF, 218
isDeprecated
ConfigCategory, 55
isDouble
ConfigCategory, 56
isDryRun
DataSender, 113
isEnumeration
ConfigCategory, 56
isExcluded
PurgeConfiguration, 280
isJSON
ConfigCategory, 57
isKVList
ConfigCategory, 57
isLastRow
ResultSet, 340
isList
ConfigCategory, 58
isNumber
ConfigCategory, 58
isolateDebugger
NorthApi, 190
SouthApi, 368
isString
ConfigCategory, 59
itemExists
ConfigCategory, 59
itemsToJSON
ConfigCategory, 59
itemToJSON
ConfigCategory, 60

Join, 152
toJSON, 153
join
Query, 297
JSONPath, 153
~JSONPath, 154
findNode, 154
JSONProperties, 155
JSONProperty, 156
JSONProvider, 156
JSONReading, 156
JSONReading, 157
JSONtoDatapoints
Reading, 305

keepItemsType
ConfigCategory, 60

LALookup, 158
containerSent, 159
latestDbUpdate
ReadingsCatalogue, 320
LibcurlHttps, 160
sendRequest, 161
setProxy, 162
limit
Query, 297
load
SchemaManager, 346
loadAssetReadingCatalogue
ReadingsCatalogue, 320
loadEmptyAssetReadingCatalogue
ReadingsCatalogue, 320
loadFilters
FilterPipeline, 125
Ingest, 145
loadStoredData
PluginData, 275
Log
OMFError, 232
Logger, 162
debug, 164
error, 164
fatal, 164
info, 165
Logger, 163
printLongString, 165
registerInterceptor, 165
setMinLevel, 166
unregisterInterceptor, 166
warn, 166

m_dateTypes
Reading, 306
ManagementApi, 167
configChange, 169
configChildCreate, 169
configChildDelete, 170
getInstance, 170
ManagementApi, 169
ping, 170
registerProvider, 171
registerStats, 171
securityChange, 171
shutdown, 171
ManagementClient, 172
addAssetTrackingTuple, 174
addAuditEntry, 175
addCategory, 175
addChildCategories, 176
addProxy, 176
addStorageAssetTrackingTuple, 177
clearAlert, 177
deleteProxy, 177
getACL, 178
getAlertByKey, 178
getAssetTrackingTuple, 179
getAssetTrackingTuples, 179
getCategories, 179
getCategory, 180
getChildCategories, 180
getDeprecatedAssetTrackingTuples, 181
getHttpClient, 181
getService, 181
getServices, 182
getStorageAssetTrackingTuple, 182
getStorageAssetTrackingTuples, 182
ManagementClient, 174
raiseAlert, 183
refreshBearerToken, 183
registerCategory, 184
registerCategoryChild, 184
registerService, 184
restartService, 185
setCategoryItemValue, 185
unregisterCategory, 185
unregisterService, 186
validateDatapoints, 186
verifyAccessBearerToken, 186
verifyBearerToken, 187
MemConnectionManager, 187
merge
ReadingSet, 328
minimumRetained
PurgeConfiguration, 281
multipleReadingsInit
ReadingsCatalogue, 321

nextRow
ResultSet, 340
NorthApi, 188
~NorthApi, 189
attachDebugger, 189
detachDebugger, 189
getDebuggerBuffer, 190
isolateDebugger, 190
NorthApi, 189
replayDebugger, 190
setDebuggerBuffer, 191
startServer, 191
stateDebugger, 191
stepDebugger, 191
suspendDebugger, 192
NorthFilterPipeline, 192
NorthFilterPipeline, 193
setupFiltersPipeline, 194
NorthPlugin, 194
init, 197
NorthPlugin, 196
send, 197
NorthPythonPluginHandle, 198
NorthPythonPluginHandle, 199
NorthService, 199
alertFailures, 201
clearFailures, 202
debugState, 202

NorthServiceProvider, 202
NoSuchCategory, 203
NoSuchCategoryItem, 204
NoSuchCategoryItemValue, 205
NotificationPythonPluginHandle, 206
    NotificationPythonPluginHandle, 207
NotSupportedDataType, 208

OCS, 209
    extractToken, 209
    OCSRetrieveAuthToken, 209
    retrieveToken, 210
OCSRetrieveAuthToken
    OCS, 209
OMF, 210
    ApplyPIServerNamingRulesInvalidChars, 212
    ApplyPIServerNamingRulesObj, 213
    ApplyPIServerNamingRulesPath, 213
    compress_string, 214
    createAssetAFH, 214
    deleteAssetAFH, 214
    extractVariable, 215
    generateAFHierarchyPrefixLevel, 215
    generateMeasurementId, 215
    generateSuffixType, 216
    getFormatType, 216
    getHashStored, 217
    getNamingScheme, 217
    getPathOrigStored, 217
    getPathStored, 218
    isDataTypeError, 218
    reportAsset, 218
    sendToServer, 219, 220
    setFormatType, 221
    setMapObjectTypes, 221
    setPathStored, 221
    unsetMapObjectTypes, 222
    variableValueHandle, 222
OMFAFLocationHint, 223
OMFBuffer, 224
    append, 225, 226
    coalesce, 227
    OMFBuffer, 224
    quote, 227
OMFData, 227
    OMFData, 228
OMFDataTypes, 228
OMFError, 229
    getEventExceptionMessage, 230
    getEventExceptionType, 231
    getEventReason, 231
    getEventSeverity, 231
    getHttpCode, 232
    getMessage, 232
    Log, 232
    setFromHttpResponse, 233
OMFHint, 233
OMFHints, 235
    getHintForChecksum, 236
    OMFHints, 235
OMFInformation, 236
    OMFInformation, 237
    saveData, 237
    send, 237
    start, 238
OMFIntegerHint, 238
OMFInterpolationHint, 239
OMFLegacyTypeHint, 240
OMFLinkedData, 241
    buildLookup, 242
    clearLALookup, 242
    flushContainers, 243
    processReading, 243
OMFMaximumHint, 244
OMFMinimumHint, 245
OMFNumberHint, 246
OMFSourceHint, 247
OMFTagHint, 248
OMFTagNameDatapointHint, 249
OMFTagNameHint, 250
OMFTypeNameHint, 251
OMFUOMHint, 252
operation
    SouthApi, 369
    SouthPlugin, 373
    SouthService, 379
operator=
    DPImage, 119

parseBucketItemValue
    ConfigCategory, 60
parseDatabaseStorageSchema
    Connection, 90
parseJson
    Datapoint, 108
passToOnwardFilter
    DataLoad, 106
    Ingest, 145
    SendingProcess, 350
pause
    DataSender, 113
PerfMon, 253
    addValue, 253
    getValues, 255
    PerfMon, 253
PerformanceMonitor, 255
    collect, 256
    PerformanceMonitor, 256
    setCollecting, 256
periodic
    DiskSpaceMonitor, 116
    IngestRate, 150
persistPluginData
    PluginData, 275
ping
    ManagementApi, 170
PipelineBranch, 257
    ~PipelineBranch, 258

ingest, 258
init, 259
setup, 259
setupConfiguration, 259
shutdown, 260
PipelineDebugger, 260
fetchBuffer, 261
process, 261
setBuffer, 262
PipelineElement, 262
attachDebugger, 264
getDebuggerBuffer, 264
setDebuggerBuffer, 264
pipelineEnd
DataLoad, 106
PipelineFilter, 265
init, 266
PipelineFilter, 266
setup, 267
setupConfiguration, 267
shutdown, 267
PipelineWriter, 268
PIWebAPI, 270
errorMessageHandler, 270
ExtractVersion, 271
GetVersion, 271
Plugin, 272
PLUGIN_ERROR, 273
PLUGIN_INFORMATION, 273
plugin_parameter, 274
PluginData, 274
loadStoredData, 275
persistPluginData, 275
PluginData, 274
PluginHandle, 276
PluginManager, 276
getInstalledPlugins, 277
getPluginsByFlags, 278
PluginNotImplementedException, 278
populate
DataBuffer, 102
populateAssetTrackingCache
AssetTracker, 25
populateStorageAssetTrackingCache
AssetTracker, 25
preallocateNewDbsRange
ReadingsCatalogue, 321
preallocateReadingsTables
ReadingsCatalogue, 321
printLongString
Logger, 165
process
PipelineDebugger, 261
StorageRegistry, 414
processConfigChange
SouthService, 379
processData
AssetTrackingTuple, 28
StorageAssetTrackingTuple, 396
processQueue
Ingest, 146
processReading
OMFLinkedData, 243
processTableDelete
StorageRegistry, 414
processTableInsert
StorageRegistry, 416
processTableUpdate
StorageRegistry, 416
ProfileItem, 279
purgeAllReadings
ReadingsCatalogue, 322
PurgeConfiguration, 280
exclude, 280
getInstance, 280
isExcluded, 280
minimumRetained, 281
purgeReadingsByRows
Connection, 91
PurgeResult, 281
PurgeSystem, 282
PythonConfigCategory, 283
PythonConfigCategory, 284
toPython, 284
PythonModule, 284
PythonPluginHandle, 285
~PythonPluginHandle, 286
ResolveSymbol, 287
PythonReading, 287
errorMessage, 289
isArray, 289
PythonReading, 288
toPython, 289
PythonReadingSet, 290
PythonReadingSet, 291
toPython, 291
PythonRuntime, 292
execute, 292
getPythonRuntime, 292
importModule, 292

Query, 293
aggregate, 296
group, 297
join, 297
limit, 297
Query, 294–296
returns, 298
sort, 298
timebucket, 298
QueryProfile, 299
queryTable
StorageClient, 400
queryTableToReadings
StorageClient, 401
queue
StorageApi, 391

quote
OMFBuffer, 227
SQLBuffer, 384

raiseAlert
ManagementClient, 183
RDSAcknowledge, 299
RDSBlockHeader, 300
RDSConnectHeader, 300
RDSReadingHeader, 300
Reading, 301
escape, 303
getAssetDateTime, 303
getAssetDateUserTime, 303
getDatapoint, 304
getFormattedDateTimeStr, 304
JSONtoDatapoints, 305
m_dateTypes, 306
removeDatapoint, 305
stringToTimestamp, 305
substitute, 306
readingAppend
StorageApi, 392
StorageClient, 401
ReadingCircularBuffer, 306
extract, 307
insert, 308
ReadingCircularBuffer, 307
readingFetch
StorageApi, 392
StorageClient, 401
readingPurge
StorageApi, 392
readingPurgeByAge
StorageClient, 402
readingPurgeByAsset
StorageClient, 402
readingPurgeBySize
StorageClient, 402
readingQuery
StorageApi, 393
StorageClient, 403
readingQueryToReadings
StorageClient, 403
ReadingsCatalogue, 309
attachDb, 312
attachDbsToAllConnections, 313
connectionAttachAllDbs, 313
connectionAttachDbList, 313
createReadingsOverflowTable, 314
detachDb, 314
evaluateGlobalId, 314
extractDbIdFromName, 315
extractReadingsIdFromName, 315
generateDbAlias, 315
generateDbFileName, 316
generateDbName, 316
generateDbNameFromTableId, 316
generateReadingsName, 317
getAllDbs, 317
getEmptyReadingTableReference, 318
getMaxReadingsId, 318
getMinGlobalId, 318
getNewDbs, 319
getReadingPosition, 319
getReadingReference, 319
getReadingsCount, 320
latestDbUpdate, 320
loadAssetReadingCatalogue, 320
loadEmptyAssetReadingCatalogue, 320
multipleReadingsInit, 321
preallocateNewDbsRange, 321
preallocateReadingsTables, 321
purgeAllReadings, 322
setUsedDbId, 322
sqlConstructMultiDb, 322
sqlConstructOverflow, 323
SQLExec, 323
storeGlobalId, 324
ReadingsCatalogue::ReadingReference, 309
ReadingSet, 324
append, 326, 327
copy, 327
getReadingId, 327
merge, 328
ReadingSet, 325, 326
ReadingSetCircularBuffer, 328
extract, 329
insert, 329, 331
ReadingSetCircularBuffer, 329
ReadingSetException, 331
ReadingStream, 332
readingStream
Connection, 91
StorageApi, 393
reconfigure
FilterPlugin, 129
recursiveJson
Datapoint, 108
refreshBearerToken
ManagementClient, 183
ServiceAuthHandler, 355
registerAsset
StorageRegistry, 416
registerAssetNotification
StorageClient, 404
registerCategory
ConfigHandler, 71
ManagementClient, 184
registerCategoryChild
ConfigHandler, 71
ManagementClient, 184
registerInterceptor
Logger, 165
registerMicroService
CoreManagementApi, 100
registerProvider

ManagementApi, 171
registerService
ManagementClient, 184
ServiceRegistry, 360
registerStats
ManagementApi, 171
registerTable
StorageRegistry, 417
registerTableNotification
StorageClient, 404
release
ConnectionManager, 96
DataSender, 113
removeDatapoint
Reading, 305
removeItems
ConfigCategory, 61
removeItemsType
ConfigCategory, 61
replayDebugger
NorthApi, 190
SouthApi, 369
reportAsset
OMF, 218
ResolveSymbol
PythonPluginHandle, 287
restartService
ManagementClient, 185
ResultException, 333
ResultIncorrectTypeException, 334
ResultNoMoreRowsException, 335
ResultNoSuchColumnException, 336
ResultSet, 336
columnName, 338
columnType, 338
findColumn, 339
firstRow, 339
hasNextRow, 339
isLastRow, 340
nextRow, 340
ResultSet, 337
ResultSet::ColumnValue, 42
getInteger, 42
getNumber, 43
getString, 43
ResultSet::Row, 341
getColumn, 341
getType, 342
retrieveToken
OCS, 210
Returns, 341
returns
Query, 298
running
Ingest, 146

saveData
OMFInformation, 237
saveFile
FormData, 137
Schema, 342
attach, 343
Schema, 343
upgrade, 344
SchemaManager, 344
create, 345
exists, 346
getInstance, 346
load, 346
securityChange
ManagementApi, 171
ServiceAuthHandler, 356
send
NorthPlugin, 197
OMFInformation, 237
SendingProcess, 347
createNewStream, 349
createStream, 349
getLastSentReadingId, 349
getLoadBufferIndex, 350
getLoadBufferIndexPtr, 350
passToOnwardFilter, 350
SendingProcess, 349
setLoadBufferIndex, 351
useFilteredData, 351
sendRequest
LibcurlHttps, 161
SimpleHttp, 362
SimpleHttps, 364
sendToServer
OMF, 219, 220
ServiceAuthHandler, 351
AuthenticationMiddlewareACL, 353
AuthenticationMiddlewareCommon, 354
AuthenticationMiddlewarePUT, 354
createSecurityCategories, 355
getAuthenticatedCaller, 355
refreshBearerToken, 355
securityChange, 356
setAuthenticatedCaller, 356
updateSecurityCategory, 356
ServiceHandler, 357
ServiceRecord, 358
ServiceRegistry, 359
findService, 359
getUUID, 360
registerService, 360
unRegisterService, 360
setAuthenticatedCaller
ServiceAuthHandler, 356
setBuffer
PipelineDebugger, 262
setCategoryItemValue
ConfigurationManager, 82
ManagementClient, 185
setCollecting
PerformanceMonitor, 256

setConfig
FledgeFilter, 134
setDataSource
DataLoad, 107
setDebuggerBuffer
FilterPipeline, 126
NorthApi, 191
PipelineElement, 264
SouthApi, 369
setDefault
ConfigCategory, 61
setDescription
ConfigCategory, 61
setError
ConnectionManager, 96
setFormatType
OMF, 221
setFromHttpResponse
OMFError, 233
setItemAttribute
ConfigCategory, 62
setItemDisplayName
ConfigCategory, 62
setLoadBufferIndex
SendingProcess, 351
setMapObjectTypes
OMF, 221
setMinLevel
Logger, 166
setPathStored
OMF, 221
setPoint
SouthApi, 369
SouthService, 379
setProxy
LibcurlHttps, 162
SimpleHttp, 363
SimpleHttps, 365
setPurgeBlockSize
ConnectionManager, 96
setResourceLimit
Ingest, 146
setStatistics
Ingest, 147
SetThreadTransactionStart
TransactionBoundary, 424
setTrace
Connection, 91
setup
PipelineBranch, 259
PipelineFilter, 267
setupConfiguration
PipelineBranch, 259
PipelineFilter, 267
setupFiltersPipeline
FilterPipeline, 126
NorthFilterPipeline, 194
setUsedDbId
ReadingsCatalogue, 322
setValue
ConfigCategory, 63
DatapointValue, 111, 112
shrinkPool
ConnectionManager, 97
shutdown
ConnectionManager, 97
ManagementApi, 171
PipelineBranch, 260
PipelineFilter, 267
shutdownSaveData
FilterPlugin, 129
SimpleHttp, 361
sendRequest, 362
setProxy, 363
SimpleHttps, 363
sendRequest, 364
setProxy, 365
Sort, 366
sort
Query, 298
SouthApi, 366
~SouthApi, 367
attachDebugger, 367
detachDebugger, 368
getDebuggerBuffer, 368
isolateDebugger, 368
operation, 369
replayDebugger, 369
setDebuggerBuffer, 369
setPoint, 369
SouthApi, 367
startServer, 370
stateDebugger, 370
stepDebugger, 370
suspendDebugger, 371
SouthPlugin, 371
operation, 373
SouthPlugin, 373
write, 373
SouthPythonPluginHandle, 374
SouthPythonPluginHandle, 375
SouthService, 376
configChange, 378
debugState, 378
operation, 379
processConfigChange, 379
setPoint, 379
SouthServiceProvider, 380
SQLBuffer, 381
append, 382, 383
coalesce, 384
quote, 384
SQLBuffer, 382
sqlConstructMultiDb
ReadingsCatalogue, 322
sqlConstructOverflow

ReadingsCatalogue, 323
SQLExec
ReadingsCatalogue, 323
start
Ingest, 147
OMFInformation, 238
startData
FilterPlugin, 130
startServer
NorthApi, 191
SouthApi, 370
stateDebugger
NorthApi, 191
SouthApi, 370
StatsHistory, 384
std::hash< AssetTrackingTuple >, 138
std::hash< AssetTrackingTuple * >, 137
std::hash< StorageAssetTrackingTuple >, 138
std::hash< StorageAssetTrackingTuple * >, 138
stepDebugger
NorthApi, 191
SouthApi, 370
STORAGE_CONFIGURATION, 386
StorageApi, 386
~StorageApi, 389
commonDelete, 389
commonInsert, 389
commonQuery, 390
commonSimpleQuery, 390
commonUpdate, 390
createStorageSchema, 391
createStorageStream, 391
queue, 391
readingAppend, 392
readingFetch, 392
readingPurge, 392
readingQuery, 393
readingStream, 393
storageTableDelete, 393
storageTableInsert, 394
storageTableQuery, 394
storageTableSimpleQuery, 394
storageTableUpdate, 395
StorageAssetTrackingTuple, 395
processData, 396
StorageAssetTrackingTuplePtrEqual, 397
StorageClient, 397
deleteTable, 399, 400
queryTable, 400
queryTableToReadings, 401
readingAppend, 401
readingFetch, 401
readingPurgeByAge, 402
readingPurgeByAsset, 402
readingPurgeBySize, 402
readingQuery, 403
readingQueryToReadings, 403
registerAssetNotification, 404
registerTableNotification, 404
unregisterAssetNotification, 404
unregisterTableNotification, 405
StorageConfiguration, 406
getDefaultCategory, 406
StorageConfiguration, 406
StorageOperation, 407
StoragePerformanceMonitor, 408
StoragePlugin, 409
StoragePlugin, 411
StoragePluginConfiguration, 412
getConfiguration, 412
getDefaultCategory, 412
StorageRegistry, 413
process, 414
processTableDelete, 414
processTableInsert, 416
processTableUpdate, 416
registerAsset, 416
registerTable, 417
StorageRegistry, 414
unregisterAsset, 417
unregisterTable, 417
StorageService, 418
StorageStats, 419
storageTableDelete
StorageApi, 393
storageTableInsert
StorageApi, 394
storageTableQuery
StorageApi, 394
storageTableSimpleQuery
StorageApi, 394
storageTableUpdate
StorageApi, 395
storeGlobalId
ReadingsCatalogue, 324
StreamHandler, 420
~StreamHandler, 421
createStream, 421
handler, 421
stringToTimestamp
Reading, 305
substitute
Reading, 306
suspendDebugger
NorthApi, 192
SouthApi, 371

TableReference, 421
TableRegistration, 422
Timebucket, 422
timebucket
Query, 298
to_string
ConfigCategory, 63
toJSON
ConfigCategory, 63
ConfigCategory::CategoryItem, 39

Join, 153
toPython
PythonConfigCategory, 284
PythonReading, 289
PythonReadingSet, 291
toString
DatapointValue, 112
TrackingTuple, 423
TransactionBoundary, 423
ClearThreadTransaction, 424
GetMinReadingId, 424
SetThreadTransactionStart, 424
triggerRead
DataLoad, 107
tune
AssetTracker, 25

Unauthorized, 425
unDeprecateAssetTrackingRecord
Ingest, 147
unDeprecateStorageAssetTrackingRecord
Ingest, 148
unregisterAsset
StorageRegistry, 417
unregisterAssetNotification
StorageClient, 404
unregisterCategory
ConfigHandler, 71
ManagementClient, 185
unregisterInterceptor
Logger, 166
unRegisterService
ServiceRegistry, 360
unregisterService
ManagementClient, 186
unregisterTable
StorageRegistry, 417
unregisterTableNotification
StorageClient, 405
unsetMapObjectTypes
OMF, 222
updateCache
AssetTracker, 25
updateConfig
IngestRate, 150
UpdateModifier, 426
updateSecurityCategory
ServiceAuthHandler, 356
updateStats
Ingest, 148
UpdateValue, 426
upgrade
Schema, 344
useFilteredData
Ingest, 148
SendingProcess, 351

vacuum
Connection, 92
validateDatapoints
ManagementClient, 186
variableValueHandle
OMF, 222
verify
BearerToken, 36
verifyAccessBearerToken
ManagementClient, 186
verifyBearerToken
ManagementClient, 187

warn
Logger, 166
Where, 427
write
SouthPlugin, 373