WASP AT - Wasp Analysis Tools

Graphical Data Processor (GDP)
Routine Descriptions

Developed by:

Integrated Decision Support Group
The Water Center
Colorado State University
Fort Collins, Colorado

Developed for:

U.S Department of the Interior
Bureau of Reclamation
Technical Services Center
Denver, Colorado
# Table of Contents

## 1.0 Routine List by Source File
- AggNameChoiceDlg.h - 1
- AggregationPage.h - 1
- DataAvailPage.h - 2
- DataEditPage.h - 3
- DeleteItemsDlg.h - 5
- ExportOptionsDlg.h - 5
- gdpdata.h - 5
- GDPDoc.h - 23
- GDPViewSheet.h - 23
- GraphDlg.h - 24
- ListDialog.h - 31
- ProjectInfoPage.h - 31
- SpreadsheetAddDlg.h - 32
- UnitsData.h - 33
- UnitsPage.h - 33

## 2.0 Function Index Arranged by Class
- 35

## 2.0 Member Index Arranged by Class
- 39
1.0 Routine List by Source File

1. Source File:  AggNameChoiceDlg.h

**Class:** AggNameChoiceDlg  
**Function:** AggNameChoiceDlg(CWnd* pParent = NULL) : Constructor  
**Description:** Pops up a text dialog that asks the user to input a name for the new station or parameter aggregate. Member 'm_name' will contain the new name.  
**Input Args:** pParent: the parent window.

2. Source File:  AggregationPage.h

**Class:** AggregationPage  
**Function:** AggregationPage() : Constructor  
**Description:** The station/parameter aggregation tab in the main view.

**Class:** AggregationPage  
**Function:** SetDoc(CGDPDoc *pDoc) : void  
**Description:** Pass the document to the page so that aggregation changes can be made directly.

**Class:** AggregationPage  
**Function:** InitializeListBoxes() : void  
**Description:** Build the station/parameter and aggregation list boxes.

**Class:** AggregationPage  
**Function:** UpdateAll() : void  
**Description:** Refresh the display, usually when new data has been added to the project.

**Class:** AggregationPage  
**Member:** m_aggListBoxL : AggListBox  
**Description:** Overrides CListBox so that a third state (disabled) can be applied to each list element.

**Class:** AggregationPage  
**Member:** m_allObjectList : CStringList  
**Description:** The list of all objects being considered, either stations or parameters.

**Class:** AggregationPage  
**Member:** m_allObjectList : CList<BOOL, BOOL>  
**Description:** The list corresponding to m_allObjectList indicating whether the object has been aggregated.
Class: AggregationPage
Member: m_allObjectList : CStringList
Description: The list of aggregated objects. These will be a subset of m_allObjectList.

Class: AggregationPage
Member: m_aggObjectAliasList : CStringList
Description: The list corresponding to m_allObjectList of each aggregated object's alias.

Class: AggregationPage
Member: m_objectIndexMap : CMap<CString, LPCSTR, int, int>
Description: A map that returns the index of the object given its name.

3. Source File: DataAvailPage.h

Class: DataAvailPage
Function: DataAvailPage() : Constructor
Description: Constructor for the Data Availability tab.

Class: DataAvailPage
Function: SetDoc(CGDPDoc *pDoc) : void
Description: Passes a pointer to the document so that the page can directly access project data.

Class: DataAvailPage
Function: UpdateAll() : void
Description: Refresh the display, usually after the project data has been changed.

Class: DataAvailPage
Function: InitializeDateControls() : void
Description: Initialize the date combo boxes with the start and end dates that fully bound all the project data.

Class: DataAvailPage
Function: BuildDayCombos() : void
Description: Sets the number of days in the day combos to have the right number of days for the given month and year.

Class: DataAvailPage
Function: InitializeListBoxes() : void
Description: Updates the list box labels according to whether the stations or parameters are the main display (i.e. if m_listMode == 0 or 1). Initializes the left (main) list box.

Class: DataAvailPage
Function: BuildRightListBox() : void
Description: Initializes the right list box with parameters that are available for the selected stations in the left list box, or visa-versa if parameters are displayed in the left list box.
1.0 Routine List by Source File

**Class:** DataAvailPage  
**Function:** BuildReportTable() : void  
**Description:** Adds the appropriate number of columns and labels them according to the date spacing and units (m_intervalCount and m_plotIntervalMode).

**Class:** DataAvailPage  
**Function:** UpdateReportTable() : void  
**Description:** Fills in the report table based on the left and right list box selections. If m_doFillData is TRUE, then do simple linear interpolation to fill in missing data points (missing end points use the closest valid data point rather than interpolate from zero).

**Class:** DataAvailPage  
**Function:** GetStartDate() : IDSTime  
**Description:** Builds and returns an IDSTime object that represents the start day selected using the start date combo boxes.

**Class:** DataAvailPage  
**Function:** GetEndDate() : IDSTime  
**Description:** Builds and returns an IDSTime object that represents the end day selected using the end date combo boxes.

**Class:** DataAvailPage  
**Function:** SetGraphRadioSensitivity() : IDSTime  
**Description:** (De)Sensitizes the radio buttons based on user selections. If m_graphMode == 0 (multiple parameters are being plotted), then only data availability is available.

**Class:** DataAvailPage  
**Function:** CopyToClipboard() : CString  
**Description:** This is a WASP-specific function that copies the data for the selected parameter and station to the clipboard as a timeseries in days.

**Class:** DataAvailPage  
**Function:** UpdatePlotUI() : void  
**Description:** (De)Sensitizes the radio buttons based on user selections. If data availability is selected, then only a median plot is available.

**Class:** DataAvailPage  
**Member:** m_reportTable : TableListCtrl  
**Description:** The report table widget.

**Class:** DataAvailPage  
**Member:** m_aggObjectList : CStringList  
**Description:** The list of aggregated objects.

4. Source File: DataEditPage.h

**Class:** DataEditPage  
**Function:** DataEditPage() : Constructor  
**Description:** Constructor for the Data Edit page.
Class: DataEditPage
Function: SetDoc(CGDPDoc *pDoc) : void
Description: Passes a pointer to the document so that the page can directly access project data.

Class: DataEditPage
Function: UpdateAll() : void
Description: Refresh the display, usually after the project data has been changed.

Class: DataEditPage
Function: BuildMainTable() : void
Description: Generate column names appropriate to whether stations or parameters are selected (m_type == 0 or 1). (De)Sensitize the data table if parameter mode is selected or not.

Class: DataEditPage
Function: FillMainTable() : void
Description: Populate the main (top) table with either station or parameter data, depending on m_type.

Class: DataEditPage
Function: FillDataTable() : void
Description: Populate the data (bottom) table with project data from the selected station in the main table.

Class: DataEditPage
Function: EndDataEdit() : void
Description: Sorts the project data if the time of a data item has been changed.

Class: DataEditPage
Member: m_mainTable : TableListCtrl
Description: The main (top) table widget.

Class: DataEditPage
Member: m_dataTable : TableListCtrl
Description: The data (bottom) table widget.

Class: DataEditPage
Member: m_allObjectList : CStringList
Description: The list of all objects being considered, either stations or parameters.

Class: DataEditPage
Member: m_pFieldRec : GDPFieldRec*
Description: The data edits will be done to the m_pFieldRec object directly.

Class: DataEditPage
Member: m_isModified : BOOL
Description: If the makes any changes, then set this to TRUE. This will cause UpdateAll to be sent to all other pages.
5. Source File: **DeleteItemsDlg.h**

**Class:** DeleteItemsDlg  
**Function:** DeleteItemsDlg(CWnd* pParent = NULL) : Constructor  
**Description:** Constructor for the delete items chooser. Use SetData to build a list of items that the user will choose from. The indices of the choices will be stored in reverse order to facilitate deletion in m_itemArray.

**Class:** DeleteItemsDlg  
**Function:** SetData(CArray<CString, CString>& &items) : void  
**Description:** The user will choose from members of the 'items' parameter.  
**Input Args:** items: Initialize the choice list with items from 'items'.

**Class:** DeleteItemsDlg  
**Function:** SetData(CArray<int, int>& &items) : void  
**Description:** The user will choose from members of the 'items' parameter.  
**Input Args:** items: Initialize the choice list with items from 'items'.

**Class:** DeleteItemsDlg  
**Member:** m_itemArray : CArray<CString, CString>  
**Description:** Contains the user selection in reverse order to make deletion from lists easier.

6. Source File: **ExportOptionsDlg.h**

**Class:** ExportOptionsDlg  
**Function:** ExportOptionsDlg(CWnd* pParent = NULL) : Constructor  
**Description:** Constructor for the report export option dialog. The choice is stored in m_option.

7. Source File: **gdpdata.h**

**Class:** GDPFieldRec  
**Function:** GDPFieldRec() : Constructor  
**Description:** A class that manages data for a single station and parameter. There can be any amount of data points for the given combination.

**Class:** GDPFieldRec  
**Function:** GetDateSpan(int &startDay, int &startMonth, int &startYear, int &endDay, int &endMonth, int &endYear) : void  
**Description:** If this has a data point that falls before the startDay, startMonth, and startYear, then update these parameters with the earliest date. The same is done for the end year. If the dates are uninitialized, then set startYear to -1.

**Input Args:**  
- startDay: The current minimum project data day component. If this has a data point with an earlier date, then this member will be set to that day.  
- startMonth: The current minimum project data month component. If this has a data point with an earlier date, then this member will be set to that month.  
- startYear: The current minimum project data year component. If this has a data point with an earlier date, then this member will be set to that year. NOTE: set this argument to -1 if the dates are uninitialized.  
- endDay: The current maximum project data day component. If this has a data point
with a later date, then this member will be set to that day.

Input Args:  
- **endMonth**: The current maximum project data month component. If this has a data point with a later date, then this member will be set to that month.
- **endYear**: The current maximum project data year component. If this has a data point with a later date, then this member will be set to that year.

Class: **GDPFieldRec**

Function: **AddData(IDSTime time, double val, BOOL mask=FALSE)** : void

Description: Add a new data point to this record. The data will be inserted so that each data point is sorted by time.

Input Args:  
- **time**: The time of the new data point.
- **val**: The value of the new data point.
- **mask**: If TRUE, then this data point should not be included in graphs and aggregations.

Class: **GDPFieldRec**

Function: **AddData(DeleteData(int index))** : void

Description: Remove the data point indexed by 'index'.

Input Args:  
- **index**: The index into m_valueArray and m_readingTimeArray that should be removed.

Class: **GDPFieldRec**

Function: **ReplaceDataUnsorted(int index, IDSTime time, double val, BOOL mask=FALSE)** : BOOL

Description: Update the given data point with the new values. Do not sort (caller must eventually call Sort or else the data will be in an inconsistent state).

Input Args:  
- **index**: The index into m_valueArray and m_readingTimeArray that should be updated.
- **time**: The time of the new data point.
- **val**: The value of the new data point.
- **mask**: If TRUE, then this data point should not be included in graphs and aggregations.

Returns: TRUE if successful.

Class: **GDPFieldRec**

Function: **Sort(BOOL force=FALSE)** : void

Description: Sort the data points by time. Ignored if m_isUnsorted is FALE unless force is set to TRUE.

Input Args:  
- **force**: If TRUE then ignore m_isUnsorted member and sort the data.

Class: **GDPFieldRec**

Function: **GenerateDataFile()** : void

Description: Output a spreadsheet named '<m_stationName>.txt' and output a line for every data point.

Class: **GDPFieldRec**

Function: **operator<<(ofstream &ostr, GDPFieldRec &fieldRec)** : ofstream&

Description: Dump this to the output stream.

Input Args:  
- **ostr**: The output stream to write to.
- **fieldRec**: The object to save.

Returns: ostr
1.0 Routine List by Source File

Class: GDPFieldRec
Function: operator>>(ifstream &istr, GDPFieldRec &fieldRec) : ifstream&
Description: Read this from the input stream.
Input Args: istr: The input stream to read from.
Input Args: fieldRec: The object to load the new state to.
Returns: istr

Class: GDPFieldRec
Member: m_valueArray : CArray<double, double>
Description: The list of values of all the data points. One element for each time. Corresponds to m_readingTimeArray.

Class: GDPFieldRec
Member: m_readingTimeArray : CArray<IDSTime, IDSTime&>
Description: The list of times of all the data points. One element for each value. Corresponds to m_valueArray. This allows data points to be not included in data availability graphs and report generation.

Class: GDPFieldRec
Member: m_maskArray : CArray<BOOL, BOOL>
Description: The list of mask settings of all the data points. One element for each value. Corresponds to m_valueArray. This allows data points to be not included in data availability graphs and report generation.

Class: GDPFieldRec
Member: m_stationName : CString
Description: The name of the station.

Class: GDPFieldRec
Member: m_stationLoc : CString
Description: The location of the station. Currently unused.

Class: GDPFieldRec
Member: m_parameter : CString
Description: The name of the parameter of this data.

Class: GDPFieldRec
Member: m_units : CString
Description: The units of the parameter.

Class: GDPFieldRec
Member: m_isUnsorted : BOOL
Description: Set this to TRUE if the data array is unsorted. This will usually only be set in ReplaceDataUnsorted.

Class: GDPFieldRec
Member: m_offset : double
Description: Auxiliary members to allow for scaling (ie, if well measurements need to be offset by the height of the well, etc.).
Class: GDPFieldRec
Member: m_mult : double
Description: Auxiliary members to allow for scaling (ie, if well measurements need to be offset by the height of the well, etc.).

Class: GDPFieldFile
Function: GDPFieldFile() : Constructor
Description: A class that manages a group of fieldRecs, usually from the same file.

Function: ReadFile(CString filename) : BOOL
Description: Tests the file suffix and calls either ReadStoret or ReadWatstore.
Input Args: filename: Name of the file to read from.
Returns: TRUE if file successfully read.

Function: ReadStoret(CString filename) : BOOL
Description: Parse the file assuming STORET database format.
Input Args: filename: Name of the file to read from.
Returns: TRUE if file successfully read.

Function: ReadWatstore(CString filename) : BOOL
Description: Parse the file assuming WATSTORE database format.
Input Args: filename: Name of the file to read from.
Returns: TRUE if file successfully read.

Function: ReadSpreadsheetFile(CString filename, int format, char separator, CString paramName, CString unitsName) : BOOL
Description: Read data using spreadsheet format. If format == 0, then call ReadSpreadsheetStreamSingle, otherwise call ReadSpreadsheetStreamMulti.
Input Args: filename: Name of the file to read from.
Input Args: format: if 0, then assume file uses single station and parameter format.
Input Args: separator: Token separator, usually either ',' or '.'.
Input Args: paramName: The name of the parameter to use when saving data.
Input Args: unitsName: The name of the units to use when saving data.
Returns: TRUE if file successfully read.

Function: ReadSpreadsheetClipboard(int format, char separator, CString paramName, CString unitsName) : BOOL
Description: Read data using spreadsheet format. Identical to ReadSpreadsheetFile except that data is read from the clipboard.
Input Args: format: if 0, then assume file uses single station and parameter format.
Input Args: separator: Token separator, usually either ',' or '.'.
Input Args: paramName: The name of the parameter to use when saving data.
Input Args: unitsName: The name of the units to use when saving data.
1.0 Routine List by Source File

Returns: TRUE if file successfully read.

Class: GDPFieldFile
Function: ReadSpreadsheetStreamSingle(istream istr, char separator, CString paramName, CString unitsName) : BOOL
Description: Read data using spreadsheet format. Assumes that the file contains data for a single station and parameter.
Input Args: istr: the stream to read from.
Input Args: separator: Token separator, usually either ‘ ’ or ','. 
Input Args: paramName: The name of the parameter to use when saving data.
Input Args: unitsName: The name of the units to use when saving data.
Returns: TRUE if file successfully read.

Class: GDPFieldFile
Function: ReadSpreadsheetStreamMulti(istream istr, char separator, CString paramName, CString unitsName) : BOOL
Description: Read data using spreadsheet format. Assumes that the file contains data for several stations and one parameter.
Input Args: istr: the stream to read from.
Input Args: separator: Token separator, usually either ‘ ’ or ','. 
Input Args: paramName: The name of the parameter to use when saving data.
Input Args: unitsName: The name of the units to use when saving data.
Returns: TRUE if file successfully read.

Class: GDPFieldFile
Function: GetFieldRecFromStationAndParam(CString stationName, CString paramName, int &icount) : GDPFieldRec*
Description: Return a pointer to the field record that corresponds to the given station and parameter. If icount is given, then start the search at record ‘icount’. The index of the returned record could be assigned to icount, but currently is set to 0.
Input Args: stationName: The name of the station's data.
Input Args: paramName: The name of the parameter of interest.
Input Args: icount: The index to begin the search. The index of the returned record could be assigned to icount, but currently is set to 0.
Returns: Pointer to the field record that corresponds to the given station and parameter.

Class: GDPFieldFile
Function: operator<<(ofstream &ostr, GDPFieldFile &fieldRec) : ofstream&
Description: Dump this to the output stream.
Input Args: ostr: The output stream to write to.
Input Args: fieldRec: The object to save.
Returns: ostr
Class: **GDPFIELDFILE**
Function: `operator>>(istream &istr, GDPFIELDFILE &fieldRec) : ifstream &` 
Description: Read this from the input stream. 
Input Args: `istr`: The input stream to read from. 
Input Args: `fieldRec`: The object to load the new state to. 
Returns: `istr`

Class: **GDPFIELDFILE**
Function: `MonthStringToInt(CString month) : int` 
Description: Converts the three-letter capitalized month string to an integer (1-12). 
Input Args: `month`: The three-letter capitalized month string to convert. 
Returns: The corresponding number of the month, from 1-12.

Class: **GDPFIELDFILE**
Function: `GetDateSpan(int &startDay, int &startMonth, int &startYear, int &endDay, int &endMonth, int &endYear) : void` 
Description: If this has any data points that falls before the startDay, startMonth, and startYear, then update these parameters with the earliest date. The same is done for the end year. If the dates are uninitialized, then set startYear to -1. 
Input Args: `startDay`: The current minimum project data day component. If this has a data point with an earlier date, then this member will be set to that day. 
Input Args: `startMonth`: The current minimum project data month component. If this has a data point with an earlier date, then this member will be set to that month. 
Input Args: `startYear`: The current minimum project data year component. If this has a data point with an earlier date, then this member will be set to that year. NOTE: set this argument to -1 if the dates are uninitialized. 
Input Args: `endDay`: The current maximum project data day component. If this has a data point with a later date, then this member will be set to that day. 
Input Args: `endMonth`: The current maximum project data month component. If this has a data point with a later date, then this member will be set to that month. 
Input Args: `endYear`: The current maximum project data year component. If this has a data point with a later date, then this member will be set to that year.

Class: **GDPFIELDFILE**
Function: `GenerateDataFiles() : void` 
Description: Store the station data of all the stations in this group to a files using the name of the station.

Class: **GDPFIELDFILE**
Member: `m_fileName : CString` 
Description: Name of the file used to build this record's data.

Class: **GDPFIELDFILE**
Member: `m_stationArray : CArray<GDPFIELDREC, GDPFIELDREC&>` 
Description: List of station groups.
1.0 Routine List by Source File

**Class: GDPStationRec**
Function: GDPStationRec() : Constructor
Description: This class organizes records by station. Every station will have a list of valid parameters.

**Class: GDPStationRec**
Member: m_stationName : CString
Description: Name of the station that this record organizes.

**Class: GDPStationRec**
Member: m.alias : CString
Description: A possible alias for this station. Most likely will be used when this station leads an aggregate of other stations.

**Class: GDPStationRec**
Member: m_isHeadOfAggregate : BOOL
Description: If TRUE, then this station contains info for an aggregate (the m_alias member will be valid as will the m_recordList member).

**Class: GDPStationRec**
Member: m_isPartOfAggregate : BOOL
Description: If TRUE, then this station is in another station's aggregation list.

**Class: GDPStationRec**
Member: m_aggregateStationList : CStringList
Description: A list of aggregate stations of this station.

**Class: GDPStationRec**
Member: m_recordList : CList<int, int>
Description: Helper list to facilitate lookup of field records based on station. This list will index into the field manager's list.

**Class: GDPParamRec**
Function: GDPParamRec() : Constructor
Description: This class organizes records by parameter. Every parameter will have a list of associated stations.

**Class: GDPParamRec**
Member: m_paramName : CString
Description: The name of the parameter that this object organizes.

**Class: GDPParamRec**
Member: m_alias : CString
Description: A possible alias for this parameter. Most likely will be used when this parameter leads an aggregate of other parameters.

**Class: GDPParamRec**
Member: m_units : CString
Description: The name of the associated parameter's units.
7. Source File: gdpdata.h

Class: GDPParamRec
Member: m_isHeadOfAggregate : BOOL
Description: If TRUE, then this parameter contains info for an aggregate (the m_alias member will be valid as will the m_recordList member).

Class: GDPParamRec
Member: m_isPartOfAggregate : BOOL
Description: If TRUE, then this parameter is in another parameter's aggregation list.

Class: GDPParamRec
Member: m_aggregateParamList : CList
Description: The list of aggregate parameters of this parameter.

Class: GDPParamRec
Member: m_recordList : CList<int, int>
Description: Helper member to facilitate lookup of field records based on parameter. This list will index into the field manager's list.

Class: GDPUunitsRec
Function: GDPUunitsRec() : Constructor
Description: Class for handling unit conversions and name changes.

Class: GDPUunitsRec
Function: operator==(const GDPUunitsRec&) const : BOOL
Description: Compares the m_name members and returns TRUE if they are the same.

Class: GDPUunitsRec
Member: m_name : CStr
Description: The name by which the units are referred to in the raw data. This will be the key used to lookup the units conversion when processing GDPFieldRecs.

Class: GDPUunitsRec
Member: m_alias : CStr
Description: An optional alias, usually will be the name of the units that this conversion represents (ie, if m_unitsName is 'celcius', then m_unitsAlias might be 'fahrenheit').

Class: GDPUunitsRec
Member: m_comment : CStr
Description: An optional comment, like "F -> C"

Class: GDPUunitsRec
Member: m_mult : double
Description: Conversion multiplier.

Class: GDPUunitsRec
Member: m_offset : double
Description: Conversion additive offset.

Class: GDPFieldManager
Function: GDPFieldManager() : Constructor
Description: Class for managing a collection of GDPFieldFile.
1.0 Routine List by Source File

Class: GDPFieldManager
Function: operator[](int index) : GDPFieldRec
Description: Shortcut to GetFieldRec. Returns the index'th field record. The count starts at the first field group and continues to the end.
Input Args: index: The index of the field record to return.
Returns: Reference to the field record.

Class: GDPFieldManager
Function: operator[](int index) : GDPFieldRec
Description: Returns the index'th field record. The count starts at the first field group and continues to the end.
Input Args: index: The index of the field record to return.
Returns: Reference to the field record.

Class: GDPFieldManager
Function: GetLastFieldRecordIndex() : int
Description: Calculates the total number of field records in the project minus one.
Returns: The total number of field records in the project minus one.

Class: GDPFieldManager
Function: ReadProject(CString filename) : BOOL
Description: Load the project data from file. Initialize supporting members.
Input Args: filename: Name of the file to read from.
Returns: TRUE if the file was parsed correctly.

Class: GDPFieldManager
Function: WriteProject(CString filename) : BOOL
Description: Save the project data to file.
Input Args: filename: Name of the file to write to.
Returns: TRUE if the file was able to be written to.

Class: GDPFieldManager
Function: ReadFile(CString filename) : BOOL
Description: Add new data in 'filename' to the project. Update helper members.
Input Args: filename: Name of the file to read from.
Returns: TRUE if the file was able to be read.

Class: GDPFieldManager
Function: ReadFiles(CStringList &rawFilesList) : BOOL
Description: Add new data from the list of file names to the project. Update helper members.
Input Args: rawFilesList: List of file names to add new data from.
Returns: TRUE if the files were able to be read.

Class: GDPFieldManager
Function: ReadSpreadsheetFile(CString filename, int format, char separator, CString
Class: GDPFieldManager
Function: ReadSpreadsheetClipboard(int format, char separator, CString paramName, CString unitsName) : BOOL
Description: Add project data using spreadsheet format from the clipboard.
Input Args: format: If 0, then use single station format; otherwise use multiple station format.
Input Args: separator: The token separator, usually ‘ ’ or ‘,’.
Input Args: paramName: The name of the parameter to save the data to.
Input Args: unitsName: The name of the parameter’s units.
Returns: TRUE if the files were able to be read.

Class: GDPFieldManager
Function: InitializeStationList() : void
Description: Builds m_stationMap, the list of all the stations and the indices of all the records that have data for each station.

Class: GDPFieldManager
Function: BuildStationReferences() : void
Description: Populate the helper list in each GDPStationRec in m_stationMap with indices to the data records.

Class: GDPFieldManager
Function: InitializeParamList() : void
Description: Initialize m_paramMap, the list of all the parameters and the indices of all the stations that have data for each parameter.

Class: GDPFieldManager
Function: BuildParamReferences() : void
Description: Populate the helper list in GDPParamRec in m_paramMap with indices to the data records.

Class: GDPFieldManager
Function: InitializeUnitsList() : void
Description: Create the list of all the units, m_unitsMap.

Class: GDPFieldManager
Function: WeedUnitsList() : void
Description: Throw away stale units references in m_unitsMap, usually called after deleting parameters.
1.0 Routine List by Source File

Class: GDPFieldManager
Function: GetFileNameArray(CArray<CString, LPCSTR> &fileNameArray) : void
Description: Return the list of filenames used to get the raw data.

Class: GDPFieldManager
Function: GetDateSpan(int &startDay, int &startMonth, int &startYear, int &endDay, int &endMonth, int &endYear) : void
Description: If this has any data points that falls before the startDay, startMonth, and startYear, then update these parameters with the earliest date. The same is done for the end year. If the dates are uninitialized, then set startYear to -1.
Input Args: startDay: The current minimum project data day component. If this has a data point with an earlier date, then this member will be set to that day.
Input Args: startMonth: The current minimum project data month component. If this has a data point with an earlier date, then this member will be set to that month.
Input Args: startYear: The current minimum project data year component. If this has a data point with an earlier date, then this member will be set to that year. NOTE: set this argument to -1 if the dates are uninitialized.
Input Args: endDay: The current maximum project data day component. If this has a data point with a later date, then this member will be set to that day.
Input Args: endMonth: The current maximum project data month component. If this has a data point with a later date, then this member will be set to that month.
Input Args: endYear: The current maximum project data year component. If this has a data point with a later date, then this member will be set to that year.

Class: GDPFieldManager
Function: GetUnits(CString paramName) : CString
Description: Return the units of the given parameter.
Input Args: The name of the parameter to query.
Returns: The units of the given parameter.

Class: GDPFieldManager
Function: GetDataByStation(CString station, CString param, IDSTime startDate, IDSTime endDate, IDSTime::IDSTimeIntervalMode, int intervalCount, CArray<double, double> dataArray[4], int doFillGaps=0) : BOOL
Description: Load the dataArray with the data values for the station and parameter in the span of time from startDate to endDate. The dataArray is a reference to a 4D array with min, max, mean, hasData.
Input Args: station: The name of the station to query. Results will include any aggregated stations.
Input Args: param: The name of the parameter to query. Results will include any aggregated parameters.
Input Args: startDate: Lower bound of the date to search in.
Input Args: endDate: Upper bound of the date to search in.
Input Args: intervalMode: Granularity of the search; a unit of time.
Input Args: intervalCount: Number of time units to skip; i.e. for every week, use 1 and set intervalMode to IDSTIME_WEEK.
Input Args: dataArray: Results go here. First index is data min; second is max; third is data mean;
fourth is availability (if non-zero, then the other three arrays are valid).

**Input Args:**  
- `doFillGaps`: If TRUE, then interpolate any missing data.

**Returns:**  
- TRUE if successful.

**Class:** GDPFieldManager  
**Function:** GetDataByParameter(CString param, CString station, IDSTime startDate, IDSTime endDate, IDSTime::IDSInterval intervalMode, int intervalCount, CArray<double, double> &dataArray, int doFillGaps=0) : BOOL

**Description:** Load the dataArray with the availability results.

**Input Args:**  
- `param`: The name of the parameter to query. Results will include any aggregated parameters.
- `station`: The name of the station to query. Results will include any aggregated stations.
- `startDate`: Lower bound of the date to search in.
- `endDate`: Upper bound of the date to search in.
- `intervalMode`: Granularity of the search; a unit of time.
- `intervalCount`: Number of time units to skip; i.e. for every week, use 1 and set intervalMode to IDSTIME_WEEK.
- `dataArray`: Availability results go here. If non-zero, then there is data in this interval.

**Returns:**  
- TRUE if successful.

**Class:** GDPFieldManager  
**Function:** SearchRecords(int y, CList<int, int> &fieldRecIndicesList, IDSTime lowerBound, IDSTime upperBound, CArray<double, double> &dataArray[4]) : void

**Description:** Load the dataArray with the data values for the station and parameter in the span of time from startDate to endDate. The dataArray is a reference to a 4D array with min, max, mean, hasData.

**Input Args:**  
- `y`: The index into dataArray to store results.
- `fieldRecIndicesList`: The data records to query.
- `lowerBound`: The lower bound of the search date.
- `upperBound`: The upper bound of the search date.
- `dataArray`: Results go here. First index is data min; second is max; third is data mean; fourth is availability (if non-zero, then the other three arrays are valid).

**Class:** GDPFieldManager  
**Function:** SearchRecords(int y, CList<int, int> &fieldRecIndicesList, IDSTime lowerBound, IDSTime upperBound, CArray<double, double> &dataArray) : void

**Description:** Load the dataArray with the availability results.

**Input Args:**  
- `y`: The index into dataArray to store results.
- `fieldRecIndicesList`: The data records to query.
- `lowerBound`: The lower bound of the search date.
- `upperBound`: The upper bound of the search date.
- `dataArray`: Availability results go here. If non-zero, then there is data in this interval.

**Class:** GDPFieldManager  
**Member:** m_stationMap : CMap<CString, LPCSTR, GDPStationRec, GDPStationRec>&

**Description:** Given the name of a station (key), retrieve the aggregation info.
1.0 Routine List by Source File

Class: GDPFieldManager
Member: m_paramMap : CMap&lt;CString, LPCSTR, GDPStationRec, GDPStationRec&
Description: Given the name of a parameter (key), retrieve the aggregation info.

Class: GDPFieldManager
Member: m_unitsMap : CMap&lt;CString, LPCSTR, GDPStationRec, GDPStationRec&
Description: Given the name of some units (key), retrieve the aggregation info.

Class: GDPFieldManager
Function: RebuildAliasMap(AggregationType) : void
Description: Build the m_stationAliasMap/m_paramAliasMap, based on the input argument.
Input Args: AggregationType: either STATION or PARAMETER.

Class: GDPFieldManager
Member: m_stationAliasMap : CMap&lt;CString, LPCSTR, CString, LPCSTR>
Description: Because the m_stationMap keys on alias, there needs to be a way to do lookups based on object name. These will convert from object name to object alias. Note that only aggregated objects will appear in these maps.

Class: GDPFieldManager
Member: m_paramAliasMap : CMap&lt;CString, LPCSTR, CString, LPCSTR>
Description: Because the m_paramMap keys on alias, there needs to be a way to do lookups based on object name. These will convert from object name to object alias. Note that only aggregated objects will appear in these maps.

Class: GDPFieldManager
Member: m_units AliasMap : CMap&lt;CString, LPCSTR, CString, LPCSTR>
Description: Because the m_unitsMap keys on alias, there needs to be a way to do lookups based on object name. These will convert from object name to object alias. Note that only aggregated objects will appear in these maps.

Class: GDPFieldManager
Member: m_fieldFileArray : CArray&lt;GDFFieldFile, GDFFieldFile&
Description: The list of GDFFieldFiles that this class manages.

Class: GDPFieldManager
Function: GetUnitsList(CStringList &unitsList) : void
Description: Return a list of the names of all units. The name string can be used to lookup unit data.
Input Args: unitsList: Store results here.

Class: GDPFieldManager
Function: GetUnitsRec(CString unitName, GDPUnitsRec &unitsRec) : BOOL
Description: Get the GDPUnitsRec for the given units.
Input Args: unitName: Name of the units to retrieve information for.
Input Args: unitsRec: Results go here. WARNING: any changes to this object must be saved using SetUnitsRec.
Returns: TRUE if the units record was found.
Class: GDPFieldManager
Function: SetUnitsRec(CString oldUnits, GDPUnitsRec &newUnitsRec) : void
Description: Update value in m_unitsMap for these units.
Input Args: oldUnits: Name of the units to update.
Input Args: newUnitsRec: Replace the old value with this.

Class: GDPFieldManager
Function: GetAllObjectList(CStringList &objectList, AggregationType whichType) : void
Description: Return the list of all the stations in the project if whichType == 0, otherwise all the parameters.
Input Args: objectList: Output parameter.
Input Args: whichType: either STATION or PARAMETER.

Class: GDPFieldManager
Function: GetCombinedObjectList(CStringList &allList, AggregationType whichType) : void
Description: Return the list of all stations or parameters that aggregate other objects or are not part of another aggregate.
Input Args: allList: Output parameter.
Input Args: whichType: either STATION or PARAMETER.

Class: GDPFieldManager
Function: GetStationRec(CString stationName, GDPSationRec &stationRec) : BOOL
Description: Return the GDPSationRec in m_stationMap for this station.
Input Args: stationName: Name of the station to retrieve information for. If this is an alias, then this is used to get the real station name.
Input Args: stationRec: Output parameter. WARNING: any changes to this object must be saved using SetStationRec.
Returns: TRUE if the record was found.

Class: GDPFieldManager
Function: SetStationRec(CString stationName, GDPSationRec &stationRec) : void
Description: Update the GDPSationRec data for this station in m_stationMap.
Input Args: stationName: Name of the station to update information for. If this is an alias, then this is used to get the real station name.
Input Args: stationRec: New value to assign to m_stationAliasMap.

Class: GDPFieldManager
Function: DelStationRec(CString stationName) : void
Description: Remove the GDPSationRec data for this station in m_stationMap.
Input Args: stationName: Name of the station to delete. If this is an alias, then this is used to get the real station name.

Class: GDPFieldManager
Function: GetParamRec(CString paramName, GDPParamRec &paramRec) : BOOL
Description: Return the Return the GDPParamRec in m_paramMap for this parameter.
Input Args: paramName: Name of the parameter to retrieve information for. If this is an alias, then this is used to get the real parameter name.
Input Args: paramRec: Output parameter. WARNING: any changes to this object must be saved
using SetParamRec.

Returns: TRUE if the record was found.

Class: GDPFieldManager
Function: GetParamRec(CString paramName, GDPParamRec &paramRec) : void
Description: Update the GDPParamRec data for this station in m_paramMap.
Input Args: paramName: Name of the parameter to retrieve information for. If this is an alias, then this is used to get the real parameter name.
Input Args: paramRec: Output parameter. WARNING: any changes to this object must be saved using SetParamRec.
Returns: TRUE if the record was found.

Class: GDPFieldManager
Function: DelParamRec(CString paramName) : void
Description: Remove the GDPParamRec data from m_paramMap for this station.
Input Args: paramName: Name of the parameter to delete. If this is an alias, then this is used to get the real parameter name.

Class: GDPFieldManager
Function: AddNewStation(int count) : BOOL
Description: Add 'count' stations to the project. A ListDialog will be displayed for the user to select an associated parameter from.
Input Args: count: Number of stations to add to the project.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: AddNewParameter(int count, CString paramName="parameter", CString unitsName="units") : BOOL
Description: Add 'count' parameters to the project with units 'unitsName'. Each parameter will be named 'parameter1', 'parameter2', etc.
Input Args: count: Number of parameters to add to the project.
Input Args: paramName: Base name of the new parameter.
Input Args: unitsName: Name of the parameter's units.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: AddStationData(CString stationName, CString paramName, int count) : BOOL
Description: Add 'count' data points to the record with station 'stationName' and parameter 'paramName'. The new data points will be added to the end of any existing data; otherwise the current time is used. The new data is set to zero.
Input Args: stationName: Name of the station to add data to.
Input Args: paramName: Name of the parameter to add data to.
Input Args: count: Number of data points to add to the project.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: DeleteStationData(CString stationName, CString paramName, CArray<int, int>
Class: GDPFieldManager
Function: RemoveDataPoints(BOOLEAN &indices) : BOOL
Description: Remove data points indexed by 'indices'. Assumes that the indices array is in reverse order (highest index first).
Input Args: stationName, paramName, indices
Returns: TRUE if successful.

Class: GDPFieldManager
Function: DeleteStations(CArray&lt;CString, CString&> &stationArray) : BOOL
Description: Delete all stations in 'stationArray' from the project.
Input Args: stationArray
Returns: TRUE if successful.

Class: GDPFieldManager
Function: DeleteParameters(CArray&lt;CString, CString&> &paramArray) : BOOL
Description: Delete all records with parameters in 'paramArray' from the project.
Input Args: paramArray
Returns: TRUE if successful.

Class: GDPFieldManager
Function: HasStationName(CString stationName) : BOOL
Description: Checks if there is a station called 'stationName' in the project.
Input Args: stationName
Returns: TRUE if successful.

Class: GDPFieldManager
Function: ChangeStationName(CString oldName, CString newName) : BOOL
Description: Rename a station 'oldName' to 'newName'.
Input Args: oldName, newName
Returns: TRUE if successful.

Class: GDPFieldManager
Function: ChangeParameterForStation(CString station, CString newParam) : BOOL
Description: Change all parameters of station 'station' to 'newParam'.
Input Args: station, newParam
Returns: TRUE if successful.

Class: GDPFieldManager
Function: HasParamName(CString paramName) : BOOL
Description: Checks if there is a parameter called 'paramName' in the project.
Input Args: paramName
Returns: TRUE if successful.
1.0 Routine List by Source File

Class: GDPFieldManager
Function: ChangeParamName(CString oldName, CString newName) : BOOL
Description: Change all references to parameter 'oldName' to 'newName'.
Input Args: oldName: Name of the parameter to change.
Input Args: newName: New name of the parameter.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: ChangeUnitsForParam(CString paramName, CString oldUnits, CString newUnits) : BOOL
Description: Change all references to units 'oldUnits' to 'newUnits' in parameter 'paramName'.
Input Args: paramName: Name of the parameter to alter units for.
Input Args: oldUnits: Name of the units to change.
Input Args: newUnits: Name of the new units.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: GetFieldRecFromStationAndParam(CString stationName, CString paramName) : GDPFieldRec*
Description: Return the field record for the given station and parameter.
Input Args: stationName: The station name to query.
Input Args: paramName: The parameter name to query.
Returns: A pointer to the field record.

Class: GDPFieldManager
Function: AddAggregationItem(CStringList &pickList, CString alias, AggregationType whichType = NONE) : BOOL
Description: Create a new aggregation item. The first item in the list is the aggregation controller; use this station/parameter to look up agg information.
Input Args: pickList: The list of stations or parameters to combine.
Input Args: alias: The name of the aggregate.
Input Args: whichType: Select STATION or PARAMETER.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: RemoveAggregationItem(CString alias, CStringList &newList, AggregationType whichType = NONE) : BOOL
Description: Remove the aggregation item with name 'alias'. Returns the unaggregated item names in 'newList'.
Input Args: alias: The name of the aggregation item to undo.
Input Args: newList: The names of the unaggregated stations or parameters.
Input Args: whichType: Select STATION or PARAMETER. If NONE, then the type is determined by m_currentAggEditType.
Returns: TRUE if successful.

Class: GDPFieldManager
Function: GetAggObjectList(CStringList &aggList, CStringList &aggAliasList,
AggregationType whichType = NONE) : void  
Description: Return lists of aggregate heads and aliases of all the stations in the project if whichType == 0; otherwise all the parameters.
Input Args:  
- `aggList`: Output parameter with the names of all the stations that are aggregate controllers.
- `aggAliasList`: Output parameter with the aliases of the aggregate controllers.
- `whichType`: Select STATION or PARAMETER. If NONE, then the type is determined by m_currentAggEditType.

Class: GDPFieldManager  
Function: GetAggItemObjectList(CString index, CStringList &aggList, AggregationType whichType) : BOOL  
Description: Return the indices of all the aggregated objects from the object referenced from 'index'.
Input Args:  
- `index`: The aggregation alias to query.
- `aggAliasList`: Output parameter with the names of all the aggregated items.
- `whichType`: Select STATION or PARAMETER. If NONE, then the type is determined by m_currentAggEditType.
Returns: TRUE if successful.

Class: GDPFieldManager  
Function: GetAssociatedObjects(CString index, CStringList &assocList, AggregationType whichType) : BOOL  
Description: Given a station name or aggregation alias, return all the parameters associated with the station(s), and visa-versa. **Note that for the station data, there are two additional elements, the multiplier and the offset. Make sure the assocList is read <param1> <mult1> <offset1> <param2> <mult2> <offset2> etc.
Input Args:  
- `index`: The aggregation alias to query.
- `assocList`: Output parameter with the parameters associated with the station(s) or parameter(s).
- `whichType`: Select STATION or PARAMETER. If NONE, then the type is determined by m_currentAggEditType.
Returns: TRUE if successful.

Class: GDPFieldManager  
Function: StartAggEdits(AggregationType whichType) : void  
Description: Assigns to m_currentAggEditType.
Input Args:  
- `whichType`: Select STATION or PARAMETER.

Class: GDPFieldManager  
Member: m_currentAggEditType : AggregationType  
Description: Shortcut to identify whether stations or parameters are currently being aggregated.
Input Args:  
- `whichType`: Select STATION or PARAMETER.

Class: GDPFieldManager  
Function: GenerateDataFiles() : void  
Description: Store the station data of all the stations in this group to a files using the name of the station.
1.0 Routine List by Source File

8. Source File: GDPDoc.h

Class: CGDPDoc
Function: AddInputFiles(): void
Description: Add Storet or Watstore input files to project.

Class: CGDPDoc
Function: ReadSpreadsheetFile(CString filename, int format, char separator, CString paramName, CString unitsName): void
Description: Add stations from spreadsheet files to project.
Input Args: filename: Name of the spreadsheet file to add.
Input Args: format: If 0, then use single station format; otherwise use multiple station format.
Input Args: separator: The token separator, usually ',' or '.'.
Input Args: paramName: The name of the parameter to save the data to.
Input Args: unitsName: The name of the parameter's units.

Class: CGDPDoc
Function: ReadSpreadsheetClipboard(int format, char separator, CString paramName, CString unitsName): void
Description: Add stations from spreadsheet data in the clipboard to the project.
Input Args: format: If 0, then use single station format; otherwise use multiple station format.
Input Args: separator: The token separator, usually ',' or '.'.
Input Args: paramName: The name of the parameter to save the data to.
Input Args: unitsName: The name of the parameter's units.

Class: CGDPDoc
Function: UpdateAll(): void
Description: Update the view to display new data.

Class: CGDPDoc
Member: m_gdpFieldManager: GDPFieldManager
Description: Manages the input data.

Class: CGDPDoc
Member: m_unitsData: UnitsData
Description: Manages standard unit conversions.

Class: CGDPDoc
Member: m_unitsFilename: CString
Description: The default units filename.

9. Source File: GDPViewSheet.h

Class: GDPViewSheet
Function: GDPViewSheet(GDPPDoc*pDoc, CWnd*pParentWnd = NULL): Constructor
Description: Input view constructor.
Class: GDPViewSheet
Member:  m_projectInfoPage : PageInfo
Description:  Tab with the input files that compose the project data. Only display Storet and Watstore inputs.

Class: GDPViewSheet
Member:  m_aggregationPage : AggregationPage
Description:  The aggregation display and creation tab.

Class: GDPViewSheet
Member:  m_dataAvailPage : DataAvailPage
Description:  The data availability tab.

Class: GDPViewSheet
Member:  m_dataEditPage : DataEditPage
Description:  The data edit tab.

Class: GDPViewSheet
Member:  m_unitsPage : UnitsPage
Description:  The units display and editor tab.

Class: GDPViewSheet
Function:  Resize() : void
Description:  Causes all the tabs to resize themselves to fit the view.

Class: GDPViewSheet
Function:  UpdateAll() : void
Description:  Redisplay their data to reflect any changes.

10. Source File:  GraphDlg.h

Class: GraphDlg
Function:  GraphDlg(CWnd* pParent = NULL) : Constructor
Description:  Front-end tp CGraph class.

Class: GraphDlg
Function:  ~GraphDlg() : Destructor
Description:  Frees dynamically allocated memory.

Class: GraphDlg
Function:  SetDimensions(int nframes, int nsets, int ncols, int nlabels, int missingVal=0, BOOL useLabelSkip=TRUE, BOOL useDistData=FALSE) : void
Description:  Assigns the m_nSets, m_nCols, and m_nLabels variables and allocates memory.
Input Args:  nframes: The number of frames that will be allocated. Each frame is a slide that is displayed after a click event. Also can be induced from GraphControlDlg.
Input Args:  nsets: The number of sets that will be graphed. Each set represents a group of data and
1.0 Routine List by Source File

can be assigned a legend title, color, etc, using other routines.

Input Args:  
- **ncols**: The number of X data points per set.
- **nlables**: The number of X data labels that should be allocated.
- **missingVal**: The initial value to be assigned to m_setMissing (0 means point should be displayed, 128 means point is missing).
- **useLabelSkip**: if TRUE, then try to come up with a reasonable LabelEvery value so that column labels won't be too squished together.
- **useDistData**: Allocate additional data structures (m_setDist) to allow for distance data to be inputted (allows for arbitrary X points).

**Class**: **GraphDlg**  
**Function**: SetTitles() : void  
**Description**: Assigns the graph's title, x and y titles, and a title for the dialog window.  
**Input Args**:  
- **title**: the graph title (m_graphTitle)
- **xtitle**: the X axis title (m_XTitle)
- **ytitle**: the Y axis title (m_YTitle)
- **windowTitle**: the window dialog title (m_windowTitle)

**Class**: **GraphDlg**  
**Function**: DrawGraph() : void  
**Description**: Set the graph class data from the GraphDlg data and draw.

**Class**: **GraphDlg**  
**Function**: SetFrameData() : void  
**Description**: Load the current frame's data into the graph class and display.

**Class**: **GraphDlg**  
**Member**: m_size : CSize  
**Description**: Sets the size of the dialog window. Assign before calling DoModal (unused if Create() is used to instantiate the dialog).

**Class**: **GraphDlg**  
**Member**: m_graphType : CGraph::enumGraphType  
**Description**: Sets the graph type (see cgraph.h)

**Class**: **GraphDlg**  
**Member**: m_graphStyle : CGraph::enumGraphStyle  
**Description**: Sets the graph style (see cgraph.h)

**Class**: **GraphDlg**  
**Member**: m_linePattern : int *  
**Description**: If non-Null, then sets CGraph's PatternData (see cgraph.h)

**Class**: **GraphDlg**  
**Member**: m_colors : int *  
**Description**: If non-Null, then sets CGraph's ColorData (see cgraph.h)
Class: GraphDlg
Member: m_useColors : BOOL
Description: If TRUE, then assign m_colors to CGraph's ColorData.

Class: GraphDlg
Member: m_overlayColors : int *
Description: If non-Null, then sets CGraph's OverlayColor (see cgraph.h)

Class: GraphDlg
Member: m_overlayColors : int
Description: Sets CGraph's ThickLines (see cgraph.h)

Class: GraphDlg
Member: m_currentFrame : int
Description: SetFrameData() uses this member to index the current frame data.

Class: GraphDlg
Member: m_nFrames : int
Description: The number of frames of data. Each frame consists of a complete graph's worth of data.

Class: GraphDlg
Member: m_nSets : int
Description: The number of sets of data. Each set is a data plot on the graph.

Class: GraphDlg
Member: m_setData : double***
Description: The Y data values. Index by [iframe][iset][icol]

Class: GraphDlg
Member: m_setDist : double***
Description: Optional X data values. Index by [iframe][iset][icol]. Only read if m_useDistData set to TRUE.

Class: GraphDlg
Member: m_setDist : double***
Description: Optional missing data values. Index by [iframe][iset][icol]. Always read but defaults to 0 (not missing).

Class: GraphDlg
Member: m_setTitle : CString*
Description: Graph title for each frame.

Class: GraphDlg
Member: m_useDistData : BOOL
Description: If true, then allocate space for the distance array and assign it to CGraph.

Class: GraphDlg
Member: m_overlayGraphType : int
Description: Assign to CGraph::OverlayGraphType. Only used if m_nOverlaySets > 0.
1.0 Routine List by Source File

Class:          GraphDlg
Member:         m_overlayGraphStyle : int
Description:    Assign to CGraph::OverlayGraphStyle. Only used if m_nOverlaySets > 0.

Class:          GraphDlg
Member:         m_nOverlaySets : int
Description:    Assign to CGraph::OverlayNumSets. Note that the calling routine must handle all data
allocs for m_overlaySetData and m_overlaySetDist, but the destructor will free any memory.

Class:          GraphDlg
Member:         m_overlaySetData : double***
Description:    Assign to CGraph::OverlayData (Y data points). Note that the calling routine must
allocate the memory for this memory, but the destructor will free it when the graph is
closed. Index by [iframe][iset][icol].

Class:          GraphDlg
Member:         m_overlaySetData : double***
Description:    Assign to CGraph::OverlayXPosData, optional X data points. Note that the calling
routine must allocate the memory for this memory, but the destructor will free it when
the graph is closed. Index by [iframe][iset][icol].

Class:          GraphDlg
Member:         m_overlaySetMissing : int***
Description:    Assign to CGraph::OverlayExtraData, optional missing data points. Note that the
calling routine must allocate the memory for this memory, but the destructor will free
it when the graph is closed. Index by [iframe][iset][icol].

Class:          GraphDlg
Member:         m_overlaySetTitle : CString
Description:    Assign to CGraph::RightTitle. Note this is always used even no overlay data are
defined. Currently implementation uses one right title for every frame.

Class:          GraphDlg
Member:         m_overlayTitleStyle : int
Description:    Assign to CGraph::RightTitleStyle. Note this is always used even no overlay data are
defined.

Class:          GraphDlg
Member:         m_overlaySetTitle : CString*
Description:    Appended to CGraph::LegendText (after the regular data titles have been assigned).

Class:          GraphDlg
Member:         m_useOverlayDistData : BOOL
Description:    If true, then DrawGraph will assign m_overlaySetDist to CGraph::OverlayXPosData.

Class:          GraphDlg
Member:         m_nCols : int
Description:    Number of X points.
Class: `GraphDlg`
Member: `m_nLabels` : int
Description: If non-zero, SetDimensions will allocated memory for `m_labels`.

Class: `GraphDlg`
Member: `m_labels` : CString*
Description: Assigns to CGraph::LabelText if non-NULL.

Class: `GraphDlg`
Member: `m_useDataLabels` : BOOL
Description: Sets whether or not to use the `m_dataLabels` member.

Class: `GraphDlg`
Member: `m_dataLabels` : CString***
Description: By allocating memory to this member, SetFrameData will assign CGraph::DataLabelText. Index using [iframe][iset][icol]. The destructor will take care of freeing the memory.

Class: `GraphDlg`
Member: `m_XAxisTicks` : int
Description: Assigns to CGraph::XAxisTicks

Class: `GraphDlg`
Member: `m_XAxisTicksMinor` : int
Description: Assigns to CGraph::XAxisMinorTicks

Class: `GraphDlg`
Member: `m_tickStyle` : CGraph::enumTickStyle
Description: Assigns to CGraph::TickStyle

Class: `GraphDlg`
Member: `m_labelXDateStart` : CString
Description: Assigns to CGraph::LabelXDateStart if non-empty. Note that currently CGraph::LabelXFormat will be "mmm". CGraph::LabelXType is 1.

Class: `GraphDlg`
Member: `m_labelXDateInc` : CString
Description: Assigns to CGraph::LabelXDateInc if non-empty. Note that currently CGraph::LabelXFormat will be "mmm". CGraph::LabelXType is 1.

Class: `GraphDlg`
Member: `m_labelXEvery` : int
Description: Assigns to CGraph::LabelEvery if non-empty.

Class: `GraphDlg`
Member: `m_showYAxisTicksLeft` : BOOL
Description: If TRUE, then `m_YAxisTicksLeft` will be assigned to CGraph::YAxisTicks[CGraph::yAxisLeft].
1.0 Routine List by Source File

Class: GraphDlg
Member: m_YAxisTicksLeft : int
Description: The number of left Y axis ticks to use. This will also be the dimension of m_YAxisTextLeft is Y text labels are used.

Class: GraphDlg
Member: m_YAxisTicksMinorLeft : int
Description: Assigns to CGraph::YAxisMinorTicks[CGraph::yAxisLeft]. Note that the sign will be reversed (not sure why this needs to be so). Defaults to 0.

Class: GraphDlg
Member: m_YAxisTicksRight : int
Description: Assigns to CGraph::YAxisTicks[CGraph::yAxisRight]. Defaults to 0.

Class: GraphDlg
Member: m_YAxisTicksMinorRight : int
Description: Assigns to CGraph::YAxisMinorTicks[CGraph::yAxisRight]. Note that the sign will be reversed (not sure why this needs to be so). Defaults to 0.

Class: GraphDlg
Member: m_YAxisTextLeft : CString*
Description: Assigns to CGraph::YLabelText[CGraph::yAxisLeft] if non-NULL. The dimension must be equal to m_YAxisTicksLeft

Class: GraphDlg
Member: m_YAxisTextRight : CString*
Description: Assigns to CGraph::YLabelText[CGraph::yAxisRight] if non-NULL. The dimension must be equal to m_YAxisTicksRight

Class: GraphDlg
Member: m_windowTitle : CString*
Description: The window title that should be displayed for the current frame. There must be one for each frame.

Class: GraphDlg
Member: m_graphTitle : CString*
Description: The graph title that should be displayed for the current frame. There must be one for each frame.

Class: GraphDlg
Member: m_XTitle : CString*
Description: The X axis title that should be displayed for the current frame. There must be one for each frame.
Class: **GraphDlg**
Member: `m_YTitle` : CString*
Description: The left Y axis title that should be displayed for the current frame. There must be one for each frame.

Class: **GraphDlg**
Member: `m_backgroundColor` : CGraph::enumColor
Description: The background color of the graph.

Class: **GraphDlg**
Member: `m_legendPos` : CGraph::enumLegendPos
Description: Assigns to CGraph::LegendPos. Defaults to bottom of screen.

Class: **GraphDlg**
Member: `m_xAxisMin` : double
Description: Assigns to CGraph::XAxisMin. Only used if > -10000. Also causes CGraph to use XAxisStyle = CGraph::userDefined

Class: **GraphDlg**
Member: `m_xAxisMax` : double
Description: Assigns to CGraph::XAxisMax. Only used if > -10000. Also causes CGraph to use XAxisStyle = CGraph::userDefined.

Class: **GraphDlg**
Member: `m_yAxisMinLeft` : double
Description: Assigns to CGraph::YAxisMin[CGraph::yAxisLeft]. Only used if > -10000. Also causes CGraph to use YAxisStyle[CGraph::yAxisLeft] = CGraph::userDefined.

Class: **GraphDlg**
Member: `m_yAxisMaxLeft` : double
Description: Assigns to CGraph::YAxisMax[CGraph::yAxisLeft]. Only used if > -10000. Also causes CGraph to use YAxisStyle[CGraph::yAxisLeft] = CGraph::userDefined.

Class: **GraphDlg**
Member: `m_yAxisMinRight` : double
Description: Assigns to CGraph::YAxisMin[CGraph::yAxisRight]. Only used if > -10000. Also causes CGraph to use YAxisStyle[CGraph::yAxisRight] = CGraph::userDefined.

Class: **GraphDlg**
Member: `m_yAxisMaxRight` : double
Description: Assigns to CGraph::YAxisMax[CGraph::yAxisRight]. Only used if > -10000. Also causes CGraph to use YAxisStyle[CGraph::yAxisRight] = CGraph::userDefined.

Class: **GraphDlg**
Member: `m_useGridX` : BOOL
Description: Adds CGraph::vertical to CGraph::GridStyle. Displays a vertical grid at every X tick.

Class: **GraphDlg**
Member: `m_useGridY` : BOOL
Description: Adds CGraph::horizontal to CGraph::GridStyle. Displays a horizontal grid at every Y tick.
1.0 Routine List by Source File

Class: GraphDlg
Member: m_doNotDelete : BOOL
Description: If FALSE, then the dialog will delete itself when closed. Otherwise it will wait for the calling program to do so (used when GraphControlDlg is driving the graphs).

Class: GraphDlg
Function: StartAnimation(int elapseVal) : void
Description: Installs the timeout proc and starts timeseries graphing.
Input Args: elapseVal: the time in milliseconds to advance each frame.

Class: GraphDlg
Function: AdvanceFrame() : void
Description: Called to advance the frame manually instead of using a mouse click or timer. Uses m_currentFrame.

11. Source File: ListDialog.h

Class: ListDialog
Function: ListDialog(CString title, CStringList &list, CWnd* pParent = NULL) : Constructor
Description: A dialog for choosing an item for a list. m_resultStr contains the result of the selection and m_listIdx contains the index in 'list'.
Input Args: title: Dialog window's title.
Input Args: list: List of strings to display.

Class: ListDialog
Member: m_listIdx : int
Description: When OK is selected, this will be assigned to the index of the listbox selection.

Class: ListDialog
Member: m_resultStr : CString
Description: When OK is selected, this will be assigned to the listbox entry that was selected.

Class: ListDialog
Member: m_title : CString
Description: The title to display in the dialog frame.

Class: ListDialog
Member: m_listData : CStringList
Description: The items to display in the listbox.

12. Source File: ProjectInfoPage.h

Class: ProjectInfoPage
Function: ProjectInfoPage() : Constructor
Description: The project data tab.
Class: ProjectInfoPage
Function: SetDoc(CGDPDoc *pDoc) : void
Description: A pointer to the document so project data can be accessed directly.
Input Args: pDoc: Pointer to the project document.

Class: ProjectInfoPage
Function: UpdateAll() : void
Description: Refresh the project input file display.

Class: ProjectInfoPage
Member: m_pDoc : CGDPDoc*
Description: A pointer to the document so project data can be accessed directly.

13. Source File: SpreadsheetAddDlg.h

Class: SpreadsheetAddDlg
Function: SpreadsheetAddDlg(CGDPDoc* pDoc, CWnd* pParent = NULL) : Constructor
Description: Displays a dialog for entering spreadsheet import options.
Input Args: pDoc: Pointer to the project document.

Class: SpreadsheetAddDlg
Function: UpdateParameterList() : void
Description: Refresh the list of parameters in the project.

Class: SpreadsheetAddDlg
Function: UpdateUI() : void
Description: Certain options are not allowed, such as using ',,' as a separator to read the clipboard (only tabs are allowed).

Class: SpreadsheetAddDlg
Member: m_fileNameList: CStringList
Description: List of file names selected.

Class: SpreadsheetAddDlg
Member: m_paramName: CString
Description: The parameter (if selected) to use for the new station data. Assigned in OnOK using the parameter listbox selection.

Class: SpreadsheetAddDlg
Member: m_paramName: CString
Description: The units (if selected) to use for the new station data. Assigned in OnOK using the parameter listbox selection.

Class: SpreadsheetAddDlg
Member: m_allParameterList: CStringList
Description: The list of parameters in the project.
1.0 Routine List by Source File

14. Source File: UnitsData.h

Class: SpreadsheetAddDlg
Function: UnitsData() : Constructor
Description: Class for holding information about default units data. Conversion equation is:
new_unit_val = old_unit_val * mult + offset.

Class: SpreadsheetAddDlg
Function: ReadFile(CString filename) : BOOL
Description: Load units data from file.
Input Args: filename: The name of the file to read default units data from.
Returns: TRUE if successful.

Class: SpreadsheetAddDlg
Function: WriteFile(CString filename) : BOOL
Description: Write units data to file.
Input Args: filename: The name of the file to write default units data to.
Returns: TRUE if successful.

Class: SpreadsheetAddDlg
Function: AddNewEntry() : void
Description: Create space for a new default units entry.

Class: SpreadsheetAddDlg
Function: DeleteEntry(int index) : void
Description: Create space for a new default units entry.
Input Args: index: The row number of the default units to remove.
Returns: TRUE if successful.

Class: SpreadsheetAddDlg
Member: m_multiArray : CArray<double, double>
Description: One item for each default units entry; the multiplier to use for the unit conversion.

Class: SpreadsheetAddDlg
Member: m_offsetArray : CArray<double, double>
Description: One item for each default units entry; the offset to use for the unit conversion.

Class: SpreadsheetAddDlg
Member: m_nameArray : CArray<CString, LPCSTR>
Description: One item for each default units entry; the name to use for the unit conversion.

15. Source File: UnitsPage.h

Class: UnitsPage
Function: UnitsPage() : Constructor
Description: The units data tab.
Class: UnitsPage
   Function: SetDoc(CGDPDoc *pDoc) : void
   Description: A pointer to the document so project data can be accessed directly.
   Input Args: pDoc: Pointer to the project document.

Class: UnitsPage
   Function: UpdateAll() : void
   Description: Refresh the project input file display.

Class: UnitsPage
   Function: BuildUnitsTable() : void
   Description: Populate m_unitsTable with current units data for the project.

Class: UnitsPage
   Function: BuildConversionTable() : void
   Description: Populate m_conversionTable with default units for the project.

Class: UnitsPage
   Function: SetApplyButtonLabel() : void
   Description: Assigns the button label to the default conversion apply button indicating which default conversion will be used for the currently selected entry in the project units table.

Class: UnitsPage
   Member: m_unitsTable : TableListCtrl
   Description: The table for display and edit of units in the project.

Class: UnitsPage
   Member: m_conversionTable : TableListCtrl
   Description: The table for display and edit of default units that can be applied to units in the project.
2.0 Function Index Arranged by Class

A

AggNameChoiceDlg
AggNameChoiceDlg 1

AggregationPage
AggregationPage 1
InitializeListBoxes 1
SetDoc 1
UpdateAll 1

C

CGDPDoc
AddInputFiles 23
ReadSpreadsheetClipboard 23
ReadSpreadsheetFile 23
UpdateAll 23

D

DataAvailPage
BuildDayCombos 2
BuildReportTable 3
BuildRightListBox 2
CopyToClipboard 3
DataAvailPage 2
GetEndDate 3
GetStartDate 3

InitializeDateControls 2
InitializeListBoxes 2
SetDoc 2
SetGraphRadioSensitivity 3
UpdateAll 2
UpdatePlotUI 3
UpdateReportTable 3

DataEditPage
BuildMainTable 4
DataEditPage 3
EndDateEdit 4
FillDataTable 4
FillMainTable 4
SetDoc 4
UpdateAll 4

DeleteItemsDlg
DeleteItemsDlg 5
ExportOptionsDlg 5
SetData 5

G

GDPFieldFile
GDPFieldFile 8
GenerateDataFiles 10
GetDateSpan 10
GetFieldRecFromStationAndParam 9
MonthStringToInt 10
ReadFile 8
ReadSpreadsheetClipboard 8
ReadSpreadsheetFile 8
ReadSpreadsheetStreamMulti
ReadSpreadsheetStreamSingle
ReadStoret
ReadWastore

**GDPFieldManager**
AddAggregationItem
AddNewParameter
AddNewStation
AddStationData
BuildParamReferences
BuildStationReferences
ChangeParameterForStation
ChangeParamName
ChangeStationName
ChangeUnitsForParam
DeleteParameters
DeleteStationData
DeleteStations
DelParamRec
DelStationRec
GDPFieldManager
GenerateDataFiles
GetAggItemObjectList
GetAggObjectList
GetAllObjectList
GetAssociatedObjects
GetCombinedObjectList
GetDataByParameter
GetDataByStation
GetDateSpan
GetFieldRecFromStationAndParam
GetFileNameArray
GetLastFieldRecordIndex
GetParamRec
GetStationRec
GetUnits
GetUnitsList
GetUnitsRec
HasParamName
HasStationName
InitializeParamList
InitializeStationList
InitializeUnitsList

ReadFile
ReadFiles
ReadProject
ReadSpreadsheetClipBoard
ReadSpreadsheetFile
RebuildAliasMap
RemoveAggregationItem
SearchRecords
SetStationRec
SetUnitsRec
StartAggEdits
WeedUnitsList
WriteProject

**GDPFieldRec**
AddData
GDPFieldRec
GenerateDataFile
GetDateSpan
ReplaceDataUnsorted
Sort

**GDPStationRec**
GDPStationRec

**GDPUnitsRec**
GDPUnitsRec

**GDPViewSheet**
GDPViewSheet
Resize
UpdateAll

**GraphDlg**
~GraphDlg
AdvanceFrame
DrawGraph
GraphDlg
SetDimensions
SetFrameData
SetTitles

**L**

**ListDialog**
ListDialog
## 2.0 Function Index Arranged by Class

### P

**ProjectInfoPage**
- ProjectInfoPage 31
- SetDoc 32
- UpdateAll 32

### S

**SpreadsheetAddDlg**
- SpreadsheetAddDlg 32
- UpdateParameterList 32
- UpdateUI 32

### U

**UnitsData**
- AddNewEntry 33
- DeleteEntry 33
- ReadFile 33
- UnitsData 33
- WriteFile 33

**UnitsPage**
- BuildUnitsTable 34
- SetApplyButtonLabel 34
- SetDoc 34
- UnitsPage 33
- UpdateAll 34
2.0 Member Index Arranged by Class

A
AggregationPage
  m_aggListBoxL 1
  m_aggObjectAliasList 2
  m_allObjectList 1, 2
  m_objectIndexMap 2

C
CGDPDoc
  m_gdpFieldManager 23
  m_unitsData 23
  m_unitsFilename 23

D
DataAvailPage
  m_aggObjectList 3
  m_reportTable 3
DataEditPage
  m_allObjectList 4
  m_dataTable 4
  m_isModified 4
  m_mainTable 4
  m_pFieldRec 4
DeleteItemsDlg
  m_itemArray 5

G
GDPFieldFile
  m_fileName 10
  m_stationArray 10
GDPFieldManager
  m_currentAggEditType 22
  m_fieldFileArray 17
  m_paramAliasMap 17
  m_paramMap 17
  m_stationAliasMap 17
  m_stationMap 16
  m_unitsAliasMap 17
  m_unitsMap 17
GDPFieldRec
  m_isUnsorted 7
  m_maskArray 7
  m_mult 8
  m_offset 7
  m_parameter 7
  m_readingTimeArray 7
  m_stationLoc 7
  m_stationName 7
  m_units 7
  m_valueArray 7
GDPParamRec
  m_aggregateParamList 12
  m_alias 11
  m_isHeadOfAggregate 12
  m_isPartOfAggregate 12
m_paramName 11
m_recordList 12
m_units 11

GDPStationRec
m_aggregateStationList 11
m_alias 11
m_isHeadOfAggregate 11
m_isPartOfAggregate 11
m_recordList 11
m_stationName 11

GDPUnitsRec
m_alias 12
m_comment 12
m_mult 12
m_name 12
m_offset 12

GDPViewSheet
m_aggregationPage 24
m_dataAvailPage 24
m_dataEditPage 24
m_projectInfoPage 24
m_unitsPage 24

GraphDlg
m_backgroundColor 30
m_colors 25
m_currentFrame 26
m_dataLabels 28
m_doNotDelete 31
m_graphStyle 25
m_graphTitle 29
m_graphType 25
m_labels 28
m_labelXDataInc 28
m_labelXDataStart 28
m_labelXEvery 28
m_legendPos 30
m_linePattern 25
m_listIdx 31
m_nCols 27
m_nFrames 26
m_nLabels 28
m_nOverlaySets 27
m_nSets 26
m_overlayColors 26
m_overlayGraphStyle 27
m_overlayGraphType 26
m_overlaySetData 27
m_overlaySetMissing 27
m_overlaySetTitle 27
m_overlayTitle 27
m_overlayTitleStyle 27
m_setData 26
m_setDist 26
m_setTitle 26
m_showYAxisTicksLeft 28
m_showYAxisTicksRight 29
m_size 25
m_tickStyle 28
m_useColors 26
m_useDataLabels 28
m_useDistData 26
m_useGridX 30
m_useGridY 30
m_useOverlayDistData 27
m_windowTitle 29
m_xAxisMax 30
m_xAxisMin 30
m_XAxisTicks 28
m_XAxisTicksMinor 28
m_XTitle 29
m_yAxisMaxLeft 30
m_yAxisMaxRight 30
m_yAxisMinLeft 30
m_yAxisMinRight 30
m_YAxisTextLeft 29
m_YAxisTextRight 29
m_YAxisTicksLeft 29
m_YAxisTicksMinorLeft 29
m_YAxisTicksMinorRight 29
m_YAxisTicksRight 29
m_YTitle 30
2.0 Member Index Arranged by Class

L

ListDialog
   m_listData 31
   m_pDoc 32
   m_resultStr 31
   m_title 31

S

SpreadsheetAddDlg

m_allParameterList 32
m_fileNameList 32
m_multArray 33
m_nameArray 33
m_offsetArray 33
m_paramName 32

U

UnitsPage
   m_conversionTable 34
   m_unitsTable 34