Passive-source Seismic-processing (PsSp)

0.1.0

Generated by Doxygen 1.9.8

Chapter 1

Passive-source Seismic-processing (PsSp)

1.1 PsSp

1.1.1 Doxygen documentation

Chapter 2

Todo List

File Constants.hpp

So far these are only related to SAC records and are used to prototype the interface. In the future, they'll be supplied by the sac-format library and not needed to be defined here.

File Enums.hpp

Non-enums (constants) belong in PsSp/Utility/Constants.hpp

Class pssp::Console_Sink< Mutex >

At present it doesn't do log formatting (formatting is handled with console codes in the logs themselves). Formatting should be moved to here in the future for generality.

Class pssp::ConsoleSink< Mutex >

At present it doesn't do log formatting (formatting is handled with console codes in the logs themselves). Formatting should be moved to here in the future for generality.

Member pssp::Field

This is for prototyping SAC-records, in the future this will be supplied by the sac-format library (once we're ready to read in SAC-files).

Class pssp::InputManager

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Namespace Documentation

6.1 pssp Namespace Reference

Namespaces

- namespace about
- namespace constants
- namespace datasheet
- namespace mw
- namespace structs
- namespace welcome

Classes

- class About_Window
- class Application

  Main application class.
- class Console_Sink

  Sink (receiver) of log messages for PsSp console.
- class ConsoleSink

  Sink (receiver) of log messages for PsSp console.
- class Datasheet
- class InputManager

  Manager of user-input.
- class Main_Window
- class SheetManager
- class StatusBar
- struct trace_info

  Information for.
- class Welcome_Window

Typedefs

- using Console_Sink_mt = Console_Sink< std::mutex >

  Multi-thread safe Console_Sink.
- using Console_Sink_st = Console_Sink< spdlog::details::null_mutex >

  Single-thread Console_Sink.
- using ConsoleSink_mt = ConsoleSink< std::mutex >

  Multi-thread safe Console_Sink.
- using ConsoleSink_st = ConsoleSink< spdlog::details::null_mutex >

  Single-thread Console_Sink.

Enumerations

- enum class Type {
  string_ , int_ , float_ , double_ ,
  bool_ }

  Data-type enumeration.
- enum class Field {
  depmin , depmax , odelta , resp0 ,
  resp1 , resp2 , resp3 , resp4 ,
  resp5 , resp6 , resp7 , resp8 ,
  resp9 , stel , stdp , evel ,
  evdp , mag , user0 , user1 ,
  user2 , user3 , user4 , user5 ,
  user6 , user7 , user8 , user9 ,
  dist , az , baz , gcarc ,
  depmen , cmpaz , cmpinc , xminimum ,
  xmaximum , yminimum , ymaximum , delta ,
  b , e , o , a ,
  t0 , t1 , t2 , t3 ,
  t4 , t5 , t6 , t7 ,
  t8 , t9 , f , stla ,
  stlo , evla , evlo , sb ,
  sdelta , nzyear , nzjday , nzhour ,
  nzmin , nzsec , nzmsec , nvhdr ,
  norid , nevid , npts , nsnpts ,
  nwfid , nxsize , nysize , iftype ,
  idep , iztype , iinst , istreg ,
  ievreg , ievtyp , iqual , isynth ,
  imagtyp , imagsrc , ibody , leven ,
  lpspol , lovrok , lcalda , kstnm ,
  kevnm , khole , ko , ka ,
  kt0 , kt1 , kt2 , kt3 ,
  kt4 , kt5 , kt6 , kt7 ,
  kt8 , kt9 , kf , kuser0 ,
  kuser1 , kuser2 , kcmpnm , knetwk ,
  kdatrd , kinst , data1 , data2 }

  SAC-header/footer field enumeration.

Variables

- const std::unordered_map< Type, const std::string > type_names

  Map Type to string-name.
- const std::unordered_map< size_t, Field > field_num
- const std::unordered_map< Field, trace_info > field_info

6.1.1 Typedef Documentation

6.1.1.1 Console_Sink_mt

```
using pssp::Console_Sink_mt = typedef Console_Sink<std::mutex>
```

Multi-thread safe Console_Sink.

6.1.1.2 Console_Sink_st

```
using pssp::Console_Sink_st = typedef Console_Sink<spdlog::details::null_mutex>
```

Single-thread Console_Sink.

6.1.1.3 ConsoleSink_mt

```
using pssp::ConsoleSink_mt = typedef ConsoleSink<std::mutex>
```

Multi-thread safe Console_Sink.

6.1.1.4 ConsoleSink_st

```
using pssp::ConsoleSink_st = typedef ConsoleSink<spdlog::details::null_mutex>
```

Single-thread Console_Sink.

6.1.2 Enumeration Type Documentation

6.1.2.1 Field

```
enum class pssp::Field  [strong]
```

SAC-header/footer field enumeration.

Todo This is for prototyping SAC-records, in the future this will be supplied by the sac-format library (once we're ready to read in SAC-files).

Enumerator

depmin	
depmax	
odelta	
resp0	
resp1	
resp2	
resp3	
resp4	

Enumerator

resp5	
resp6	
resp7	
resp8	
resp9	
stel	
stdp	
evel	
evdp	
mag	
user0	
user1	
user2	
user3	
user4	
user5	
user6	
user7	
user8	
user9	
dist	
az	
baz	
gcarc	
depmen	
cmpaz	
cmpinc	
xminimum	
xmaximum	
yminimum	
ymaximum	
delta	
b	
e	
o	
a	
t0	
t1	
t2	
t3	
t4	
t5	
t6	
t7	
t8	
t9	
f	
stla	
stlo	
evla	
evlo	

Enumerator

sb	
sdelta	
nzyear	
nzjday	
nzhour	
nzmin	
nzsec	
nzmsec	
nvhdr	
norid	
nevid	
npts	
nsnpts	
nwfid	
nxsize	
nysize	
iftype	
idep	
iztype	
iinst	
istreg	
ievreg	
ievtyp	
iqual	
isynth	
imagtyp	
imagsrc	
ibody	
leven	
lpspol	
lovrok	
lcalda	
kstnm	
kevnm	
khole	
ko	
ka	
kt0	
kt1	
kt2	
kt3	
kt4	
kt5	
kt6	
kt7	
kt8	
kt9	
kf	
kuser0	
kuser1	

Enumerator

kuser2	
kcmpnm	
knetwk	
kdatrd	
kinst	
data1	
data2	

```
00075                   {
00076   depmin,
00077   depmax,
00078   odelta,
00079   resp0,
00080   resp1,
00081   resp2,
00082   resp3,
00083   resp4,
00084   resp5,
00085   resp6,
00086   resp7,
00087   resp8,
00088   resp9,
00089   stel,
00090   stdp,
00091   evel,
00092   evdp,
00093   mag,
00094   user0,
00095   user1,
00096   user2,
00097   user3,
00098   user4,
00099   user5,
00100   user6,
00101   user7,
00102   user8,
00103   user9,
00104   dist,
00105   az,
00106   baz,
00107   gcarc,
00108   depmen,
00109   cmpaz,
00110   cmpinc,
00111   xminimum,
00112   xmaximum,
00113   yminimum,
00114   ymaximum,
00115   delta,
00116   b,
00117   e,
00118   o,
00119   a,
00120   t0,
00121   t1,
00122   t2,
00123   t3,
00124   t4,
00125   t5,
00126   t6,
00127   t7,
00128   t8,
00129   t9,
00130   f,
00131   stla,
00132   stlo,
00133   evla,
00134   evlo,
00135   sb,
00136   sdelta,
00137   nzyear,
00138   nzjday,
00139   nzhour,
00140   nzmin,
00141   nzsec,
00142   nzmsec,
00143   nvhdr,
00144   norid,
00145   nevid,
00146   npts,
```

```
00147   nsnpts,
00148   nwfid,
00149   nxsize,
00150   nysize,
00151   iftype,
00152   idep,
00153   iztype,
00154   iinst,
00155   istreg,
00156   ievreg,
00157   ievtyp,
00158   iqual,
00159   isynth,
00160   imagtyp,
00161   imagsrc,
00162   ibody,
00163   leven,
00164   lpspol,
00165   lovrok,
00166   lcalda,
00167   kstnm,
00168   kevnm,
00169   khole,
00170   ko,
00171   ka,
00172   kt0,
00173   kt1,
00174   kt2,
00175   kt3,
00176   kt4,
00177   kt5,
00178   kt6,
00179   kt7,
00180   kt8,
00181   kt9,
00182   kf,
00183   kuser0,
00184   kuser1,
00185   kuser2,
00186   kcmpnm,
00187   knetwk,
00188   kdatrd,
00189   kinst,
00190   data1,
00191   data2
00192 };
```

6.1.2.2 Type

```
enum class pssp::Type  [strong]
```

Data-type enumeration.

Allows maintaining the type of data (string, integer, float, double, bool) for an object since this isn't supported by default in C++.

Enumerator

string_	String data-type.
int_	Integer data-type.
float_	Float data-type.
double_	Double data-type.
bool_	Boolean data-type.

```
00032                {
00033   string_,
00034   int_,
00035   float_,
00036   double_,
```

```
00037   bool_,
00038 };
```

6.1.3 Variable Documentation

6.1.3.1 field_info

```
const std::unordered_map<Field, trace_info> pssp::field_info
00318                                                       {
00319     // Floats
00320     {Field::depmin, {0, 0, "DepMin", Type::float_}},
00321     {Field::depmax, {1, 1, "DepMax", Type::float_}},
00322     {Field::odelta, {2, 2, "ODelta", Type::float_}},
00323     {Field::resp0, {3, 3, "Resp0", Type::float_}},
00324     {Field::resp1, {4, 4, "Resp1", Type::float_}},
00325     {Field::resp2, {5, 5, "Resp2", Type::float_}},
00326     {Field::resp3, {6, 6, "Resp3", Type::float_}},
00327     {Field::resp4, {7, 7, "Resp4", Type::float_}},
00328     {Field::resp5, {8, 8, "Resp5", Type::float_}},
00329     {Field::resp6, {9, 9, "Resp6", Type::float_}},
00330     {Field::resp7, {10, 10, "Resp7", Type::float_}},
00331     {Field::resp8, {11, 11, "Resp8", Type::float_}},
00332     {Field::resp9, {12, 12, "Resp9", Type::float_}},
00333     {Field::stel, {13, 13, "StEl", Type::float_}},
00334     {Field::stdp, {14, 14, "StDp", Type::float_}},
00335     {Field::evel, {15, 15, "EvEl", Type::float_}},
00336     {Field::evdp, {16, 16, "EvDp", Type::float_}},
00337     {Field::mag, {17, 17, "Mag", Type::float_}},
00338     {Field::user0, {18, 18, "User0", Type::float_}},
00339     {Field::user1, {19, 19, "User1", Type::float_}},
00340     {Field::user2, {20, 20, "User2", Type::float_}},
00341     {Field::user3, {21, 21, "User3", Type::float_}},
00342     {Field::user4, {21, 22, "User4", Type::float_}},
00343     {Field::user5, {23, 23, "User5", Type::float_}},
00344     {Field::user6, {24, 24, "User6", Type::float_}},
00345     {Field::user7, {25, 25, "User7", Type::float_}},
00346     {Field::user8, {26, 26, "User8", Type::float_}},
00347     {Field::user9, {27, 27, "User9", Type::float_}},
00348     {Field::dist, {28, 28, "Dist", Type::float_}},
00349     {Field::az, {29, 29, "Az", Type::float_}},
00350     {Field::baz, {30, 30, "BAz", Type::float_}},
00351     {Field::gcarc, {31, 31, "GCArc", Type::float_}},
00352     {Field::depmen, {32, 32, "DepMen", Type::float_}},
00353     {Field::cmpaz, {33, 33, "CmpAz", Type::float_}},
00354     {Field::cmpinc, {34, 34, "CmpInc", Type::float_}},
00355     {Field::xminimum, {35, 35, "XMinimum", Type::float_}},
00356     {Field::xmaximum, {36, 36, "XMaximum", Type::float_}},
00357     {Field::yminimum, {37, 37, "YMinimum", Type::float_}},
00358     {Field::ymaximum, {38, 38, "YMaximum", Type::float_}},
00359     // Doubles
00360     {Field::delta, {39, 0, "Delta", Type::double_}},
00361     {Field::b, {40, 1, "B", Type::double_}},
00362     {Field::e, {41, 2, "E", Type::double_}},
00363     {Field::o, {42, 3, "O", Type::double_}},
00364     {Field::a, {43, 4, "A", Type::double_}},
00365     {Field::t0, {44, 5, "T0", Type::double_}},
00366     {Field::t1, {45, 6, "T1", Type::double_}},
00367     {Field::t2, {46, 7, "T2", Type::double_}},
00368     {Field::t3, {47, 8, "T3", Type::double_}},
00369     {Field::t4, {48, 9, "T4", Type::double_}},
00370     {Field::t5, {49, 10, "T5", Type::double_}},
00371     {Field::t6, {50, 11, "T6", Type::double_}},
00372     {Field::t7, {51, 12, "T7", Type::double_}},
00373     {Field::t8, {52, 13, "T8", Type::double_}},
00374     {Field::t9, {53, 14, "T9", Type::double_}},
00375     {Field::f, {54, 15, "F", Type::double_}},
00376     {Field::stla, {55, 16, "StLa", Type::double_}},
00377     {Field::stlo, {56, 17, "StLo", Type::double_}},
00378     {Field::evla, {57, 18, "EvLa", Type::double_}},
00379     {Field::evlo, {58, 19, "EvLo", Type::double_}},
00380     {Field::sb, {59, 20, "sB", Type::double_}},
00381     {Field::sdelta, {60, 21, "sDelta", Type::double_}},
00382     // Ints
00383     {Field::nzyear, {61, 0, "nzYear", Type::int_}},
00384     {Field::nzjday, {62, 1, "nzJDay", Type::int_}},
00385     {Field::nzhour, {63, 2, "nzHour", Type::int_}},
00386     {Field::nzmin, {64, 3, "nzMin", Type::int_}},
00387     {Field::nzsec, {65, 4, "nzSec", Type::int_}},
00388     {Field::nzmsec, {66, 5, "nzMSec", Type::int_}},
00389     {Field::nvhdr, {67, 6, "nVHdr", Type::int_}},
```

```
00390     {Field::norid, {68, 7, "nOrID", Type::int_}},
00391     {Field::nevid, {69, 8, "nEvID", Type::int_}},
00392     {Field::npts, {70, 9, "nPts", Type::int_}},
00393     {Field::nsnpts, {71, 10, "nsnPts", Type::int_}},
00394     {Field::nwfid, {72, 11, "nWfID", Type::int_}},
00395     {Field::nxsize, {73, 12, "nXSize", Type::int_}},
00396     {Field::nysize, {74, 13, "nYSize", Type::int_}},
00397     {Field::iftype, {75, 14, "iFType", Type::int_}},
00398     {Field::idep, {76, 15, "iDep", Type::int_}},
00399     {Field::iztype, {77, 16, "iZType", Type::int_}},
00400     {Field::iinst, {78, 17, "iInst", Type::int_}},
00401     {Field::istreg, {79, 18, "iStReg", Type::int_}},
00402     {Field::ievreg, {80, 19, "iEvReg", Type::int_}},
00403     {Field::ievtyp, {81, 20, "iEvTyp", Type::int_}},
00404     {Field::iqual, {82, 21, "iQual", Type::int_}},
00405     {Field::isynth, {83, 22, "iSynth", Type::int_}},
00406     {Field::imagtyp, {84, 23, "iMagTyp", Type::int_}},
00407     {Field::imagsrc, {85, 24, "iMagSrc", Type::int_}},
00408     {Field::ibody, {86, 25, "iBody", Type::int_}},
00409     // Bools
00410     {Field::leven, {87, 0, "lEven", Type::bool_}},
00411     {Field::lpspol, {88, 1, "lPsPol", Type::bool_}},
00412     {Field::lovrok, {89, 2, "lOvrOK", Type::bool_}},
00413     {Field::lcalda, {90, 3, "lCalDA", Type::bool_}},
00414     // Strings
00415     {Field::kstnm, {91, 0, "kStNm", Type::string_}},
00416     {Field::kevnm, {92, 1, "kEvNm", Type::string_}},
00417     {Field::khole, {93, 2, "kHole", Type::string_}},
00418     {Field::ko, {94, 3, "kO", Type::string_}},
00419     {Field::ka, {95, 4, "kA", Type::string_}},
00420     {Field::kt0, {96, 5, "kT0", Type::string_}},
00421     {Field::kt1, {97, 6, "kT1", Type::string_}},
00422     {Field::kt2, {98, 7, "kT2", Type::string_}},
00423     {Field::kt3, {99, 8, "kT3", Type::string_}},
00424     {Field::kt4, {100, 9, "kT4", Type::string_}},
00425     {Field::kt5, {101, 10, "kT5", Type::string_}},
00426     {Field::kt6, {102, 11, "kT6", Type::string_}},
00427     {Field::kt7, {103, 12, "kT7", Type::string_}},
00428     {Field::kt8, {104, 13, "kT8", Type::string_}},
00429     {Field::kt9, {105, 14, "kT9", Type::string_}},
00430     {Field::kf, {106, 15, "kF", Type::string_}},
00431     {Field::kuser0, {107, 16, "kUser0", Type::string_}},
00432     {Field::kuser1, {108, 17, "kUser1", Type::string_}},
00433     {Field::kuser2, {109, 18, "kUser2", Type::string_}},
00434     {Field::kcmpnm, {110, 19, "kCmpNm", Type::string_}},
00435     {Field::knetwk, {111, 20, "kNetwk", Type::string_}},
00436     {Field::kdatrd, {112, 21, "kDatRd", Type::string_}},
00437     {Field::kinst, {113, 22, "kInst", Type::string_}},
00438     // Data
00439     {Field::data1, {114, 0, "Data1", Type::int_}},
00440     {Field::data2, {115, 1, "Data2", Type::int_}}};
```

6.1.3.2 field_num

```
const std::unordered_map<size_t, Field> pssp::field_num
00195                                                 {// Floats
00196                                                    {0, Field::depmin},
00197                                                    {1, Field::depmax},
00198                                                    {2, Field::odelta},
00199                                                    {3, Field::resp0},
00200                                                    {4, Field::resp1},
00201                                                    {5, Field::resp2},
00202                                                    {6, Field::resp3},
00203                                                    {7, Field::resp4},
00204                                                    {8, Field::resp5},
00205                                                    {9, Field::resp6},
00206                                                    {10, Field::resp7},
00207                                                    {11, Field::resp8},
00208                                                    {12, Field::resp9},
00209                                                    {13, Field::stel},
00210                                                    {14, Field::stdp},
00211                                                    {15, Field::evel},
00212                                                    {16, Field::evdp},
00213                                                    {17, Field::mag},
00214                                                    {18, Field::user0},
00215                                                    {19, Field::user1},
00216                                                    {20, Field::user2},
00217                                                    {21, Field::user3},
00218                                                    {22, Field::user4},
00219                                                    {23, Field::user5},
00220                                                    {24, Field::user6},
00221                                                    {25, Field::user7},
00222                                                    {26, Field::user8},
```

```
00223                                   {27, Field::user9},
00224                                   {28, Field::dist},
00225                                   {29, Field::az},
00226                                   {30, Field::baz},
00227                                   {31, Field::gcarc},
00228                                   {32, Field::depmen},
00229                                   {33, Field::cmpaz},
00230                                   {34, Field::cmpinc},
00231                                   {35, Field::xminimum},
00232                                   {36, Field::xmaximum},
00233                                   {37, Field::yminimum},
00234                                   {38, Field::ymaximum},
00235                                   // Doubles
00236                                   {39, Field::delta},
00237                                   {40, Field::b},
00238                                   {41, Field::e},
00239                                   {42, Field::o},
00240                                   {43, Field::a},
00241                                   {44, Field::t0},
00242                                   {45, Field::t1},
00243                                   {46, Field::t2},
00244                                   {47, Field::t3},
00245                                   {48, Field::t4},
00246                                   {49, Field::t5},
00247                                   {50, Field::t6},
00248                                   {51, Field::t7},
00249                                   {52, Field::t8},
00250                                   {53, Field::t9},
00251                                   {54, Field::f},
00252                                   {55, Field::stla},
00253                                   {56, Field::stlo},
00254                                   {57, Field::evla},
00255                                   {58, Field::evlo},
00256                                   {59, Field::sb},
00257                                   {60, Field::sdelta},
00258                                   // Ints
00259                                   {61, Field::nzyear},
00260                                   {62, Field::nzjday},
00261                                   {63, Field::nzhour},
00262                                   {64, Field::nzmin},
00263                                   {65, Field::nzsec},
00264                                   {66, Field::nzmsec},
00265                                   {67, Field::nvhdr},
00266                                   {68, Field::norid},
00267                                   {69, Field::nevid},
00268                                   {70, Field::npts},
00269                                   {71, Field::nsnpts},
00270                                   {72, Field::nwfid},
00271                                   {73, Field::nxsize},
00272                                   {74, Field::nysize},
00273                                   {75, Field::iftype},
00274                                   {76, Field::idep},
00275                                   {77, Field::iztype},
00276                                   {78, Field::iinst},
00277                                   {79, Field::istreg},
00278                                   {80, Field::ievreg},
00279                                   {81, Field::ievtyp},
00280                                   {82, Field::iqual},
00281                                   {83, Field::isynth},
00282                                   {84, Field::imagtyp},
00283                                   {85, Field::imagsrc},
00284                                   {86, Field::ibody},
00285                                   // Bools
00286                                   {87, Field::leven},
00287                                   {88, Field::lpspol},
00288                                   {89, Field::lovrok},
00289                                   {90, Field::lcalda},
00290                                   // Strings
00291                                   {91, Field::kstnm},
00292                                   {92, Field::kevnm},
00293                                   {93, Field::khole},
00294                                   {94, Field::ko},
00295                                   {95, Field::ka},
00296                                   {96, Field::kt0},
00297                                   {97, Field::kt1},
00298                                   {98, Field::kt2},
00299                                   {99, Field::kt3},
00300                                   {100, Field::kt4},
00301                                   {101, Field::kt5},
00302                                   {102, Field::kt6},
00303                                   {103, Field::kt7},
00304                                   {104, Field::kt8},
00305                                   {105, Field::kt9},
00306                                   {106, Field::kf},
00307                                   {107, Field::kuser0},
00308                                   {108, Field::kuser1},
00309                                   {109, Field::kuser2},
```

```
00310                                                   {110, Field::kcmpnm},
00311                                                   {111, Field::knetwk},
00312                                                   {112, Field::kdatrd},
00313                                                   {113, Field::kinst},
00314                                                   // Data
00315                                                   {114, Field::data1},
00316                                                   {115, Field::data2}};
```

6.1.3.3 type_names

```
const std::unordered_map<Type, const std::string> pssp::type_names
```

Initial value:

```
{
    {Type::string_, "string"},
    {Type::int_, "int"},
    {Type::float_, "float"},
    {Type::double_, "double"},
    {Type::bool_, "bool"}}
```

Map Type to string-name.

Used to provide labels for the trace_info struct.

```
00045                                                   {
00046     {Type::string_, "string"},
00047     {Type::int_, "int"},
00048     {Type::float_, "float"},
00049     {Type::double_, "double"},
00050     {Type::bool_, "bool"}};
```

6.2 pssp::about Namespace Reference

Variables

- constexpr int button_width {75}
- constexpr int button_height {25}
- constexpr int text_height {90}
- constexpr int height {text_height + button_height + 10}
- constexpr int text_width {330}
- constexpr int width {text_width + 50}

6.2.1 Variable Documentation

6.2.1.1 button_height

```
constexpr int pssp::about::button_height {25}  [constexpr]
00022 {25};
```

6.2.1.2 button_width

```
constexpr int pssp::about::button_width {75}  [constexpr]
00021 {75};
```

6.2.1.3 height

```
constexpr int pssp::about::height {text_height + button_height + 10}  [constexpr]
00024 {text_height + button_height + 10};
```

6.2.1.4 text_height

```
constexpr int pssp::about::text_height {90}  [constexpr]
00023 {90};
```

6.2.1.5 text_width

```
constexpr int pssp::about::text_width {330}  [constexpr]
00025 {330};
```

6.2.1.6 width

```
constexpr int pssp::about::width {text_width + 50}  [constexpr]
00026 {text_width + 50};
```

6.3 pssp::constants Namespace Reference

Variables

- constexpr int sac_float {39}

  Number of float columns for SAC records.
- constexpr int sac_double {22}

  Number of double columns for SAC records.
- constexpr int sac_int {26}

  Number of integer columns for SAC records.
- constexpr int sac_bool {4}

  Number of boolean columns for SAC records.
- constexpr int sac_string {22 + 1}

  Number of string columns for SAC records.
- constexpr int sac_data {2}

  Number of possible data vectors for a SAC record.

6.3.1 Variable Documentation

6.3.1.1 sac_bool

```
constexpr int pssp::constants::sac_bool {4}  [constexpr]
```

Number of boolean columns for SAC records.
```
00024 {4};
```

6.3.1.2 sac_data

```
constexpr int pssp::constants::sac_data {2}  [constexpr]
```

Number of possible data vectors for a SAC record.
```
00028 {2};
```

6.3.1.3 sac_double

```
constexpr int pssp::constants::sac_double {22}  [constexpr]
```

Number of double columns for SAC records.
```
00020 {22};
```

6.3.1.4 sac_float

```
constexpr int pssp::constants::sac_float {39}  [constexpr]
```

Number of float columns for SAC records.
```
00018 {39};
```

6.3.1.5 sac_int

```
constexpr int pssp::constants::sac_int {26}  [constexpr]
```

Number of integer columns for SAC records.
```
00022 {26};
```

6.3.1.6 sac_string

```
constexpr int pssp::constants::sac_string {22 + 1}  [constexpr]
```

Number of string columns for SAC records.
```
00026 {22 + 1};
```

6.4 pssp::datasheet Namespace Reference

Classes

- struct Cell
- struct Spec

Variables

- constexpr int font_size {14}
- constexpr int cell_buffer {3}
- constexpr int max_chars {10}
- const std::string edit_chars {"0123456789+-\r\n"}

6.4.1 Variable Documentation

6.4.1.1 cell_buffer

```
constexpr int pssp::datasheet::cell_buffer {3}  [constexpr]
00049 {3};
```

6.4.1.2 edit_chars

```
const std::string pssp::datasheet::edit_chars {"0123456789+-\r\n"}
00051 {"0123456789+-\r\n"};
```

6.4.1.3 font_size

```
constexpr int pssp::datasheet::font_size {14}  [constexpr]
00048 {14};
```

6.4.1.4 max_chars

```
constexpr int pssp::datasheet::max_chars {10}  [constexpr]
00050 {10};
```

6.5 pssp::mw Namespace Reference

Variables

- constexpr int minimum_x {300}
- constexpr int minimum_y {300}
- constexpr int menu_height {25}

6.5.1 Variable Documentation

6.5.1.1 menu_height

```
constexpr int pssp::mw::menu_height {25}  [constexpr]
00032 {25};
```

6.5.1.2 minimum_x

```
constexpr int pssp::mw::minimum_x {300}  [constexpr]
00030 {300};
```

6.5.1.3 minimum_y

```
constexpr int pssp::mw::minimum_y {300}  [constexpr]
00031 {300};
```

6.6 pssp::structs Namespace Reference

Classes

- struct Geometry
- struct Grid

6.7 pssp::welcome Namespace Reference

Variables

- constexpr int button_width {125}
- constexpr int button_height {25}
- constexpr int text_height {50}
- constexpr int height {text_height + button_height + 10}
- constexpr int text_width {380}
- constexpr int width {text_width + 20}

6.7.1 Variable Documentation

6.7.1.1 button_height

```
constexpr int pssp::welcome::button_height {25}  [constexpr]
00020 {25};
```

6.7.1.2 button_width

```
constexpr int pssp::welcome::button_width {125}  [constexpr]
00019 {125};
```

6.7.1.3 height

```
constexpr int pssp::welcome::height {text_height + button_height + 10}  [constexpr]
00022 {text_height + button_height + 10};
```

6.7.1.4 text_height

```
constexpr int pssp::welcome::text_height {50}  [constexpr]
00021 {50};
```

6.7.1.5 text_width

```
constexpr int pssp::welcome::text_width {380}  [constexpr]
00023 {380};
```

6.7.1.6 width

```
constexpr int pssp::welcome::width {text_width + 20}  [constexpr]
00024 {text_width + 20};
```

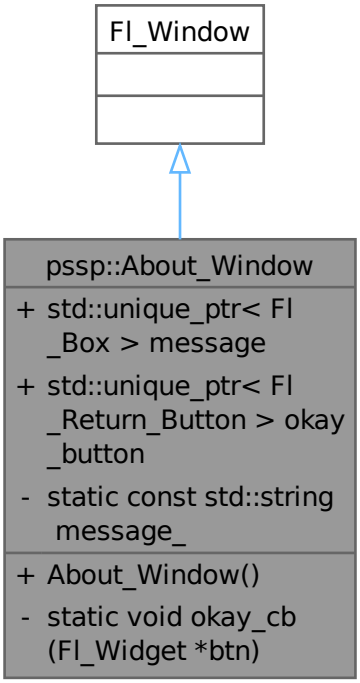
Chapter 7

Class Documentation

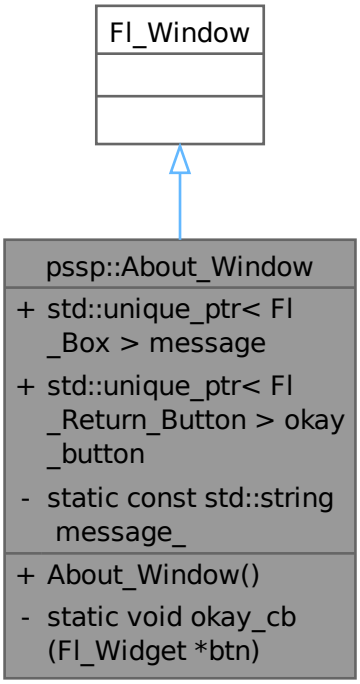
7.1 pssp::About_Window Class Reference

`#include <About.hpp>`

Inheritance diagram for pssp::About_Window:



Collaboration diagram for pssp::About_Window:



Public Member Functions

- About_Window ()

Public Attributes

- std::unique_ptr< Fl_Box > message {}
- std::unique_ptr< Fl_Return_Button > okay_button {}

Static Private Member Functions

- static void okay_cb (Fl_Widget *btn)

Static Private Attributes

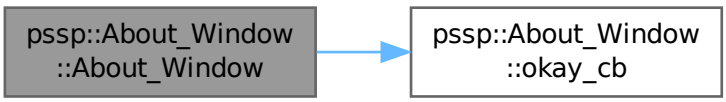
- static const std::string message_

7.1.1 Constructor & Destructor Documentation

7.1.1.1 About_Window()

```
pssp::About_Window::About_Window ( )
00006                           : Fl_Window(0, 0, 0, 0, "About") {
00007   this->begin();
00008   structs::Geometry geo{};
00009   Fl::screen_work_area(geo.x_pos, geo.y_pos, geo.width, geo.height);
00010   geo.x_pos = ((geo.width - about::width) / 2);
00011   geo.y_pos = ((geo.height - about::height) / 2);
00012   this->resize(geo.x_pos, geo.y_pos, about::width, about::height);
00013   this->box(FL_BORDER_BOX);
00014   set_modal();
00015   message = std::make_unique<Fl_Box>(about::width - about::text_width, 0,
00016                                      about::text_width, about::text_height);
00017   okay_button = std::make_unique<Fl_Return_Button>(
00018       (about::width - about::button_width) / 2, about::text_height,
00019       about::button_width, about::button_height, "Okay");
00020   message->label(message_.c_str());
00021   message->align(FL_ALIGN_CENTER);
00022   okay_button->callback(okay_cb);
00023   this->end();
00024 }
```

Here is the call graph for this function:

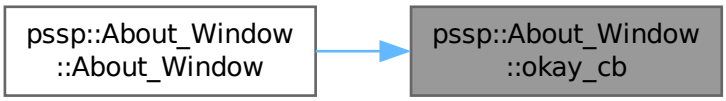


7.1.2 Member Function Documentation

7.1.2.1 okay_cb()

```
void pssp::About_Window::okay_cb (
             Fl_Widget * btn )  [static], [private]
00026 { btn->parent()->hide(); }
```

Here is the caller graph for this function:



7.1.3 Member Data Documentation

7.1.3.1 message

```
std::unique_ptr<Fl_Box> pssp::About_Window::message {}
00032 {};
```

7.1.3.2 message_

```
const std::string pssp::About_Window::message_  [inline], [static], [private]
```

Initial value:

```
{"Website: https://arbCoding.github.io/PsSp/\n"
              "GitHub: https://arbCoding.github.com/PsSp\n"
              "Developer: Alexander R. Blanchette <arbCoding@gmail.com>"
              "License: MIT"}
00039              {"Website: https://arbCoding.github.io/PsSp/\n"
00040               "GitHub: https://arbCoding.github.com/PsSp\n"
00041               "Developer: Alexander R. Blanchette <arbCoding@gmail.com>"
00042               "License: MIT"};
```

7.1.3.3 okay_button

```
std::unique_ptr<Fl_Return_Button> pssp::About_Window::okay_button {}
00033 {};
```

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

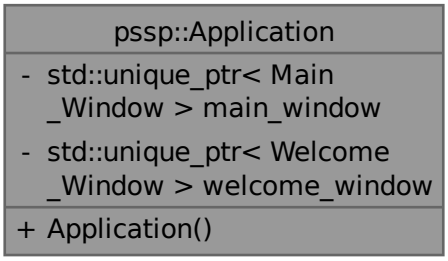
- include/PsSp/Windows/About.hpp
- src/Windows/About.cpp

7.2 pssp::Application Class Reference

Main application class.

```
#include <Application.hpp>
```

Collaboration diagram for pssp::Application:



Public Member Functions

- Application ()

  Application constructor.

Private Attributes

- std::unique_ptr< Main_Window > main_window {}

  Unique Pointer to the Main_Window object.
- std::unique_ptr< Welcome_Window > welcome_window {}

  Unique Pointer to the Welcome_Window object.

7.2.1 Detailed Description

Main application class.

This manages the application (created in main()).

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Application()

```
pssp::Application::Application ( )
```

Application constructor.

Creates the main_window object and the welcome_window object.

Logs status after creation.

```
00020                           {
00021     main_window = std::make_unique<Main_Window>();
00022     main_window->show();
00023     welcome_window = std::make_unique<Welcome_Window>();
00024     welcome_window->show();
00025     spdlog::trace("Application ready.");
00026 }
```

7.2.3 Member Data Documentation

7.2.3.1 main_window

```
std::unique_ptr<Main_Window> pssp::Application::main_window {}  [private]
```

Unique Pointer to the Main_Window object.

```
00038 {};
```

7.2.3.2 welcome_window

```
std::unique_ptr<Welcome_Window> pssp::Application::welcome_window {}  [private]
```

Unique Pointer to the Welcome_Window object.
```
00040 {};
```

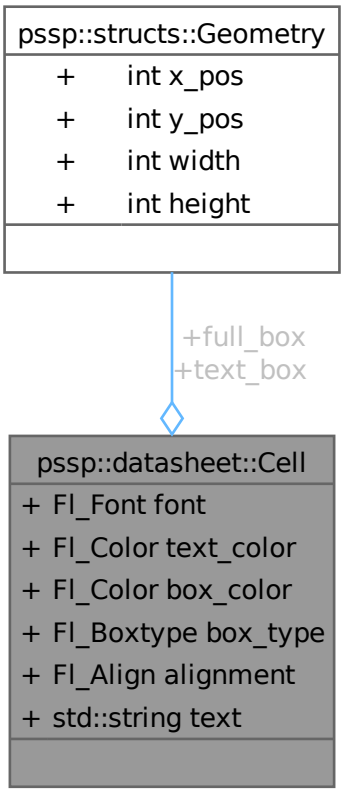
The documentation for this class was generated from the following files:

- include/PsSp/Application/Application.hpp
- src/Application/Application.cpp

7.3 pssp::datasheet::Cell Struct Reference

```
#include <Datasheet.hpp>
```

Collaboration diagram for pssp::datasheet::Cell:



Public Attributes

- structs::Geometry full_box {}
- structs::Geometry text_box {}
- Fl_Font font {FL_HELVETICA}
- Fl_Color text_color {FL_BLACK}
- Fl_Color box_color {FL_GRAY}
- Fl_Boxtype box_type {FL_THIN_UP_BOX}
- Fl_Align alignment {FL_ALIGN_CENTER}
- std::string text {}

7.3.1 Member Data Documentation

7.3.1.1 alignment

```
Fl_Align pssp::datasheet::Cell::alignment {FL_ALIGN_CENTER}
00061 {FL_ALIGN_CENTER};
```

7.3.1.2 box_color

```
Fl_Color pssp::datasheet::Cell::box_color {FL_GRAY}
00059 {FL_GRAY};
```

7.3.1.3 box_type

```
Fl_Boxtype pssp::datasheet::Cell::box_type {FL_THIN_UP_BOX}
00060 {FL_THIN_UP_BOX};
```

7.3.1.4 font

```
Fl_Font pssp::datasheet::Cell::font {FL_HELVETICA}
00057 {FL_HELVETICA};
```

7.3.1.5 full_box

```
structs::Geometry pssp::datasheet::Cell::full_box {}
00054 {};
```

7.3.1.6 text

```
std::string pssp::datasheet::Cell::text {}
00063 {};
```

7.3.1.7 text_box

```
structs::Geometry pssp::datasheet::Cell::text_box {}
00056 {};
```

7.3.1.8 text_color

```
Fl_Color pssp::datasheet::Cell::text_color {FL_BLACK}
00058 {FL_BLACK};
```

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

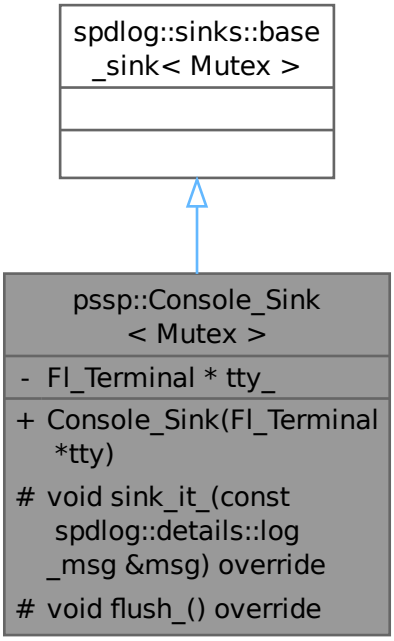
- include/PsSp/Widgets/Datasheet.hpp

7.4 pssp::Console_Sink< Mutex > Class Template Reference

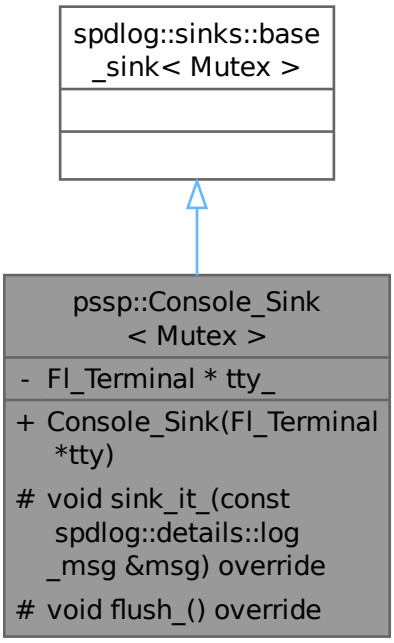
Sink (receiver) of log messages for PsSp console.

`#include <Console_Sink.hpp>`

Inheritance diagram for pssp::Console_Sink< Mutex >:



Collaboration diagram for pssp::Console_Sink< Mutex >:



Public Member Functions

- Console_Sink (Fl_Terminal ∗tty)
  Default constructor.

Protected Member Functions

- void sink_it_ (const spdlog::details::log_msg &msg) override
  Receives message from spdlog and then passes message to display console.
- void flush_ () override
  Clear (flush) Fl_Terminal.

Private Attributes

- Fl_Terminal ∗ tty_ {}
  Message receiver (console/terminal/tty).

7.4.1 Detailed Description

template<typename Mutex>
class pssp::Console_Sink< Mutex >

Sink (receiver) of log messages for PsSp console.

This class receiver logs from spdlog and passes them on to a FLTK terminal (FL_Terminal) object for presentation.

Todo At present it doesn't do log formatting (formatting is handled with console codes in the logs themselves). Formatting should be moved to here in the future for generality.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 Console_Sink()

```
template<typename Mutex >
pssp::Console_Sink< Mutex >::Console_Sink (
            Fl_Terminal * tty )  [inline], [explicit]
```

Default constructor.

Parameters

in	*tty*	Fl_Terminal* FLTK Terminal widget that will display the logs.

```
00053 { tty_ = tty; }
```

7.4.3 Member Function Documentation

7.4.3.1 flush_()

```
template<typename Mutex >
void pssp::Console_Sink< Mutex >::flush_ ( )  [inline], [override], [protected]
```

Clear (flush) Fl_Terminal.

```
00071 { tty_->clear(); }
```

7.4.3.2 sink_it_()

```
template<typename Mutex >
void pssp::Console_Sink< Mutex >::sink_it_ (
            const spdlog::details::log_msg & msg )  [inline], [override], [protected]
```

Receives message from spdlog and then passes message to display console.

Parameters

in	*msg*	spdlog::details::log_msg& Message to format and pass.

```
00061                                                   {
00062     // log_msg is a struct containing the log entry info like level, timestamp,
00063     // msg.raw contains the pre-formatted log
00064
00065     // If needed (very likely, but not madatory), the sink formats the message
00066     spdlog::memory_buf_t formatted{};
00067     spdlog::sinks::base_sink<Mutex>::formatter_->format(msg, formatted);
00068     tty_->append(fmt::to_string(formatted).c_str());
00069   }
```

7.4.4 Member Data Documentation

7.4.4.1 tty_

```
template<typename Mutex >
Fl_Terminal* pssp::Console_Sink< Mutex >::tty_ {}  [private]
```

Message receiver (console/terminal/tty).
00074 {};

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

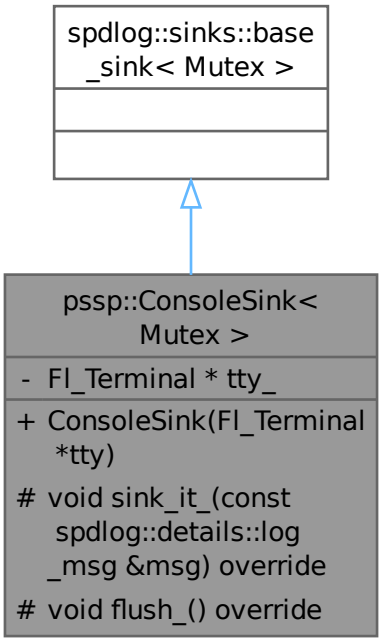
- include/PsSp/Logging/Console_Sink.hpp

7.5 pssp::ConsoleSink< Mutex > Class Template Reference

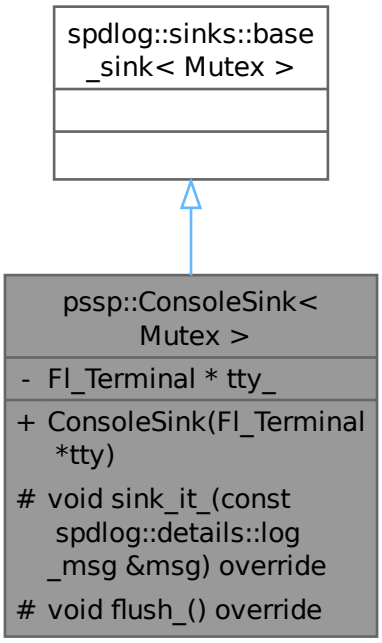
Sink (receiver) of log messages for PsSp console.

`#include <ConsoleSink.hpp>`

Inheritance diagram for pssp::ConsoleSink< Mutex >:



Collaboration diagram for pssp::ConsoleSink< Mutex >:



Public Member Functions

- ConsoleSink (Fl_Terminal ∗tty)
  Default constructor.

Protected Member Functions

- void sink_it_ (const spdlog::details::log_msg &msg) override
  Receives message from spdlog and then passes message to display console.
- void flush_ () override
  Clear (flush) Fl_Terminal.

Private Attributes

- Fl_Terminal ∗ tty_ {}
  Message receiver (console/terminal/tty).

7.5.1 Detailed Description

template<typename Mutex>
class pssp::ConsoleSink< Mutex >

Sink (receiver) of log messages for PsSp console.

This class receiver logs from spdlog and passes them on to a FLTK terminal (FL_Terminal) object for presentation.

Todo At present it doesn't do log formatting (formatting is handled with console codes in the logs themselves). Formatting should be moved to here in the future for generality.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 ConsoleSink()

```
template<typename Mutex >
pssp::ConsoleSink< Mutex >::ConsoleSink (
            Fl_Terminal * tty )  [inline], [explicit]
```

Default constructor.

Parameters

in	*tty*	Fl_Terminal* FLTK Terminal widget that will display the logs.

```
00053 { tty_ = tty; }
```

7.5.3 Member Function Documentation

7.5.3.1 flush_()

```
template<typename Mutex >
void pssp::ConsoleSink< Mutex >::flush_ ( )  [inline], [override], [protected]
```

Clear (flush) Fl_Terminal.

```
00072 { tty_->clear(); }
```

7.5.3.2 sink_it_()

```
template<typename Mutex >
void pssp::ConsoleSink< Mutex >::sink_it_ (
            const spdlog::details::log_msg & msg )  [inline], [override], [protected]
```

Receives message from spdlog and then passes message to display console.

Parameters

in	*msg*	spdlog::details::log_msg& Message to format and pass.

```
00062                                                   {
00063     // log_msg is a struct containing the log entry info like level, timestamp,
00064     // msg.raw contains the pre-formatted log
00065
00066     // If needed (very likely, but not madatory), the sink formats the message
00067     spdlog::memory_buf_t formatted{};
00068     spdlog::sinks::base_sink<Mutex>::formatter_->format(msg, formatted);
00069     tty_->append(fmt::to_string(formatted).c_str());
00070   }
```

7.5.4 Member Data Documentation

7.5.4.1 tty_

```
template<typename Mutex >
Fl_Terminal* pssp::ConsoleSink< Mutex >::tty_ {}  [private]
```

Message receiver (console/terminal/tty).
00075 {};

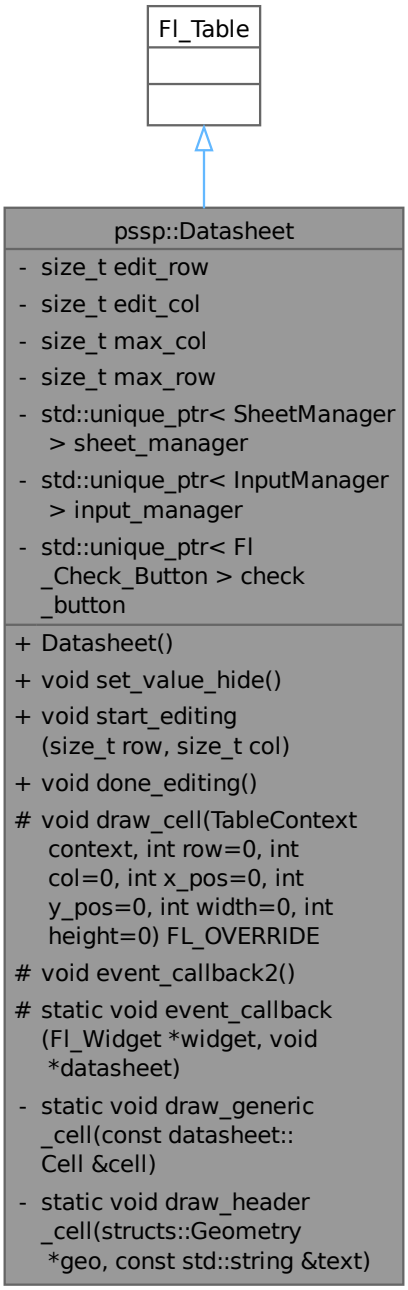
The documentation for this class was generated from the following file:

- include/PsSp/Logging/ConsoleSink.hpp

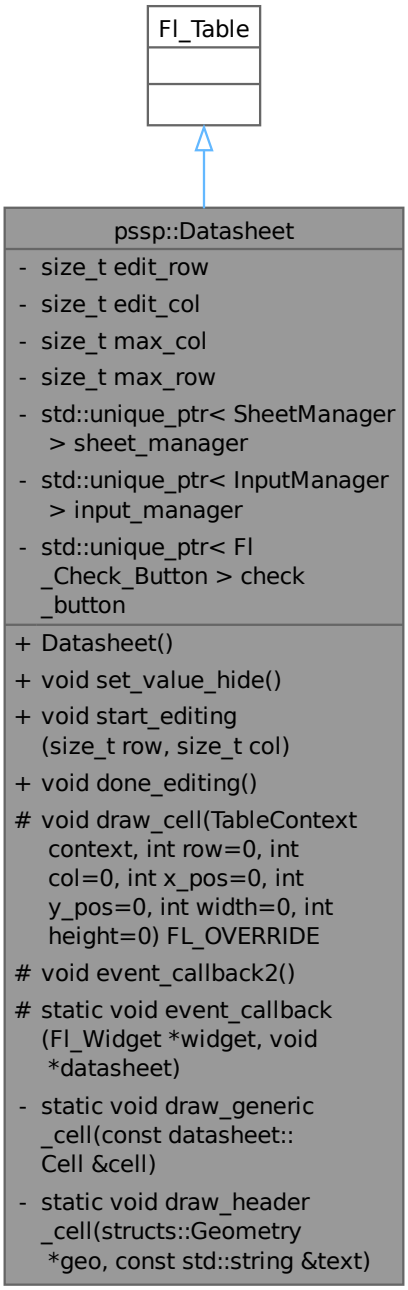
7.6 pssp::Datasheet Class Reference

#include <Datasheet.hpp>

Inheritance diagram for pssp::Datasheet:



Collaboration diagram for pssp::Datasheet:



Public Member Functions

- Datasheet ()
- void set_value_hide ()
- void start_editing (size_t row, size_t col)
- void done_editing ()

Protected Member Functions

- void draw_cell (TableContext context, int row=0, int col=0, int x_pos=0, int y_pos=0, int width=0, int height=0) FL_OVERRIDE
- void event_callback2 ()

Static Protected Member Functions

- static void event_callback (Fl_Widget *widget, void *datasheet)

Static Private Member Functions

- static void draw_generic_cell (const datasheet::Cell &cell)
- static void draw_header_cell (structs::Geometry *geo, const std::string &text)

Private Attributes

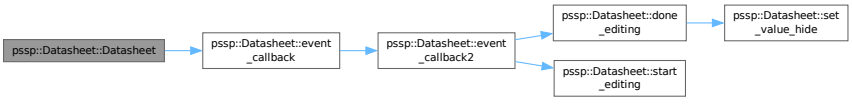
- size_t edit_row {0}
- size_t edit_col {0}
- size_t max_col {0}
- size_t max_row {0}
- std::unique_ptr< SheetManager > sheet_manager {}
- std::unique_ptr< InputManager > input_manager {}
- std::unique_ptr< Fl_Check_Button > check_button {}

7.6.1 Constructor & Destructor Documentation

7.6.1.1 Datasheet()

```
pssp::Datasheet::Datasheet ( )
00006                     : Fl_Table(0, 0, 0, 0) {
00007   spdlog::trace("Making \033[1mDatasheet\033[0m.");
00008   // trick to use event_callback2
00009   callback(&event_callback, reinterpret_cast<void *>(this));
00010   this->begin();
00011   this->when(static_cast<uchar>(FL_WHEN_NOT_CHANGED | this->when()));
00012   input_manager = std::make_unique<InputManager>();
00013   this->tab_cell_nav(1);  // enable tab navigation
00014   tooltip("Use keyboard to navigate cells:\n"
00015           "Arrow keys or Tab/Shift-Tab");
00016   sheet_manager = std::make_unique<SheetManager>();
00017   check_button = std::make_unique<Fl_Check_Button>(0, 0, 0, 0);
00018   check_button->hide();
00019   max_col = static_cast<size_t>(sheet_manager->cols());
00020   max_row = static_cast<size_t>(sheet_manager->rows());
00021   constexpr datasheet::Spec spec{25, 25, 25, 70};
00022   row_header(1);
00023   row_header_width(spec.header_width);
00024   row_height_all(spec.height);
00025   rows(static_cast<int>(max_row));
00026   col_header(1);
00027   col_header_height(spec.header_height);
00028   col_width_all(spec.width);
00029   cols(static_cast<int>(max_col));
00030   row_resize(1);
00031   col_resize(1);
00032   set_selection(0, 0, 0, 0);
00033   this->end();
00034   spdlog::trace("Done making \033[1mDatasheet\033[0m.");
00035 }
```

Here is the call graph for this function:

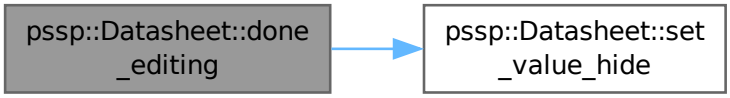


7.6.2 Member Function Documentation

7.6.2.1 done_editing()

```
void pssp::Datasheet::done_editing ( )
00120                                {
00121   if (input_manager->visible() || input_manager->modified) {
00122     set_value_hide();
00123     edit_row = 0;
00124     edit_col = 0;
00125   }
00126 }
```

Here is the call graph for this function:



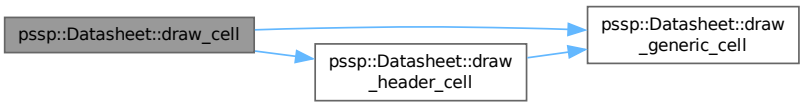
Here is the caller graph for this function:

7.6.2.2 draw_cell()

```
void pssp::Datasheet::draw_cell (
            TableContext context,
            int row = 0,
            int col = 0,
            int x_pos = 0,
            int y_pos = 0,
            int width = 0,
            int height = 0 )  [protected]
00159                                                                {
00160   // NOLINTEND(bugprone-easily-swappable-parameters)
00161   switch (context) {
00162   case CONTEXT_COL_HEADER: {
00163     structs::Geometry geo{x_pos, y_pos, width, height};
00164     draw_header_cell(
00165         &geo, field_info.at(field_num.at(static_cast<size_t>(col))).name);
00166   } break;
00167   case CONTEXT_ROW_HEADER: {
00168     structs::Geometry geo{x_pos, y_pos, width, height};
00169     draw_header_cell(&geo, std::to_string(row + 1));
00170   } break;
00171   case CONTEXT_CELL: {
00172     // This needs to be refactored
00173     datasheet::Cell cell{};
00174     cell.full_box = {x_pos, y_pos, width, height};
00175     cell.text_box = {x_pos + datasheet::cell_buffer,
```

```
00176                     y_pos + datasheet::cell_buffer,
00177                     width - (2 * datasheet::cell_buffer),
00178                     height - (2 * datasheet::cell_buffer)};
00179     const Field &field{field_num.at(static_cast<size_t>(col))};
00180     const trace_info &info{field_info.at(field)};
00181     if (info.type == Type::string_) {
00182       cell.text = sheet_manager->get_string(static_cast<size_t>(row), field);
00183     } else if (info.type == Type::int_) {
00184       std::ostringstream oss{};
00185       oss « sheet_manager->get_int(static_cast<size_t>(row), field);
00186       cell.text = oss.str();
00187     } else if (info.type == Type::float_) {
00188       std::ostringstream oss{};
00189       oss « sheet_manager->get_float(static_cast<size_t>(row), field);
00190       cell.text = oss.str();
00191     } else if (info.type == Type::double_) {
00192       std::ostringstream oss{};
00193       oss « sheet_manager->get_double(static_cast<size_t>(row), field);
00194       cell.text = oss.str();
00195     } else if (info.type == Type::bool_) {
00196       std::ostringstream oss{};
00197       oss « sheet_manager->get_bool(static_cast<size_t>(row), field);
00198       cell.text = oss.str();
00199     }
00200     cell.box_color = ((is_selected(row, col) != 0) ? FL_YELLOW : FL_WHITE);
00201     cell.alignment = FL_ALIGN_RIGHT;
00202     draw_generic_cell(cell);
00203   } break;
00204   default:
00205     return;
00206   }
00207 }
```

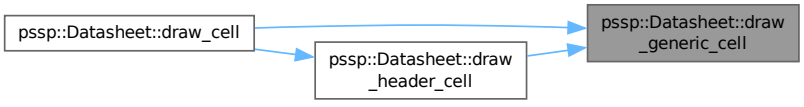
Here is the call graph for this function:



7.6.2.3 draw_generic_cell()

```
void pssp::Datasheet::draw_generic_cell (
            const datasheet::Cell & cell )  [static], [private]
00128                                                    {
00129   fl_font(cell.font, datasheet::font_size);
00130   fl_draw_box(cell.box_type, cell.full_box.x_pos, cell.full_box.y_pos,
00131               cell.full_box.width, cell.full_box.height, cell.box_color);
00132   fl_push_clip(cell.text_box.x_pos, cell.text_box.y_pos, cell.text_box.width,
00133                cell.text_box.height);
00134   fl_color(cell.text_color);
00135   fl_draw(cell.text.c_str(), cell.text_box.x_pos, cell.text_box.y_pos,
00136           cell.text_box.width, cell.text_box.height, cell.alignment);
00137   fl_pop_clip();
00138 }
```

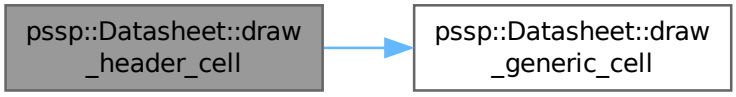
Here is the caller graph for this function:



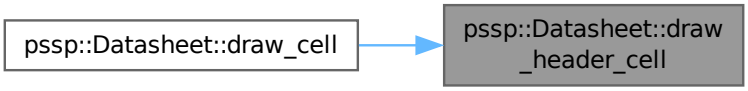
7.6.2.4 draw_header_cell()

```
void pssp::Datasheet::draw_header_cell (
            structs::Geometry * geo,
            const std::string & text )  [static], [private]
00141                                                       {
00142   datasheet::Cell cell{};
00143   cell.full_box = *geo;
00144   cell.text_box = cell.full_box;
00145   cell.font = FL_HELVETICA | FL_BOLD;
00146   cell.text = text;
00147   draw_generic_cell(cell);
00148 }
```

Here is the call graph for this function:



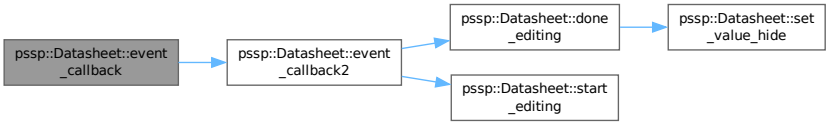
Here is the caller graph for this function:



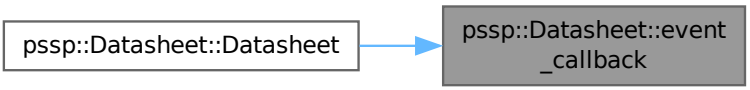
7.6.2.5 event_callback()

```
static void pssp::Datasheet::event_callback (
            Fl_Widget * widget,
            void * datasheet )  [inline], [static], [protected]
00085                                                           {
00086     (void)widget;
00087     reinterpret_cast<Datasheet *>(datasheet)->event_callback2();
00088   }
```

Here is the call graph for this function:



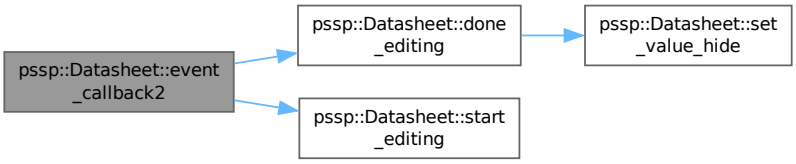
Here is the caller graph for this function:



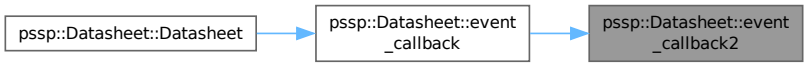
7.6.2.6 event_callback2()

```
void pssp::Datasheet::event_callback2 ( )  [protected]
00209                                {
00210   int row{callback_row()};
00211   int col{callback_col()};
00212   TableContext context{callback_context()};
00213   switch (context) {
00214   case CONTEXT_CELL: {
00215     switch (Fl::event()) {
00216     case FL_PUSH:
00217       start_editing(static_cast<size_t>(row), static_cast<size_t>(col));
00218       break;
00219     case FL_KEYBOARD:
00220       done_editing();
00221       if (Fl::event_state() == FL_COMMAND) {
00222         parent()->take_focus();
00223       } else if (datasheet::edit_chars.find(Fl::e_text[0]) !=
00224                  std::string::npos) {
00225         start_editing(static_cast<size_t>(row), static_cast<size_t>(col));
00226       }
00227       break;
00228     default:
00229       break;
00230     }
00231   } break;
00232   case CONTEXT_TABLE:
00233   case CONTEXT_ROW_HEADER:
00234   case CONTEXT_COL_HEADER:
00235     done_editing();
00236     break;
00237   default:
00238     return;
00239   }
00240 }
```

Here is the call graph for this function:



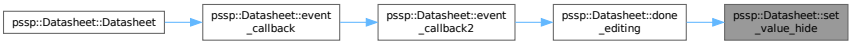
Here is the caller graph for this function:



7.6.2.7 set_value_hide()

```
void pssp::Datasheet::set_value_hide ( )
00045                                  {
00046   const Field &field{field_num.at(edit_col)};
00047   const trace_info &info{field_info.at(field)};
00048   switch (info.type) {
00049   case Type::string_:
00050     sheet_manager->set(edit_row, field, input_manager->value());
00051     break;
00052   case Type::int_:
00053     if (!input_manager->value().empty()) {
00054       sheet_manager->set(edit_row, field, std::stoi(input_manager->value()));
00055     } else {
00056       sheet_manager->set(edit_row, field, 0);
00057     }
00058     break;
00059   case Type::float_:
00060     if (!input_manager->value().empty()) {
00061       sheet_manager->set(edit_row, field, std::stof(input_manager->value()));
00062     } else {
00063       sheet_manager->set(edit_row, field, 0.0F);
00064     }
00065     break;
00066   case Type::double_:
00067     if (!input_manager->value().empty()) {
00068       sheet_manager->set(edit_row, field, std::stod(input_manager->value()));
00069     } else {
00070       sheet_manager->set(edit_row, field, 0.0);
00071     }
00072     break;
00073   case Type::bool_:
00074     // This is just junk for prototyping
00075     sheet_manager->set(edit_row, field, !input_manager->value().empty());
00076     break;
00077   default:
00078     break;
00079   }
00080   input_manager->cleanup();
00081   input_manager->modified = false;
00082   window()->cursor(FL_CURSOR_DEFAULT);  // deals with disappearing cursor
00083 }
```

Here is the caller graph for this function:



7.6.2.8 start_editing()

```
void pssp::Datasheet::start_editing (
            size_t row,
            size_t col )
```

```
00087                                                 {
00088   edit_row = row;
00089   edit_col = col;
00090   set_selection(static_cast<int>(row), static_cast<int>(col),
00091                 static_cast<int>(row), static_cast<int>(col));
00092   structs::Geometry geo{};
00093   find_cell(CONTEXT_CELL, static_cast<int>(row), static_cast<int>(col),
00094             geo.x_pos, geo.y_pos, geo.width, geo.height);
00095   // Need to refactor
00096   const Field &field{field_num.at(col)};
00097   const trace_info &info{field_info.at(field)};
00098   if (info.type == Type::string_) {
00099     input_manager->start_editing(info, geo,
00100                                  sheet_manager->get_string(row, field));
00101   } else if (info.type == Type::int_) {
00102     std::ostringstream oss{};
00103     oss « sheet_manager->get_int(row, field);
00104     input_manager->start_editing(info, geo, oss.str());
00105   } else if (info.type == Type::float_) {
00106     std::ostringstream oss{};
00107     oss « sheet_manager->get_float(row, field);
00108     input_manager->start_editing(info, geo, oss.str());
00109   } else if (info.type == Type::double_) {
00110     std::ostringstream oss{};
00111     oss « sheet_manager->get_double(row, field);
00112     input_manager->start_editing(info, geo, oss.str());
00113   } else if (info.type == Type::bool_) {
00114     std::ostringstream oss{};
00115     oss « sheet_manager->get_bool(row, field);
00116     input_manager->start_editing(info, geo, oss.str());
00117   }
00118 }
```

Here is the caller graph for this function:

7.6.3 Member Data Documentation

7.6.3.1 check_button

```
std::unique_ptr<Fl_Check_Button> pssp::Datasheet::check_button {}  [private]
00101 {};
```

7.6.3.2 edit_col

```
size_t pssp::Datasheet::edit_col {0}  [private]
00094 {0};
```

7.6.3.3 edit_row

```
size_t pssp::Datasheet::edit_row {0}  [private]
00092 {0};
```

7.6.3.4 input_manager

```
std::unique_ptr<InputManager> pssp::Datasheet::input_manager {}  [private]
00100 {};
```

7.6.3.5 max_col

```
size_t pssp::Datasheet::max_col {0}  [private]
00096 {0};
```

7.6.3.6 max_row

```
size_t pssp::Datasheet::max_row {0}  [private]
00098 {0};
```

7.6.3.7 sheet_manager

```
std::unique_ptr<SheetManager> pssp::Datasheet::sheet_manager {}  [private]
00099 {};
```

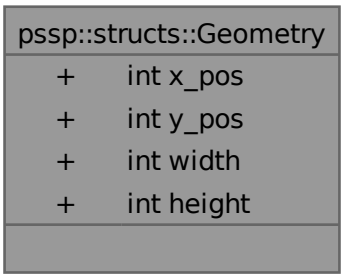
The documentation for this class was generated from the following files:

- include/PsSp/Widgets/Datasheet.hpp
- src/Widgets/Datasheet.cpp

7.7 pssp::structs::Geometry Struct Reference

```
#include <Structs.hpp>
```

Collaboration diagram for pssp::structs::Geometry:



Public Attributes

- int x_pos {0}
- int y_pos {0}
- int width {0}
- int height {0}

7.7.1 Member Data Documentation

7.7.1.1 height

```
int pssp::structs::Geometry::height {0}
00016 {0};
```

7.7.1.2 width

```
int pssp::structs::Geometry::width {0}
00014 {0};
```

7.7.1.3 x_pos

```
int pssp::structs::Geometry::x_pos {0}
00010 {0};
```

7.7.1.4 y_pos

```
int pssp::structs::Geometry::y_pos {0}
00012 {0};
```

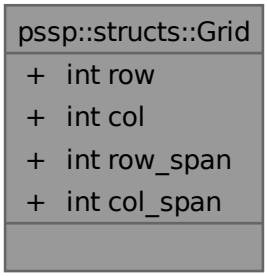
The documentation for this struct was generated from the following file:

- include/PsSp/Utility/Structs.hpp

7.8 pssp::structs::Grid Struct Reference

```
#include <Structs.hpp>
```

Collaboration diagram for pssp::structs::Grid:



Public Attributes

- int row {0}
- int col {0}
- int row_span {0}
- int col_span {0}

7.8.1 Member Data Documentation

7.8.1.1 col

```
int pssp::structs::Grid::col {0}
00023 {0};
```

7.8.1.2 col_span

```
int pssp::structs::Grid::col_span {0}
00027 {0};
```

7.8.1.3 row

```
int pssp::structs::Grid::row {0}
00021 {0};
```

7.8.1.4 row_span

```
int pssp::structs::Grid::row_span {0}
00025 {0};
```

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

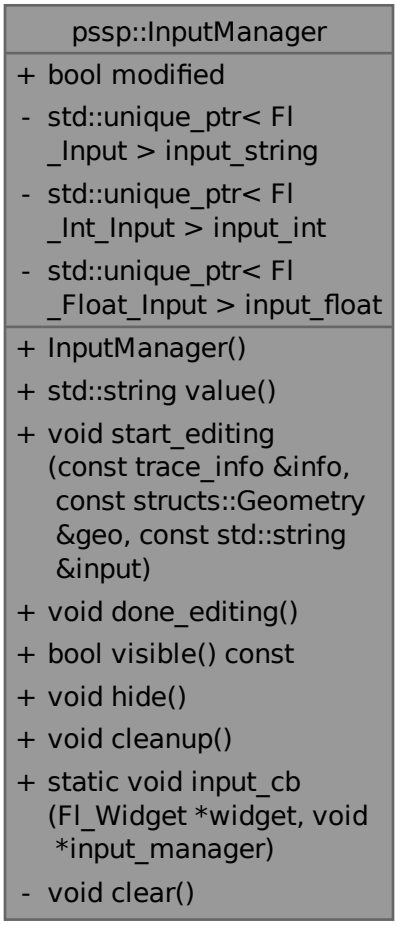
- include/PsSp/Utility/Structs.hpp

7.9 pssp::InputManager Class Reference

Manager of user-input.

```
#include <InputManager.hpp>
```

Collaboration diagram for pssp::InputManager:



Public Member Functions

- InputManager ()
- std::string value ()
- void start_editing (const trace_info &info, const structs::Geometry &geo, const std::string &input)
- void done_editing ()
- bool visible () const
- void hide ()
- void cleanup ()

Static Public Member Functions

- static void input_cb (Fl_Widget ∗widget, void ∗input_manager)

Public Attributes

- bool modified {false}

Private Member Functions

- void clear ()

Private Attributes

- std::unique_ptr< Fl_Input > input_string {}
- std::unique_ptr< Fl_Int_Input > input_int {}
- std::unique_ptr< Fl_Float_Input > input_float {}

7.9.1 Detailed Description

Manager of user-input.

This class handles taking input from the user (text/numerical) that is destined to enter the Datasheet spreadsheet display (and the underlying data-arrays).

It is designed to handle generic string input, integer input, and float input.

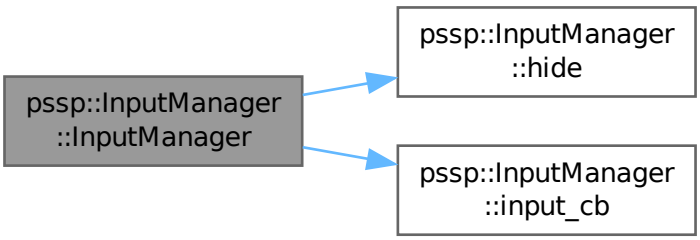
Todo

7.9.2 Constructor & Destructor Documentation

7.9.2.1 InputManager()

```
pssp::InputManager::InputManager ( )
00006                               {
00007   input_string = std::make_unique<Fl_Input>(0, 0, 0, 0);
00008   input_int = std::make_unique<Fl_Int_Input>(0, 0, 0, 0);
00009   input_float = std::make_unique<Fl_Float_Input>(0, 0, 0, 0);
00010   hide();
00011   input_string->callback(input_cb, reinterpret_cast<void *>(this));
00012   input_int->callback(input_cb, reinterpret_cast<void *>(this));
00013   input_float->callback(input_cb, reinterpret_cast<void *>(this));
00014   input_string->when(FL_WHEN_ENTER_KEY_ALWAYS);
00015   input_int->when(FL_WHEN_ENTER_KEY_ALWAYS);
00016   input_float->when(FL_WHEN_ENTER_KEY_ALWAYS);
00017   input_string->maximum_size(40);
00018   input_int->maximum_size(40);
00019   input_float->maximum_size(40);
00020   input_string->color(FL_YELLOW);
00021   input_int->color(FL_RED);
00022   input_float->color(FL_GREEN);
00023 }
```

Here is the call graph for this function:

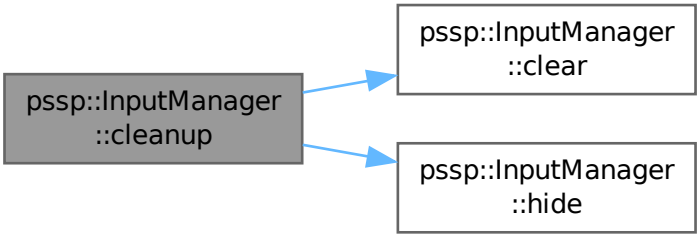


7.9.3 Member Function Documentation

7.9.3.1 cleanup()

```
void pssp::InputManager::cleanup ( )
00030                              {
00031   clear();
00032   hide();
00033 }
```

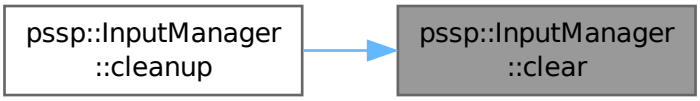
Here is the call graph for this function:



7.9.3.2 clear()

```
void pssp::InputManager::clear ( )  [private]
00041                              {
00042   input_string->value("");
00043   input_int->value("");
00044   input_float->value("");
00045 }
```

Here is the caller graph for this function:



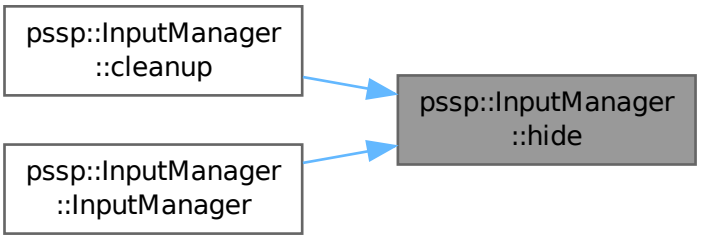
7.9.3.3 done_editing()

```
void pssp::InputManager::done_editing ( )
```

7.9.3.4 hide()

```
void pssp::InputManager::hide ( )
00035                         {
00036   input_string->hide();
00037   input_int->hide();
00038   input_float->hide();
00039 }
```

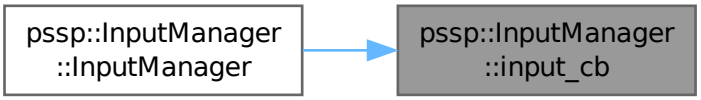
Here is the caller graph for this function:



7.9.3.5 input_cb()

```
static void pssp::InputManager::input_cb (
            Fl_Widget * widget,
            void * input_manager )  [inline], [static]
00053                                                                {
00054     (void)widget;
00055     reinterpret_cast<InputManager *>(input_manager)->modified = true;
00056   }
```

Here is the caller graph for this function:



7.9.3.6 start_editing()

```
void pssp::InputManager::start_editing (
            const trace_info & info,
```

```
            const structs::Geometry & geo,
            const std::string & input )
00063                                                       {
00064   if (info.type == Type::string_) {
00065     input_string->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00066     input_string->value(input.c_str());
00067     input_string->insert_position(0, static_cast<int>(input.size()));
00068     input_string->show();
00069     input_string->take_focus();
00070   } else if (info.type == Type::int_) {
00071     input_int->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00072     input_int->value(input.c_str());
00073     input_int->insert_position(0, static_cast<int>(input.size()));
00074     input_int->show();
00075     input_int->take_focus();
00076   } else if (info.type == Type::float_) {
00077     input_float->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00078     input_float->value(input.c_str());
00079     input_float->insert_position(0, static_cast<int>(input.size()));
00080     input_float->show();
00081     input_float->take_focus();
00082   } else if (info.type == Type::double_) {
00083     input_float->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00084     input_float->value(input.c_str());
00085     input_float->insert_position(0, static_cast<int>(input.size()));
00086     input_float->show();
00087     input_float->take_focus();
00088   } else if (info.type == Type::bool_) {
00089     input_string->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00090     input_string->value(input.c_str());
00091     input_string->insert_position(0, static_cast<int>(input.size()));
00092     input_string->show();
00093     input_string->take_focus();
00094   }
00095 }
```

7.9.3.7 value()

```
std::string pssp::InputManager::value ( )
00047                                 {
00048   std::string result{};
00049   // Which one is being used? They're empty after cleanup
00050   // so only the used one is full
00051   if (!std::string(input_string->value()).empty()) {
00052     result = input_string->value();
00053   } else if (!std::string(input_int->value()).empty()) {
00054     result = input_int->value();
00055   } else {
00056     result = input_float->value();
00057   }
00058   return result;
00059 }
```

7.9.3.8 visible()

```
bool pssp::InputManager::visible ( ) const
00025                                    {
00026   return ((input_string->visible() != 0) || (input_int->visible() != 0) ||
00027           (input_float->visible() != 0));
00028 }
```

7.9.4 Member Data Documentation

7.9.4.1 input_float

```
std::unique_ptr<Fl_Float_Input> pssp::InputManager::input_float {}  [private]
00068 {};
```

7.9.4.2 input_int

```
std::unique_ptr<Fl_Int_Input> pssp::InputManager::input_int {}  [private]
00067 {};
```

7.9.4.3 input_string

```
std::unique_ptr<Fl_Input> pssp::InputManager::input_string {}  [private]
00066 {};
```

7.9.4.4 modified

```
bool pssp::InputManager::modified {false}
00062 {false};
```

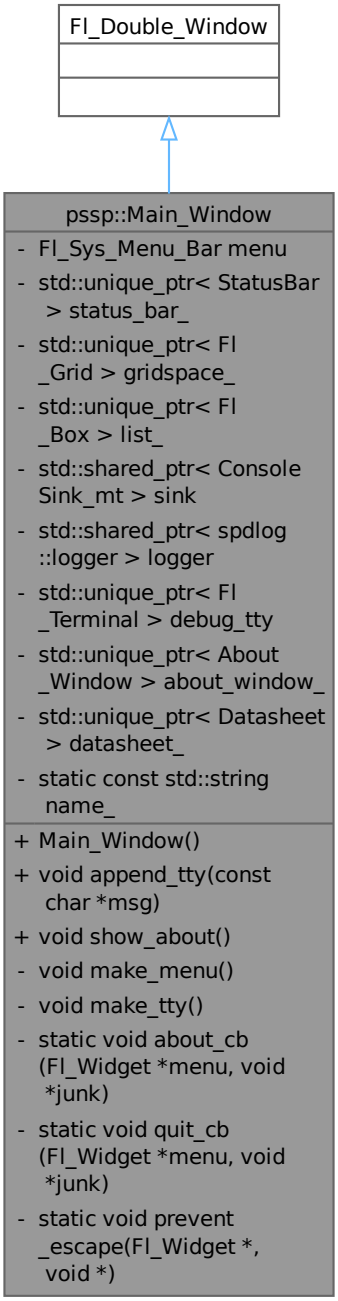
The documentation for this class was generated from the following files:

- include/PsSp/Managers/InputManager.hpp
- src/Managers/InputManager.cpp

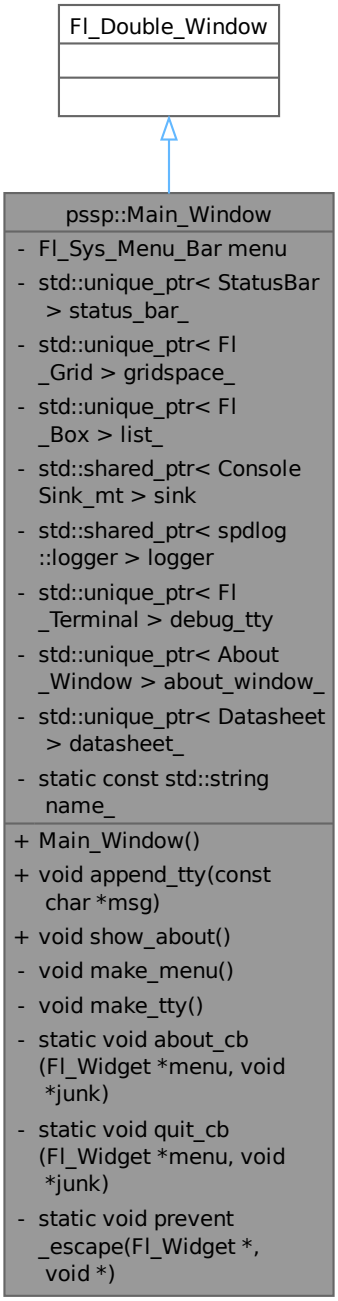
7.10 pssp::Main_Window Class Reference

```
#include <Main.hpp>
```

Inheritance diagram for pssp::Main_Window:



Collaboration diagram for pssp::Main_Window:



Public Member Functions

- Main_Window ()
- void append_tty (const char ∗msg)
- void show_about ()

Private Member Functions

- void make_menu ()
- void make_tty ()

Static Private Member Functions

- static void about_cb (Fl_Widget *menu, void *junk)
- static void quit_cb (Fl_Widget *menu, void *junk)
- static void prevent_escape (Fl_Widget *, void *)

Private Attributes

- Fl_Sys_Menu_Bar menu {0, 0, 0, mw::menu_height, nullptr}
- std::unique_ptr< StatusBar > status_bar_ {}
- std::unique_ptr< Fl_Grid > gridspace_ {}
- std::unique_ptr< Fl_Box > list_ {}
- std::shared_ptr< ConsoleSink_mt > sink {}
- std::shared_ptr< spdlog::logger > logger {}
- std::unique_ptr< Fl_Terminal > debug_tty {}
- std::unique_ptr< About_Window > about_window_ {}
- std::unique_ptr< Datasheet > datasheet_ {}

Static Private Attributes

- static const std::string name_ {"PsSp - Passive-source Seismic-processing"}

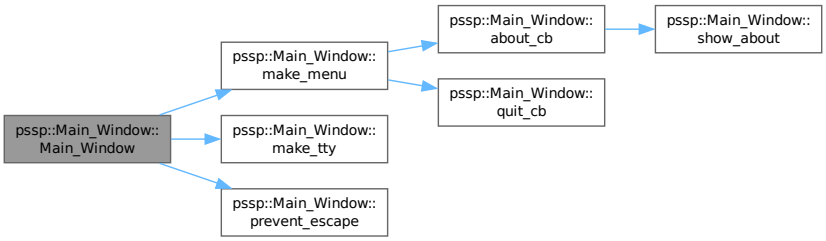
7.10.1 Constructor & Destructor Documentation

7.10.1.1 Main_Window()

```
pssp::Main_Window::Main_Window ( )
00006                         : Fl_Double_Window(0, 0, name_.c_str()) {
00007   this->callback(prevent_escape);
00008   make_tty();
00009   spdlog::trace("Building \033[1mMain_Window\033[0m.");
00010   this->begin();
00011   resizable(this);
00012   // Minimum window size width/height
00013   this->size_range(mw::minimum_x, mw::minimum_y);
00014   structs::Geometry geo{};
00015   Fl::screen_work_area(geo.x_pos, geo.y_pos, geo.width, geo.height);
00016   this->resize(geo.x_pos, geo.y_pos, geo.width, geo.height);
00017   make_menu();
00018   menu.resize(0, 0, geo.width, menu.h());
00019   status_bar_ = std::make_unique<StatusBar>(this->h(), this->w(), menu.h());
00020 #if defined(__APPLE__)
00021   const int menu_shift{0};
00022 #else
00023   const int menu_shift{menu.h()};
00024 #endif
00025   gridspace_ = std::make_unique<Fl_Grid>(0, menu_shift, this->w(),
00026                                          this->h() - menu_shift - menu.h());
00027   gridspace_->begin();
00028   gridspace_->add(debug_tty.get());
00029   gridspace_->show_grid(0);  // 1 to show guide lines
00030   constexpr structs::Grid layout{10, 10, 1, 1};
00031   gridspace_->layout(layout.row, layout.col, layout.row_span, layout.col_span);
00032   list_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "List");
00033   list_->box(FL_BORDER_BOX);
00034   list_->color(FL_WHITE);
00035   datasheet_ = std::make_unique<Datasheet>();
```

```
00036   constexpr structs::Grid tty_grid{7, 0, 3, 10};
00037   gridspace_->widget(debug_tty.get(), tty_grid.row, tty_grid.col,
00038                      tty_grid.row_span, tty_grid.col_span);
00039   constexpr structs::Grid list_grid{0, 0, 7, 2};
00040   gridspace_->widget(list_.get(), list_grid.row, list_grid.col,
00041                      list_grid.row_span, list_grid.col_span);
00042   constexpr structs::Grid ds_grid{0, 2, 7, 8};
00043   gridspace_->widget(datasheet_.get(), ds_grid.row, ds_grid.col,
00044                      ds_grid.row_span, ds_grid.col_span);
00045   gridspace_->end();
00046   this->end();
00047   this->resizable(status_bar_.get());
00048   this->resizable(datasheet_.get());
00049   this->resizable(gridspace_.get());
00050   about_window_ = std::make_unique<About_Window>();
00051   about_window_->hide();
00052   spdlog::trace("Done making \033[1mMain_Window\033[0m.");
00053 }
```

Here is the call graph for this function:

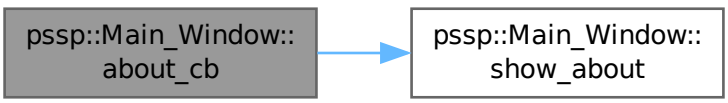


7.10.2 Member Function Documentation

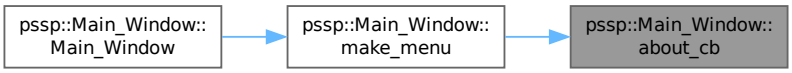
7.10.2.1 about_cb()

```
void pssp::Main_Window::about_cb (
            Fl_Widget * menu,
            void * junk )  [static], [private]
00139                                                   {
00140   (void)junk;
00141   auto *window = reinterpret_cast<Main_Window *>(menu->parent()->as_window());
00142   window->show_about();
00143 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



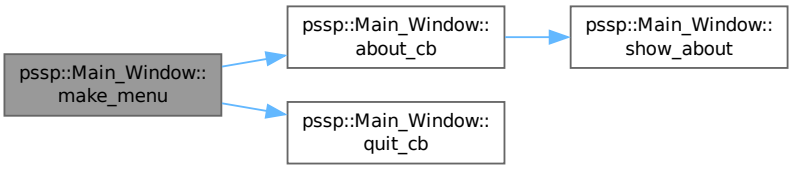
7.10.2.2 append_tty()

```
void pssp::Main_Window::append_tty (
            const char * msg )
00125 { debug_tty->append(msg); }
```

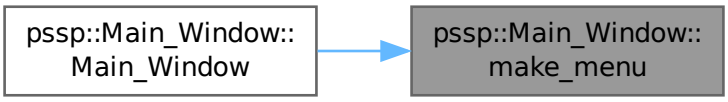
7.10.2.3 make_menu()

```
void pssp::Main_Window::make_menu ( )  [private]
00076                                {
00077   spdlog::trace("Making \033[1mMenu\033[0m.");
00078   // Program
00079   menu.add("&Program/&Quit", FL_COMMAND + 'q', quit_cb, this);
00080   // Project
00081   menu.add("&Project/&New", FL_COMMAND + 'n', nullptr, this, FL_MENU_INACTIVE);
00082   menu.add("&Project/&Load", FL_COMMAND + 'o', nullptr, this, FL_MENU_INACTIVE);
00083   menu.add("&Project/&Close", FL_COMMAND + 'c', nullptr, this,
00084            FL_MENU_INACTIVE);
00085   menu.add("&Project/&Bookmark", FL_COMMAND + 'b', nullptr, this,
00086            FL_MENU_INACTIVE);
00087   // Data
00088   menu.add("&Data/&Add File", 0, nullptr, this, FL_MENU_INACTIVE);
00089   menu.add("&Data/&Add Directory", 0, nullptr, this, FL_MENU_INACTIVE);
00090   menu.add("&Data/&Download Data", 0, nullptr, this, FL_MENU_INACTIVE);
00091   // Processing
00092   menu.add("&Processing/&Filters/&Butterworth/&Lowpass", 0, nullptr, this,
00093            FL_MENU_INACTIVE);
00094   menu.add("&Processing/&Filters/&Butterworth/&Highpass", 0, nullptr, this,
00095            FL_MENU_INACTIVE);
00096   menu.add("&Processing/&Filters/&Butterworth/&Bandpass", 0, nullptr, this,
00097            FL_MENU_INACTIVE);
00098   // Plotting
00099   menu.add("&Plot/&Single Component/&Time-series", 0, nullptr, this,
00100            FL_MENU_INACTIVE);
00101   menu.add("&Plot/&Single Component/&Spectrum/&Real-Imaginary", 0, nullptr,
00102            this, FL_MENU_INACTIVE);
00103   menu.add("&Plot/&Single Component/&Spectrum/&Amplitude-Phase", 0, nullptr,
00104            this, FL_MENU_INACTIVE);
00105   menu.add("&Plot/&Single Component/&Spectrogram", 0, nullptr, this,
00106            FL_MENU_INACTIVE);
00107   menu.add("&Plot/&Three Component/&Time-series", 0, nullptr, this,
00108            FL_MENU_INACTIVE);
00109   menu.add("&Plot/&Three Component/&Spectrum/&Real-Imaginary", 0, nullptr, this,
00110            FL_MENU_INACTIVE);
00111   menu.add("&Plot/&Three Component/&Spectrum/&Amplitude-Phase", 0, nullptr,
00112            this, FL_MENU_INACTIVE);
00113   menu.add("&Plot/&Three Component/&Spectrogram", 0, nullptr, this,
00114            FL_MENU_INACTIVE);
00115   menu.add("&Plot/&Profile", 0, nullptr, this, FL_MENU_INACTIVE);
00116   // Settings
00117   menu.add("&Settings", 0, nullptr, this, FL_MENU_INACTIVE);
00118   // Help
00119   menu.add("&Help", 0, nullptr, this, FL_MENU_INACTIVE);
00120   // About
00121   menu.add("&About", 0, about_cb, this);
00122   spdlog::trace("Done making \033[1mMenu\033[0m.");
00123 }
```

Here is the call graph for this function:



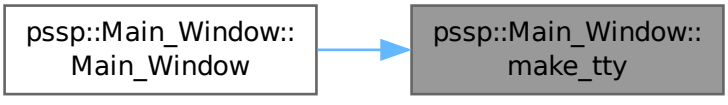
Here is the caller graph for this function:



7.10.2.4 make_tty()

```
void pssp::Main_Window::make_tty ( )  [private]
00055                           {
00056   // Debug terminal
00057   debug_tty = std::make_unique<Fl_Terminal>(0, 0, 0, 0);
00058   sink = std::make_shared<ConsoleSink_mt>(debug_tty.get());
00059   logger = std::make_shared<spdlog::logger>("tty logger", sink);
00060   spdlog::set_default_logger(logger);
00061   // levels are critical, error, warn, info, debug, trace
00062   spdlog::set_level(spdlog::level::trace);
00063   spdlog::set_pattern(
00064       "\33[1m\33[32m[%Y-%m-%d %T]\33[33m[%l]\33[36m[thread %t]\33[0m %v");
00065   debug_tty->begin();
00066   constexpr int font_size{14};
00067   debug_tty->textsize(font_size);
00068   debug_tty->redraw_style(Fl_Terminal::NO_REDRAW);
00069   constexpr int num_columns{80};
00070   debug_tty->display_columns(num_columns);
00071   spdlog::trace("Logger started.");
00072   debug_tty->end();
00073   resizable();
00074 }
```

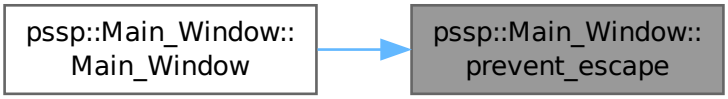
Here is the caller graph for this function:



7.10.2.5 prevent_escape()

```
void pssp::Main_Window::prevent_escape (
            Fl_Widget * caller,
            void * data )  [static], [private]
00146                                                                {
00147   (void)caller;
00148   (void)data;
00149   if ((Fl::event() == FL_SHORTCUT) && (Fl::event_key() == FL_Escape)) {
00150     return;  // ignore Escape
00151   }
00152   exit(0);
00153 }
```

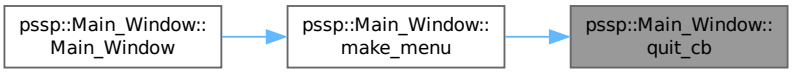
Here is the caller graph for this function:



7.10.2.6 quit_cb()

```
void pssp::Main_Window::quit_cb (
            Fl_Widget * menu,
            void * junk )  [static], [private]
00127                                                  {
00128   (void)junk;
00129   // reinterpret_cast is unnecessary, but I wanted to figure it out
00130   auto *window = reinterpret_cast<Main_Window *>(menu->parent()->as_window());
00131   if (fl_choice("Are you sure you want to quit?", "cancel", "quit", nullptr) !=
00132       0) {
00133     window->hide();
00134   }
00135 }
```

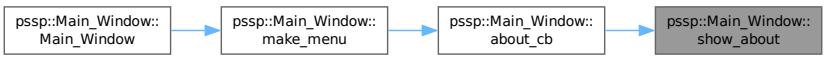
Here is the caller graph for this function:



7.10.2.7 show_about()

```
void pssp::Main_Window::show_about ( )
00137 { about_window_->show(); }
```

Here is the caller graph for this function:



7.10.3 Member Data Documentation

7.10.3.1 about_window_

```
std::unique_ptr<About_Window> pssp::Main_Window::about_window_ {}  [private]
00050 {};
```

7.10.3.2 datasheet_

```
std::unique_ptr<Datasheet> pssp::Main_Window::datasheet_ {}  [private]
00051 {};
```

7.10.3.3 debug_tty

```
std::unique_ptr<Fl_Terminal> pssp::Main_Window::debug_tty {}  [private]
00049 {};
```

7.10.3.4 gridspace_

```
std::unique_ptr<Fl_Grid> pssp::Main_Window::gridspace_ {}  [private]
00045 {};
```

7.10.3.5 list_

```
std::unique_ptr<Fl_Box> pssp::Main_Window::list_ {}  [private]
00046 {};
```

7.10.3.6 logger

```
std::shared_ptr<spdlog::logger> pssp::Main_Window::logger {}  [private]
00048 {};
```

7.10.3.7 menu

```
Fl_Sys_Menu_Bar pssp::Main_Window::menu {0, 0, 0, mw::menu_height, nullptr}  [private]
00041 {0, 0, 0, mw::menu_height, nullptr};
```

7.10.3.8 name_

```
const std::string pssp::Main_Window::name_ {"PsSp - Passive-source Seismic-processing"}  [inline],
[static], [private]
00056 {"PsSp - Passive-source Seismic-processing"};
```

7.10.3.9 sink

```
std::shared_ptr<ConsoleSink_mt> pssp::Main_Window::sink {}  [private]
00047 {};
```

7.10.3.10 status_bar_

```
std::unique_ptr<StatusBar> pssp::Main_Window::status_bar_ {}  [private]
00044 {};
```

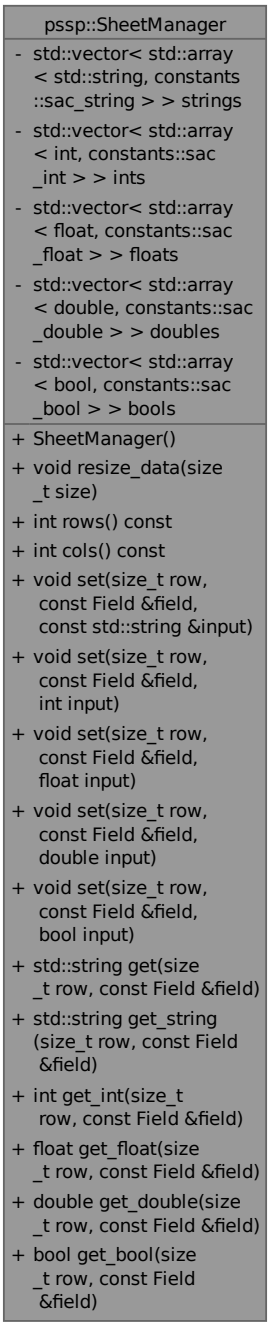
The documentation for this class was generated from the following files:

- include/PsSp/Windows/Main.hpp
- src/Windows/Main.cpp

7.11 pssp::SheetManager Class Reference

```
#include <SheetManager.hpp>
```

Collaboration diagram for pssp::SheetManager:



Public Member Functions

- SheetManager ()
- void resize_data (size_t size)
- int rows () const
- int cols () const
- void set (size_t row, const Field &field, const std::string &input)

- void set (size_t row, const Field &field, int input)
- void set (size_t row, const Field &field, float input)
- void set (size_t row, const Field &field, double input)
- void set (size_t row, const Field &field, bool input)
- std::string get (size_t row, const Field &field)
- std::string get_string (size_t row, const Field &field)
- int get_int (size_t row, const Field &field)
- float get_float (size_t row, const Field &field)
- double get_double (size_t row, const Field &field)
- bool get_bool (size_t row, const Field &field)

Private Attributes

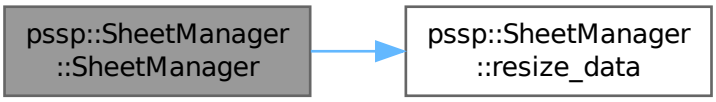
- std::vector< std::array< std::string, constants::sac_string > > strings {}
- std::vector< std::array< int, constants::sac_int > > ints {}
- std::vector< std::array< float, constants::sac_float > > floats {}
- std::vector< std::array< double, constants::sac_double > > doubles {}
- std::vector< std::array< bool, constants::sac_bool > > bools {}

7.11.1 Constructor & Destructor Documentation

7.11.1.1 SheetManager()

```
pssp::SheetManager::SheetManager ( )
00006 { resize_data(100); }
```

Here is the call graph for this function:



7.11.2 Member Function Documentation

7.11.2.1 cols()

```
int pssp::SheetManager::cols ( ) const
00018                          {
00019   size_t num_cols{strings[0].size() + ints[0].size() + floats[0].size() +
00020                   doubles[0].size() + bools[0].size()};
00021   return static_cast<int>(num_cols);
00022 }
```

7.11.2.2 get()

```
std::string pssp::SheetManager::get (
              size_t row,
              const Field & field )
00077                                                                 {
00078   std::string result{};
00079   const trace_info &info{field_info.at(field)};
00080   switch (info.type) {
00081   case Type::string_:
00082     break;
00083   case Type::int_:
00084     result = std::to_string(ints[row][info.array_col]);
00085     break;
00086   case Type::float_:
00087     result = std::to_string(floats[row][info.array_col]);
00088     break;
00089   case Type::double_:
00090     result = std::to_string(doubles[row][info.array_col]);
00091     break;
00092   case Type::bool_:
00093     result = std::to_string(static_cast<int>(bools[row][info.array_col]));
00094     break;
00095   default:
00096     break;
00097   }
00098   return result;
00099 }
```

7.11.2.3 get_bool()

```
bool pssp::SheetManager::get_bool (
              size_t row,
              const Field & field )
00149                                                                 {
00150   bool result{};
00151   const trace_info &info{field_info.at(field)};
00152   if (info.type == Type::bool_) {
00153     result = bools[row][info.array_col];
00154   } else {
00155     spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00156                   type_names.at(info.type));
00157   }
00158   return result;
00159 }
```

7.11.2.4 get_double()

```
double pssp::SheetManager::get_double (
              size_t row,
              const Field & field )
00137                                                                   {
00138   double result{};
00139   const trace_info &info{field_info.at(field)};
00140   if (info.type == Type::double_) {
00141     result = doubles[row][info.array_col];
00142   } else {
00143     spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00144                   type_names.at(info.type));
00145   }
00146   return result;
00147 }
```

7.11.2.5 get_float()

```
float pssp::SheetManager::get_float (
              size_t row,
              const Field & field )
```

```
00125                                                                    {
00126   float result{};
00127   const trace_info &info{field_info.at(field)};
00128   if (info.type == Type::float_) {
00129     result = floats[row][info.array_col];
00130   } else {
00131     spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00132                   type_names.at(info.type));
00133   }
00134   return result;
00135 }
```

7.11.2.6 get_int()

```
int pssp::SheetManager::get_int (
              size_t row,
              const Field & field )
00113                                                                {
00114   int result{};
00115   const trace_info &info{field_info.at(field)};
00116   if (info.type == Type::int_) {
00117     result = ints[row][info.array_col];
00118   } else {
00119     spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00120                   type_names.at(info.type));
00121   }
00122   return result;
00123 }
```

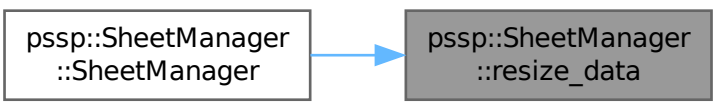
7.11.2.7 get_string()

```
std::string pssp::SheetManager::get_string (
              size_t row,
              const Field & field )
00101                                                                    {
00102   std::string result{};
00103   const trace_info &info{field_info.at(field)};
00104   if (info.type == Type::string_) {
00105     result = strings[row][info.array_col];
00106   } else {
00107     spdlog::error("Field {0} wrong type {1} for get_string.", info.name,
00108                   type_names.at(info.type));
00109   }
00110   return result;
00111 }
```

7.11.2.8 resize_data()

```
void pssp::SheetManager::resize_data (
              size_t size )
00008                                                {
00009   strings.resize(size);
00010   ints.resize(size);
00011   floats.resize(size);
00012   doubles.resize(size);
00013   bools.resize(size);
00014 }
```

Here is the caller graph for this function:



7.11.2.9 rows()

```
int pssp::SheetManager::rows ( ) const
00016 { return static_cast<int>(bools.size()); }
```

7.11.2.10 set() [1/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            bool input )
00067                                                                    {
00068   const trace_info &info{field_info.at(field)};
00069   if (info.type == Type::bool_) {
00070     bools[row][info.array_col] = input;
00071   } else {
00072     spdlog::error("Wrong type {0} inserted into field {1}.",
00073                   type_names.at(info.type), info.name);
00074   }
00075 }
```

7.11.2.11 set() [2/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            const std::string & input )
00025                                                {
00026   const trace_info &info{field_info.at(field)};
00027   if (info.type == Type::string_) {
00028     strings[row][info.array_col] = input;
00029   } else {
00030     spdlog::error("Wrong type {0} inserted into field {1}.",
00031                   type_names.at(info.type), info.name);
00032   }
00033 }
```

7.11.2.12 set() [3/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            double input )
00057                                                {
00058   const trace_info &info{field_info.at(field)};
00059   if (info.type == Type::double_) {
00060     doubles[row][info.array_col] = input;
00061   } else {
00062     spdlog::error("Wrong type {0} inserted into field {1}.",
00063                   type_names.at(info.type), info.name);
00064   }
00065 }
```

7.11.2.13 set() [4/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            float input )
00046                                               {
00047   const trace_info &info{field_info.at(field)};
00048   if (info.type == Type::float_) {
00049     floats[row][info.array_col] = input;
00050   } else {
00051     spdlog::error("Wrong type {0} inserted into field {1}.",
00052                   type_names.at(info.type), info.name);
00053   }
00054 }
```

7.11.2.14 set() [5/5]

```
void pssp::SheetManager::set (
            size_t row,
            const Field & field,
            int input )
00035                                                                    {
00036   const trace_info &info{field_info.at(field)};
00037   if (info.type == Type::int_) {
00038     ints[row][info.array_col] = input;
00039   } else {
00040     spdlog::error("Wrong type {0} inserted into field {1}.",
00041                   type_names.at(info.type), info.name);
00042   }
00043 }
```

7.11.3 Member Data Documentation

7.11.3.1 bools

```
std::vector<std::array<bool, constants::sac_bool> > pssp::SheetManager::bools {}  [private]
00057 {};
```

7.11.3.2 doubles

```
std::vector<std::array<double, constants::sac_double> > pssp::SheetManager::doubles {}  [private]
00055 {};
```

7.11.3.3 floats

```
std::vector<std::array<float, constants::sac_float> > pssp::SheetManager::floats {}  [private]
00053 {};
```

7.11.3.4 ints

```
std::vector<std::array<int, constants::sac_int> > pssp::SheetManager::ints {}  [private]
00051 {};
```

7.11.3.5 strings

```
std::vector<std::array<std::string, constants::sac_string> > pssp::SheetManager::strings {}
[private]
00049 {};
```

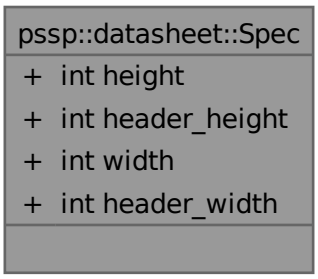
The documentation for this class was generated from the following files:

- include/PsSp/Managers/SheetManager.hpp
- src/Managers/SheetManager.cpp

7.12 pssp::datasheet::Spec Struct Reference

`#include <Datasheet.hpp>`

Collaboration diagram for pssp::datasheet::Spec:



Public Attributes

- int height {0}
- int header_height {0}
- int width {0}
- int header_width {0}

7.12.1 Member Data Documentation

7.12.1.1 header_height

```
int pssp::datasheet::Spec::header_height {0}
00042 {0};
```

7.12.1.2 header_width

```
int pssp::datasheet::Spec::header_width {0}
00046 {0};
```

7.12.1.3 height

```
int pssp::datasheet::Spec::height {0}
00040 {0};
```

7.12.1.4 width

```
int pssp::datasheet::Spec::width {0}
00044 {0};
```

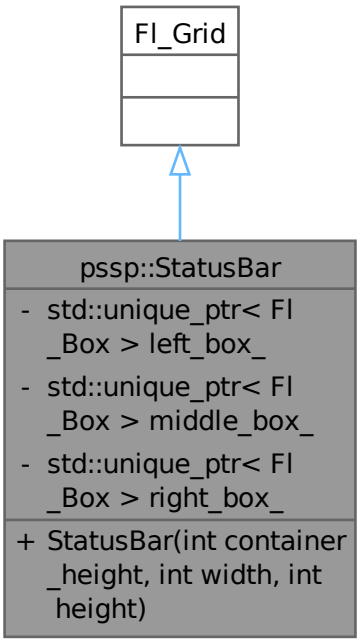
The documentation for this struct was generated from the following file:

- include/PsSp/Widgets/Datasheet.hpp

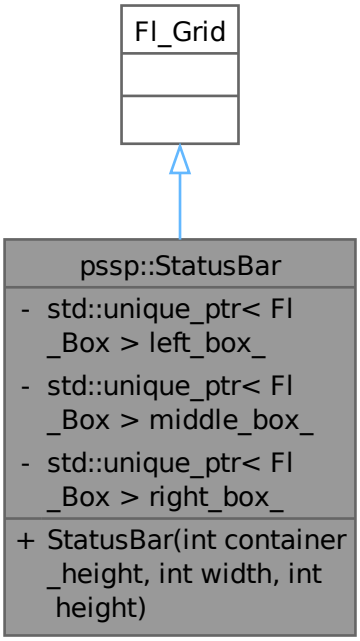
7.13 pssp::StatusBar Class Reference

```
#include <StatusBar.hpp>
```

Inheritance diagram for pssp::StatusBar:



Collaboration diagram for pssp::StatusBar:



Public Member Functions

- StatusBar (int container_height, int width, int height)

Private Attributes

- std::unique_ptr< Fl_Box > left_box_ {}
- std::unique_ptr< Fl_Box > middle_box_ {}
- std::unique_ptr< Fl_Box > right_box_ {}

7.13.1 Constructor & Destructor Documentation

7.13.1.1 StatusBar()

```
pssp::StatusBar::StatusBar (
            int container_height,
            int width,
            int height )
00007     : Fl_Grid(0, container_height - height, width, height) {
00008   spdlog::trace("Making \033[1mStatus_Bar\033[0m.");
00009   this->begin();
00010   constexpr structs::Grid layout{1, 10, 1, 1};
00011   this->layout(layout.row, layout.col, layout.row_span, layout.col_span);
00012   left_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Left Box");
00013   left_box_->box(FL_BORDER_BOX);
```

```
00014   constexpr structs::Grid left{0, 0, 1, 2};
00015   this->widget(left_box_.get(), left.row, left.col, left.row_span,
00016                left.col_span);
00017   middle_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Middle Box");
00018   middle_box_->box(FL_BORDER_BOX);
00019   constexpr structs::Grid middle{0, 2, 1, 6};
00020   this->widget(middle_box_.get(), middle.row, middle.col, middle.row_span,
00021                middle.col_span);
00022   right_box_ = std::make_unique<Fl_Box>(0, 0, 0, 0, "Right Box");
00023   right_box_->box(FL_BORDER_BOX);
00024   constexpr structs::Grid right{0, 8, 1, 2};
00025   this->widget(right_box_.get(), right.row, right.col, right.row_span,
00026                right.col_span);
00027   this->end();
00028   spdlog::trace("Done making \033[1mStatus_Bar\033[0m.");
00029 }
```

7.13.2 Member Data Documentation

7.13.2.1 left_box_

```
std::unique_ptr<Fl_Box> pssp::StatusBar::left_box_ {}  [private]
00024 {};
```

7.13.2.2 middle_box_

```
std::unique_ptr<Fl_Box> pssp::StatusBar::middle_box_ {}  [private]
00025 {};
```

7.13.2.3 right_box_

```
std::unique_ptr<Fl_Box> pssp::StatusBar::right_box_ {}  [private]
00026 {};
```

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

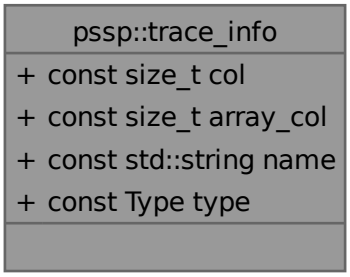
- include/PsSp/Widgets/StatusBar.hpp
- src/Widgets/StatusBar.cpp

7.14 pssp::trace_info Struct Reference

Information for.

`#include <Enums.hpp>`

Collaboration diagram for pssp::trace_info:



Public Attributes

- const size_t col {0}
- const size_t array_col {0}
- const std::string name {}

  Derived from type_names.
- const Type type {}

7.14.1 Detailed Description

Information for.

7.14.2 Member Data Documentation

7.14.2.1 array_col

```
const size_t pssp::trace_info::array_col {0}
00060 {0};
```

7.14.2.2 col

```
const size_t pssp::trace_info::col {0}
00058 {0};
```

7.14.2.3 name

```
const std::string pssp::trace_info::name {}
```

Derived from type_names.
```
00062 {};
```

7.14.2.4 type

```
const Type pssp::trace_info::type {}
00064 {};
```

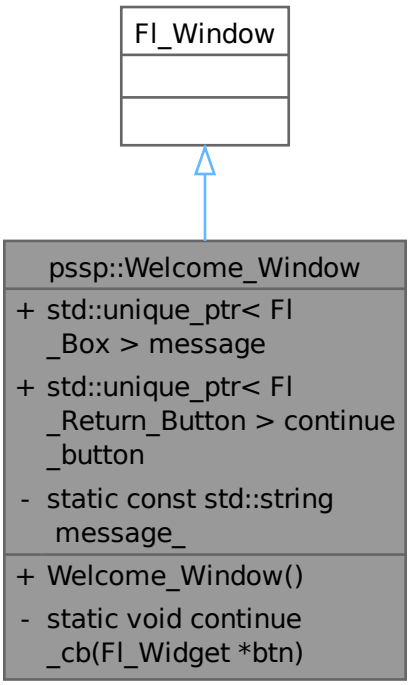
The documentation for this struct was generated from the following file:

- include/PsSp/Utility/Enums.hpp

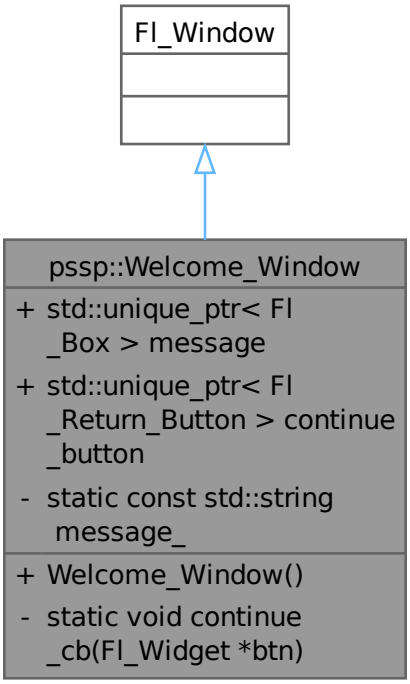
7.15 pssp::Welcome_Window Class Reference

`#include <Welcome.hpp>`

Inheritance diagram for pssp::Welcome_Window:



Collaboration diagram for pssp::Welcome_Window:



Public Member Functions

- Welcome_Window ()

Public Attributes

- std::unique_ptr< Fl_Box > message {}
- std::unique_ptr< Fl_Return_Button > continue_button {}

Static Private Member Functions

- static void continue_cb (Fl_Widget *btn)

Static Private Attributes

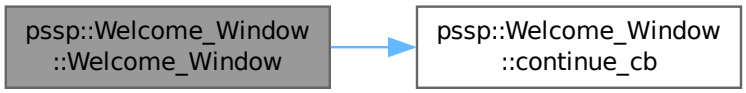
- static const std::string message_

7.15.1 Constructor & Destructor Documentation

7.15.1.1 Welcome_Window()

```
pssp::Welcome_Window::Welcome_Window ( )
00006                                : Fl_Window(0, 0, 0, 0, "Welcome!") {
00007   this->begin();
00008   int x_start{};
00009   int y_start{};
00010   int width{};
00011   int height{};
00012   Fl::screen_work_area(x_start, y_start, width, height);
00013   x_start = ((width - welcome::width) / 2);
00014   y_start = ((height - welcome::height) / 2);
00015   this->resize(x_start, y_start, welcome::width, welcome::height);
00016   this->box(FL_BORDER_BOX);
00017   set_modal();
00018   message =
00019       std::make_unique<Fl_Box>((welcome::width - welcome::text_width) / 2, 0,
00020                                welcome::text_width, welcome::text_height);
00021   continue_button = std::make_unique<Fl_Return_Button>(
00022       (welcome::width - welcome::button_width) / 2, welcome::text_height,
00023       welcome::button_width, welcome::button_height, "Continue");
00024   message->label(message_.c_str());
00025   message->align(FL_ALIGN_CENTER);
00026   continue_button->callback(continue_cb);
00027   this->end();
00028 }
```

Here is the call graph for this function:

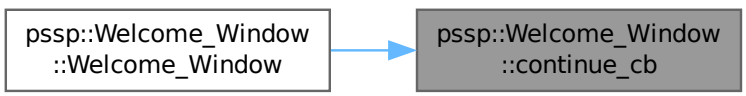


7.15.2 Member Function Documentation

7.15.2.1 continue_cb()

```
void pssp::Welcome_Window::continue_cb (
             Fl_Widget * btn )  [static], [private]
00030 { btn->parent()->hide(); }
```

Here is the caller graph for this function:



7.15.3 Member Data Documentation

7.15.3.1 continue_button

```
std::unique_ptr<Fl_Return_Button> pssp::Welcome_Window::continue_button {}
00031 {};
```

7.15.3.2 message

```
std::unique_ptr<Fl_Box> pssp::Welcome_Window::message {}
00030 {};
```

7.15.3.3 message_

```
const std::string pssp::Welcome_Window::message_  [inline], [static], [private]
```

Initial value:

```
{"Welcome to Passive-source Seismic-processing (PsSp)!\n"
              "This program is very early in development..."}
00037              {"Welcome to Passive-source Seismic-processing (PsSp)!\n"
00038               "This program is very early in development..."};
```

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

- include/PsSp/Windows/Welcome.hpp
- src/Windows/Welcome.cpp

Index