Revision 3

Protocol:

GraphRightServer

Declared In: GraphRight_API.h

Class Description

GraphRightServer protocol manages communication between the client and a running copy of GraphRight.

Instance Variables

None

Method Types

Registration Methods	GR_registerWithServer:forName:onHost:
Working with Notebooks	GR_createNewNoteBook GR_loadNoteBook: GR_loadASCIIFile:cellDelim:rowDelim:ignore:
Event Methods	GR_quitAndAllowSave: GR_hide GR_makeFront
Information Methods	GR_numberOfNotebooks GR_currentNotebooks GR_getOpenNotebook:

Instance Methods

- (int)**GR_registerWithServer**:sender
forName:(const char *)name
onHost:(const char *)host

Register client with a running copy of GraphRight on a specified host. If the client wishes to receive client events (described in GraphRightClient) then the name of the client served object root name. GraphRight will then send messages relating to user activities within GraphRight to the client (ie: such as the user closing a window, changing the value of a cell in the Data Table).

If the client wishes to receive client messages it must also register itself with an NXConnection first.

```
grClient = [NXConnection registerRoot:self
withName:"GraphRightClient"];
[grClient registerForInvalidationNotification:self];
[grClient runFromAppKit];
```

Where "GraphRightClient" would be replaced with your application specific name.

Returns:

NO_ERROR	If successful.
TOO_MANY_CLIENTS	If another client is already attached to that copy of GraphRight.
SERVER_LOCKED	If the copy of GraphRight is locked and will not accept a connection.
NO_CLIENT_FOUND	If a client is specified and GraphRight can't find the named root object.

- **GR_createNewNotebook**

Creates a new data notebook and returns a Proxy to it. On failure it returns nil.

- **GR_loadNotebook**:(const char *)filename

Loads the notebook specified by filename. "filename" should be a complete path to the .gr file.

```
[grServer GR_loadNotebook:"/Users/foo/bar.gr"]
```

Returns a Proxy to the new notebook if successful, nil if not.

- **GR_loadASCIIFile**:(const char *)filename
  cellDelim:(const char[2])cellDelim
  rowDelim:(const char[2])rowDelim
  ignore:(const char[3])ignored

Creates a new notebook from the ASCII data provided in filename. You can specify the cell delimiter and row delimiter. Up to two characters can also be specified to be ignored. For the following kinds of files set:

Source	Setting
Unix	rowDelim = "\n"
Macintosh	rowDelim = "\r"
DOS	rowDelim = "\n" ignored = "\r"

Returns a Proxy to the new Notebook if successful, nil if not.

- (int)**GR_quitAndAllowSav**e:(BOOL)allowSave

Terminates GraphRight and allows for the user to save any documents that have been changed. If allowSave is NO then GraphRight is just forced to quit.

- (int)**GR_hide**

Hides GraphRight as though the user has pressed Command-h. Returns NO_ERROR if there is no problem.

- (int)**GR_makeFront**

Un-hides GraphRight, makes it the topmost application. Returns NO_ERROR if there is no problem.

- (int)**GR_numberOfNotebooks**

  Returns the number of currently open Notebooks.

- **GR_currentNotebooks**

  Returns a Proxy to a List object containing all the currently open notebooks. Do not free or modify the List in any way. DataDocumentServer methods allow manipulation of the notebooks returned

- **GR_getOpenNotebook**:(const char *)filename

  Returns the notebook specified by filename. Returns nil if filename is not already loaded.

Protocol

DataDocumentServer

Declared In: GraphRight_API.h

Class Description

DataDocuementServer protocol manages communication between a Notebook of data and the client.

Instance Variables

None

Method Types

Manipulating Notebooks	GR_importASCIIFile: GR_writeToASCIIFile: GR_save GR_close GR_setPath: GR_filename
Manipulating Cells	GR_setCell::string: GR_setCell::val: GR_setCellinDataSet:withLabel:value: GR_setCells: GR_setSeriesTitle:forSeries: GR_getCell:
Graph Documents	GR_selectRange:bottom: GR_createGraphType: GR_addGraphType:to: GR_getGraphWithName: GR_getGraphs

Ordering on and off screen

GR_miniaturizeDocument
GR_maximiseDocument
GR_makeFront

Local Only Methods

GR_pasteDataFromNamedPasteBoard:

Instance Methods

- (int)**GR_importASCIIFile**:(const char *)filename

Imports the ASCII file "filename" into the Notebook starting with the top row and left most column. Cell delimiters can be either tabs or commas, row delimiters can be Line-feed or semi-colons. This action is the same as Paste:. Returns NO_ERROR if successful.

- (int)**GR_writeToASCIIFile**:(const char *)filename

Not currently implemented. (Slated for 1.2)

- (int)**GR_setCell**:(int)r :(int)c **string**:(in const char *)string

Sets the cell at r and c to the string value in string. It will make a copy of the data in string, so string should be freed upon completion. Returns NO_ERROR on success. One thing to note is that the top most left cell is 0,0 and that row 0 and column 0 are reserved for labels.

- (int)**GR_setCell**:(int)r :(int)c **val**:(in float)val

Sets the cell at r and c to the float value in val. Returns NO_ERROR on success. One thing to note is that the top most left cell is 0,0 and that row 0 and column 0 are reserved for labels.

- (int)**GR_setCellinDataSet:**(const char *)dataSet
withLabel:(const char *)dataLabel
value:(float)val

Sets the float value of the cell in data set and data label, finds the

first reference that matches both and sets that cell's value.

Thus to set cell 3,1 to 600.

```
[servedNotebook GR_setCellinDataSet:"Sales"
                           withLabel:"Q3"
                               value:600];
```

Returns NO_ERROR if successful.

- (int)GR_setCells:(NXData *)data

This is probably the most useful method for sending data into a Data Table. With this method you can send a whole range of data (that can be discontinuous) into a Data Table. The data is made up of GRCells and should be encoded as follows:
The first thing that needs to be written is the number of cells that are being sent over to the Data Table, then each GRCell must be encoded.

```
- sendChunckOfData:sender
{
    int i,j;
    int n = 20;
    int m = 6;
    GRCell cell;
    NXTypedStream *ts;
    NXStream *stream = NXOpenMemory(NULL, 0, NX_WRITEONLY);
    char *buffer;
    int len, maxLen;
    id data;

    ts = NXOpenTypedStream(stream,  NX_WRITEONLY);
    i = m * n;
    NXWriteType(ts, "i", &i); /* Write the number of cells we are sending */
    for(i=0;i<n;i++)
    for(j=0;j<m;j++)
    {
       cell.r = i;
       cell.c = j;
       cell.isValue = YES;
       cell.val = random()/10000000;
       cell.string = NXCopyStringBuffer("");
/* This MUST be done, it seems that there is a bug in Distributed Objects when
 * sending a NULL string. OD's don't check for NULL and just try to do a
 * strlen(string) on the NULL, this crashes the client
```

```
    */
        NXWriteType(ts, "{iicf*}", &cell);
    }

    NXCloseTypedStream(ts);
    /* encapsule the data */
    NXGetMemoryBuffer(stream, &buffer, &len, &maxLen);
    data = [[NXData alloc] initWithData:buffer size:len dealloc:YES];
    /* send to client, NXData objects are automatically sent 'bycopy' */
    printf("Sending Data\n");
    [currentDoc GR_setCells:data]; /* currentDoc is a Proxy to a DataDocument */
    [data free];

    return NO_ERROR;
}
```

This method will be encapsulated more cleanly in the near future, perhaps as an IB palette.
Returns NO_ERROR if successful.

- (int)**GR_setSeriesTitle**:(const char *)title **forSeries**:(int)n

Sets the series title for data set n. Data sets are the horizontal entries in row 0.
Returns NO_ERROR if successful.

- (int)**GR_selectRange:**(GRAddress)tl **bottom**:(GRAddress)br

Selects the range specified by top left row and column and the bottom right row and column. Returns NO_ERROR if successful.

- **GR_createGraphType**:(int)graphType

Creates a new Graph Document from the currently selected cells. GraphType is one of the #defined types in GraphRight_API.h. Returns a Proxy to the new GraphDocument on success, nil on failure.

- **GR_addGraphType**:(int)graphType **to**:(const char*)graphName

Adds a new Graph to a Graph Document already created. If the Graph Document does not already exist it is created and behaves like GR_createGraphType:. Returns a Proxy to the new GraphDocument on success, nil on failure.

- **GR_getGraphWithName**:(const char*)graphName

Returns a Proxy to the named Graph Document.

- **GR_getGraphs**

Returns a Proxy to a List object containing all the Graph Documents that the Notebook owns. Do not free the list or destroy any elements in it.

- (int)**GR_setPath**:(const char *)path

Sets the file name and location of the Notebook. Returns NO_ERROR if successful.

- (const char*)**GR_filename**

Returns the complete path to the Notebook.

- (int)**GR_getCell**:(GRCell *)dataCell

Sets the values for the cell dataCell.

```
GRCell aCell;
aCell.r = 3;
aCell.c = 4;
[servedNotebook GR_getCell:&aCell];
```

aCell.isValue, aCell.val, and aCell.string will now be filled with the values of cell 3,4. Returns NO_ERROR on success.

- (int)**GR_save**

Saves the Notebook, if the notebook has not been saved before it will prompt the user for a file name. Returns NO_ERROR on success.

- **GR_close**

Closes the Notebook. This is will not send a GR_notebookWillClose message to the client.

- **GR_miniaturizeDocument**

Miniaturizes the Notebook.

- **GR_maximiseDocument**

Maximizes the Notebook.

- (int)**GR_makeFront**

Makes the Notebook the front window. Returns NO_ERROR on success.

- (int)**GR_pasteDataFromNamedPasteBoard**:(const char *)pasteboardName

This method is only valid if the client and server are running on the same machine. This allows the client to provide data to GraphRight via a named pasteboard. Data must either NXAsciiPboardType or NXTabularTextPboardType.

Protocol

GraphDocumentServer

Declared In: GraphRight_API.h

Class Description

GraphDocumentServer protocol manages communication between the client and a Graph Document.

Instance Variables

None

Method Types

Graph Document

GR_graphName
GR_numberOfGraphs
GR_setGraphPageFrame:
GR_setPaperOrientation:

Manipulating a Graph Page

GR_importImage:::
GR_setRichText:andPlaceAt::allowWidth:
GR_setGraphicSize:
GR_setGraphicLocation::
GR_setLabelFont: forAxis:
GR_setLabelPreString:forAxis:
GR_setLabelPostString:forAxis:

Output

GR_print:
GR_writePageToEPS:
GR_writePageToTIFF:

Instance Methods

- (const char *)**GR_graphName**

Returns the name of the GraphDocument.

- (int)**GR_numberOfGraphs**

Returns the number of Graphs that exist in the Graph Page.

- (int)**GR_setGraphPageFrame**:(NXRect *)newFrame

Sets the frame for the Graph Page.

- (int)**GR_setPaperOrientation**:(char)mode

Set the orientation for the Graph Page, with mode either NX_LANDSCAPE or NX_PORTRAIT. Returns NO_ERROR on success.

- **GR_importImage**:(const char *)imagePath :(float)x :(float)y

Imports a TIFF (.tiff) or EPS (.eps) file into the Graph Page at location x y. Both x and y are in inches and are measured from the bottom left corner. Borders are not counted, measurements are in absolute units from the edge of the paper. Returns a Proxy to an NXImage object work-a-like on success, nil on failure.

- **GR_setRichText**:(char *)data **andPlaceAt**:(float)x :(float)y **allowWidth**:(float)width

Imports Rich Text at location x y. Both x and y are in inches and are measured from the top left corner. Borders are not counted, measurements are in absolute units from the edge of the paper. If width = 0, then the text will be sized such that the first row fits on a single line, all subsequent rows will be wrapped if they are wider. If a width is given then all the text will be wrapped to that width. The

data should be raw RTF text, it should have a NULL at the end of the char array. Returns a Proxy to a Text object work-a-like on success, nil on failure.

- (int)**GR_setGraphicSize**:(NXSize *)newSize **forGraphic**:graphic

Sets the size of a Graphic on a Graph Page. Width and Height are both in inches. Returns NO_ERROR on success.

- (int)**GR_setLocation**:(float *)horizontalOffset
:(float *)verticalOffset
forGraphic:graphic

Sets the location of a graphic on a Graph Page. Both x and y are in inches and are measured from the bottom left corner. Borders are not counted, measurements are in absolute units from the edge of the paper. Returns NO_ERROR on success.

- (int)**GR_setLabelFont**:(Font *)newFont **forAxis**:(enum axisLocation)where

Sets the font for the label on the specified axis to newFont. Returns NO_ERROR on success.

- (int)**GR_setLabelPreString**:(const char *)string **forAxis**:(enum axisLocation)where

Sets the pre-string for the label on the specified axis to string. Returns NO_ERROR on success.

- (int)**GR_setLabelPostString**:(const char *)string **forAxis**:(enum axisLocation)where

Sets the post-sting for the label on the specified axis to string. Returns NO_ERROR on success.

- (int)**GR_print**:(BOOL)showPanel

Prints the Graph Page to the currently selected printer or to Preview.app depending on what was last selected. If showPanel = YES then the print panel will be shown on the machine that

GraphRight is running on. Returns NO_ERROR on success.

- (int)**GR_writePageToEPS**:(const char *)fileName

Writes the Graph Page to an EPS file, if the extension of the given file is not .eps it will be added. Returns NO_ERROR on success.

- (int)**GR_writePageToTIFF**:(const char *)fileName

Writes the Graph Page to an TIFF file, if the extension of the given file is not .tiff it will be added. Returns NO_ERROR on success.

Protocol

GraphRightClient

Declared In: GraphRight_API.h

Class Description

GraphRightClient protocol manages communication between GraphRight and the client.

Instance Variables

None

Method Types

GR_notebookWillClose:
GR_graphWillClose:fromNotebook:
GR_appWillTerminate
GR_cellChanged:

Instance Methods

- **GR_notebookWillClose**:notebook

When a user closes the window for notebook this message is sent to the client. This message is not sent when a notebook is closed via the API.

- **GR_graphWillClose**:graph **fromNotebook**:notebook

When a user closes the window for the graph in notebook this message is sent to the client. This message is not sent when a

graph is closed via the API.

- **GR_appWillTerminate**

  When the user quits GraphRight this message will be sent. If GraphRight is terminated via the API this message is not sent.

- **GR_cellChanged**:(GRCell)newCell in:notebook

  When the user changes the value of a cell in notebook this message is sent. When a cell's value is changed via the API this message is not sent.

Defines

```
#define NO_ERROR 0
#define BAD_FILE 1
#define FILE_NOT_FOUND 2

#define TOO_MANY_CLIENTS 0
#define SERVER_LOCKED 1
#define NO_CLIENT_FOUND 2

#define NO_SELECTED_CELLS 0
#define NO_VALID_TYPE 1

#define LABEL_NOT_FOUND 2
#define SET_NOT_FOUND 3

#define GR_COLUMN_CHART 0
#define GR_STACKEDCOLUMN_CHART 1
#define GR_BAR_CHART 2
#define GR_STACKEDBAR_CHART 3
#define GR_PIE_CHART 4
#define GR_AREA_CHART 5
#define GR_DENSITY_CHART 6
#define GR_HIGH_LOW_CHART 7
#define GR_HIGH_LOW_CLOSE_CHART 8
#define GR_HIGH_LOW_OPEN_CLOSE_CHART 9
#define GR_SCATTER_CHART 10
#define GR_LINE_CHART 11
#define GR_XY_CHART 12
#define GR_XYLINE_CHART 13

enum axisLocation{
left_axis,
right_axis,
top_axis,
bottom_axis,
x_axis,
y_axis,
};

typedef struct{
int r,c;
BOOL isValue;
float val;
char *string;
} GRCell;
```

```
typedef struct{
int r, c;
} GRAddress;
```