Hermitech Laboratory

Address:	*Chapaeva str., 7, Zhytomyr, 10029, Ukraine*
Phone:	*+38-0412-447214*
Fax:	*+38-0412-480198*
Web:	*http://www.mmlsoft.com*
E-mail:	*info@mmlsoft.com*

FORMULATOR v3.8

API

Developer Manual

Published By
Hermitech Laboratory
7, Chapaeva str.,
Zhytomyr, 10029
Ukraine

E-mail: *info@mmlsoft.com*
Web: *http://www.mmlsoft.com*

Copyright © 2003-2008 by Hermitech Laboratory. All rights reserved.

Table of Contents

INTRODUCTION 4
FORMULATOR API METHODS 5
- SetMathML 7
- GetMathML 7
- ExportEMF 7
- ExportHEMF 7
- ExportImage 7
- ExportHBitmap 7
- ExportXHTML 7
- ExportImage2Stream 8
- ExportImage2Array 8
- ExportImage2BitmapInfo 8
- SetBackgroundColor 9
- GetBackgroundColor 9
- SetShowNesting 9
- GetShowNesting 9
- SetShowReadOnly 9
- GetShowReadOnly 9
- SetLineSpacing 9
- GetLineSpacing 9
- SetScale 10
- GetScale 10
- SetTopIndent 10
- GetTopIndent 10
- SetBottomIndent 10
- GetBottomIndent 10
- SetLeftIndent 10
- GetLeftIndent 10
- SetRightIndent 10
- GetRightIndent 10
- SetShowInvisibleElements 11
- GetShowInvisibleElements 11
- SetStyleFaceName 11
- GetStyleFaceName 11
- SetStyleBold 12
- GetStyleBold 12
- SetStyleItalic 12
- GetStyleItalic 12
- SetStyleColor 12
- GetStyleColor 12
- SetExpressionColor 12
- SetSymbolSize 12
- GetSymbolSize 12

INTRODUCTION

Formulator is a dynamic and intelligent mathematical equation editor designed for personal computers running Microsoft® Windows. This application allows you to create mathematical equations through simple point-and-click techniques. Equations can be converted into MathML, or the other textual languages using a customizable translation mechanism, and can be saved in several graphic file formats, ready to be imported into documents. Formulator supports not only import into the MathML format, but also export from this approved standard for math on the World Wide Web.

For each basic mathematical construct, Formulator provides a template containing graphics and edit boxes. There are several template groups, including fractions, radicals, sums, matrices, various types of brackets and braces, etc. Users create equations by inserting templates and filling in their edit boxes. Users can insert templates into the edit boxes of other templates, and in that way complex hierarchical formulas can be built up.

The editing functionality provided by Formulator, when combined with the API of the Formulator component edition, can be used to implement a variety of mathematical applications that can be utilized in several different contexts. For example, an interface for computer algebra systems can be created that interchange MathML data, or multimedia materials with mathematical data can be created.

FORMULATOR API METHODS

The methods of the Formulator API are listed below.

Method Name	Method Description
Basic methods	
SetMathML	Imports the equation by the MathML 2.0 text storing in the argument.
GetMathML	Exports the current version of the equation that you're working on by saving it in the argument as MathML 2.0 text.
ExportEMF	Exports the current version of the equation that you're working on by saving it on disk into EMF (Enhanced Windows Metafile) *file*.
ExportHEMF	Exports the current version of the equation in the form of *handle in memory* to EMF (Enhanced Windows Metafile).
ExportImage	Exports the current version of the equation that you're working on by saving it on disk in a graphic *file*.
ExportHBitmap	Exports the current version of the equation in the form of *handle in memory* to Bitmap.
ExportXHTML	Exports the current version of the equation for publishing mathematics on web by saving it on disk into XHTML file.
ExportImage2Stream	Exports the current version of the equation that you're working on by storing the corresponding graphic file into a stream (for the further external GDI+ objects creation).
ExportImage2Array	Exports the current version of the equation that you're working on by storing the corresponding graphic file into the byte array.
ExportImage2BitmapInfo	Exports the current version of the equation that you're working on by storing the corresponding raw GDI object into the array (for the further external GDI+ objects creation).
Fine tuning methods	
SetBackgroundColor	Sets the background color of the equation.
GetBackgroundColor	Gets the background color of the equation.
SetShowNesting	Switch the equation display into nesting mode.
GetShowNesting	Switch the equation display from the nesting mode into normal viewing mode.
SetShowReadOnly	Switch the equation display into "show read-only nodes" mode.

GetShowReadOnly	Switch the equation display from the “show read-only nodes” mode into normal viewing mode.
SetLineSpacing	Changes default value of distance between neighbour lines (in %).
GetLineSpacing	Gets default value of distance between neighbour lines (in %).
SetScale	Changes the viewing scale (in %). Acceptable values of the viewing scale are: 100, 200, 300, 500, 1000.
GetScale	Gets the viewing scale (in %).
SetTopIndent	Sets the top indent for a document in pixels (px).
GetTopIndent	Gets the top indent for a document in pixels (px).
SetBottomIndent	Sets the bottom indent for a document in pixels (px).
GetBottomIndent	Gets the bottom indent for a document in pixels (px).
SetLeftIndent	Sets the left indent for a document in pixels (px).
GetLeftIndent	Gets the left indent for a document in pixels (px).
SetRightIndent	Sets the right indent for a document in pixels (px).
GetRightIndent	Gets the right indent for a document in pixels (px).
SetShowInvisibleElements	Sets the option whether to render invisible Content MathML elements.
GetShowInvisibleElements	Gets the option whether to render invisible Content MathML elements.
SetStyleFaceName	Allows to change the font name assigned to each style.
GetStyleFaceName	Allows to get the font name assigned to each style.
SetStyleBold	Allows to change the weight of the font assigned to each style.
GetStyleBold	Allows to get the weight of the font assigned to each style.
SetStyleItalic	Allows to change the italic property of the font assigned to each style.
GetStyleItalic	Allows to get the italic property of the font assigned to each style.
SetStyleColor	Allows to change the color of the font assigned to each style.
GetStyleColor	Allows to get the color of the font assigned to each style.
SetExpressionColor	Allows to change the color of entire mathematical expression.
SetSymbolSize	Sets the font size in points (pt) for predefined types of text.
GetSymbolSize	Gets the font size in points (pt) for predefined types of text.

SetMathML

Imports the equation by the MathML 2.0 text storing in the argument.

HRESULT SetMathML([in] BSTR bstrMathML);

GetMathML

Exports the current version of the equation that you're working on by saving it in the argument as MathML 2.0 text.

HRESULT GetMathML([out] BSTR pbstrMathML);*

ExportEMF

Exports the current version of the equation that you're working on by saving it on disk into EMF (Enhanced Windows Metafile) file.

HRESULT ExportEmf([in] BSTR bstrPath);

ExportHEMF

Exports the current version of the equation in the form of handle in memory to EMF (Enhanced Windows Metafile).

HRESULT ExportHEmf([out] OLE_HANDLE phValue);*

ExportImage

Exports the current version of the equation that you're working on by saving it on disk in a graphic file.

HRESULT ExportImage([in] BSTR bstrPath);

ExportHBitmap

Exports the current version of the equation in the form of handle in memory to Bitmap.

HRESULT ExportHBitmap([out] OLE_HANDLE phValue);*

ExportXHTML

Exports the current version of the equation for publishing mathematics on web by saving it on disk into XHTML file.

HRESULT ExportXHTML([in] BSTR bstrPath);

ExportImage2Stream

Exports the current version of the equation that you're working on by storing the corresponding graphic file into a stream (for the further external GDI+ objects creation).

Parameter *pUnknown* gives the entry point of the stream, *bstrType* defines the type of the graphic file.

HRESULT ExportImage2Stream([in] IUnknown pUnknown, [in] BSTR bstrType);*

ExportImage2Array

Exports the current version of the equation that you're working on by storing the corresponding graphic file into the byte array.

Parameter *bstrType* defines the type of the graphic file.

HRESULT ExportImage2Array([out] VARIANT pArray, [in] BSTR bstrType);*

ExportImage2BitmapInfo

Exports the current version of the equation that you're working on by storing the corresponding raw GDI object into the array (for the further external GDI+ objects creation).

HRESULT ExportImage2BitmapInfo([out] LONG plWidth, [out] LONG* plHeight, [out] LONG* plStride, [out] ULONG* pulPixelFormat, [out] VARIANT* pData);*

Parameters are the same, as needed for creation of the GDI+ Bitmap object:
plWidth – the width, in pixels, of the future Bitmap.
plHeight – the height, in pixels, of the new Bitmap.
plStride – integer that specifies the byte offset between the beginning of one scan line and the next.
pulPixelFormat – the PixelFormat enumeration for the new Bitmap.
pData – Pointer to an array of bytes that contains the pixel data.

The following example shows how this function can be used for the server-side C# web-application:

```
fmlapiLib.FmlObj myMathMlImage = new FmlObj();
string sEquationCode = "<math display = 'block'><mi>&pi;</mi></math>";
myMathMlImage.SetMathML(sEquationCode);

object o = new object();
int width, height, stride;
uint pxFormat;
myMathMlImage.ExportImage2BitmapInfo(out width, out height, out stride, out pxFormat, out o);
IntPtr srcData =
        System.Runtime.InteropServices.Marshal.UnsafeAddrOfPinnedArrayElement((byte[])o, 0);
```

```
Bitmap bm = new Bitmap(width, height, stride,
        (System.Drawing.Imaging.PixelFormat)pxFormat, srcData);
```

SetBackgroundColor

Sets the background color of the equation.

HRESULT SetBackgroundColor([in] OLE_COLOR clrValue);

GetBackgroundColor

Gets the background color of the equation.

HRESULT GetBackgroundColor([out] OLE_COLOR pclrValue);*

SetShowNesting
GetShowNesting

These methods toggle the equation display between normal viewing mode and nesting mode where the you can see the hierarchical structure of your equations.

HRESULT SetShowNesting([in] LONG nValue);
HRESULT GetShowNesting([out] LONG pnValue);*

SetShowReadOnly
GetShowReadOnly

These methods toggle the equation display between normal viewing mode and “show read-only nodes” mode where the you can see highlighted that nodes which can’t be changed.

HRESULT SetShowReadOnly([in] LONG nValue);
HRESULT GetShowReadOnly([out] LONG pnValue);*

SetLineSpacing

Changes default value of distance between neighbour lines (in %).

HRESULT SetLineSpacing([in] LONG nValue);

GetLineSpacing

Gets default value of distance between neighbour lines (in %).

HRESULT GetLineSpacing([out] LONG pnValue);*

SetScale

Changes the viewing scale (in %). Acceptable values of the viewing scale are: 100, 200, 300, 500, 1000.

HRESULT SetScale([in] LONG nValue);

GetScale

Gets the viewing scale (in %).

HRESULT GetScale([out] LONG pnValue);*

SetTopIndent
GetTopIndent

Gets or sets the top indent for a document in pixels (px).

HRESULT SetTopIndent([in] LONG nValue);
HRESULT GetTopIndent([out] LONG pnValue);*

SetBottomIndent
GetBottomIndent

Gets or sets the bottom indent for a document in pixels (px).

HRESULT SetBottomIndent([in] LONG nValue);
HRESULT GetBottomIndent([out] LONG pnValue);*

SetLeftIndent
GetLeftIndent

Gets or sets the left indent for a document in pixels (px).

HRESULT SetLeftIndent([in] LONG nValue);
HRESULT GetLeftIndent([out] LONG pnValue);*

SetRightIndent
GetRightIndent

Gets or sets the right indent for a document in pixels (px).

HRESULT SetRightIndent([in] LONG nValue);
HRESULT GetRightIndent([out] LONG pnValue);*

SetShowInvisibleElements
GetShowInvisibleElements

Gets or sets the option whether to render invisible Content MathML elements.

HRESULT SetShowInvisibleElements([in] LONG nValue);
HRESULT GetShowInvisibleElements([out] LONG pnValue);*

SetStyleFaceName
GetStyleFaceName

These methods allows to set/get the font name assigned to each style.

HRESULT SetStyleFaceName([in] LONG nStyle, [in] BSTR bstrMathML);
HRESULT GetStyleFaceName([in] LONG nStyle, [out] BSTR pbstrMathML);*

Method accepts one argument of type *long* that defines a new style and one argument for the font name.

Each character in a Formulator equation can be directly assigned a specific font and character style or can be of one of eleven styles. Each styles is defined as a combination of a font and character style. By changing the style of a text fragment, a user can quickly define its appearance and behavior rules and by changing the definition of a style, a user can quickly change the appearance of all the characters that use it.

The 11 styles available in Formulator are Text, Variable, Function, Greek, Vector-Matrix, Number, Fixed, Operator, Extra-Math, User 1, and User 2. Formulator assigns styles to certain kinds of characters automatically, based on its knowledge of mathematics and typesetting conventions. This intelligent assignment of styles is a useful feature of Formulator which can significantly simplify your work. Assigning style Math to a set of characters a user can explicitly define this intelligent behavior to be a rule for this characters.

```
#define STYLE_MATH          0
#define STYLE_TEXT          1
#define STYLE_VARIABLE      2
#define STYLE_FUNCTION      3
#define STYLE_GREEK         4
#define STYLE_VECTOR        5
#define STYLE_NUMBER        6
#define STYLE_FIXED         7
#define STYLE_OPERATOR      8
#define STYLE_EXTRAMATH     9
#define STYLE_USER1         10
#define STYLE_USER2         11
```

SetStyleBold
GetStyleBold

These methods allows to set/get the weight of the font assigned to each style. Meaning of arguments is the same as for SetStyleFaceName/ GetStyleFaceName.

HRESULT SetStyleBold([in] LONG nStyle, [in] LONG nValue);
HRESULT GetStyleBold([in] LONG nStyle, [out] LONG pnValue);*

SetStyleItalic
GetStyleItalic

These methods allows to set/get the italic property of the font assigned to each style. Meaning of arguments is the same as for SetStyleFaceName/ GetStyleFaceName.

HRESULT SetStyleItalic([in] LONG nStyle, [in] LONG nValue);
HRESULT GetStyleItalic([in] LONG nStyle, [out] LONG pnValue);*

SetStyleColor
GetStyleColor

These methods allows to set/get the color of the font assigned to each style. Meaning of arguments is the same as for SetStyleFaceName/ GetStyleFaceName.

HRESULT SetStyleColor([in] LONG nStyle, [in] OLE_COLOR clrValue);
HRESULT GetStyleColor([in] LONG nStyle, [out] OLE_COLOR pclrValue);*

SetExpressionColor

This method allows to set the color of the entire mathematical expression that is contained in the current document. It applies the given color to all existing styles and graphics (such as, lines, arcs, etc.).

HRESULT SetExpressionColor([in] OLE_COLOR clrValue);

SetSymbolSize
GetSymbolSize

These methods allows to set/get the font size in points (pt) for predefined types of text.

HRESULT SetSymbolSize([in] LONG nType, [in] LONG nValue);
HRESULT GetSymbolSize([in] LONG nType, [out] LONG pnValue);*

Method accepts one argument of type *long* that defines the predefined type of a text and one argument of type *long* for the actual value of font size.

Formulator provides default font sizes for each edit box in a mathematical expression. There are four reserved cases of changing font size:

- large symbols (like sums, products, integrals, etc.) (*nType must be 0*);
- regular text (*nType must be 1*);
- subscript/superscript text (*nType must be 2*);
- subscript/superscript nested in the subscript/superscript text, i.e. subscript/superscript of the next level (*nType must be 3*).