

Customizing ArcIMS™

ArcIMS™ 3

Java™ Viewer



Copyright © 2000 Environmental Systems Research Institute, Inc.

All rights reserved.

Printed in the United States of America.

The information contained in this document is the exclusive property of Environmental Systems Research Institute, Inc. This work is protected under United States copyright law and the copyright laws of the given countries of origin and applicable international laws, treaties and/or conventions. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage or retrieval system, except as expressly permitted in writing by Environmental Systems Research Institute, Inc. All requests should be sent to Attention: Contracts Manager, Environmental Systems Research Institute, Inc., 380 New York Street, Redlands, CA 92373-8100, USA.

The information contained in this document is subject to change without notice.

U.S. GOVERNMENT RESTRICTED/LIMITED RIGHTS

Any software, documentation, and/or data delivered hereunder is subject to the terms of the License Agreement. In no event shall the U.S. Government acquire greater than RESTRICTED/LIMITED RIGHTS. At a minimum, use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in FAR §52.227-14 Alternates I, II, and III (JUN 1987); FAR §52.227-19 (JUN 1987) and/or FAR §12.211/12.212 (Commercial Technical Data/Computer Software); and DFARS §252.227-7015 (NOV 1995) (Technical Data) and/or DFARS §227.7202 (Computer Software), as applicable. Contractor/Manufacturer is Environmental Systems Research Institute, Inc., 380 New York Street, Redlands, CA 92373-8100, USA.

ESRI and the ESRI globe logo are trademarks of Environmental Systems Research Institute, Inc., registered in the United States and certain other countries; registration is pending in the European Community. ArcGIS, ArcIMS, ArcSDE, ArcExplorer, GIS by ESRI, and the ArcIMS logo are trademarks and www.esri.com and @esri.com are service marks of Environmental Systems Research Institute, Inc. Netscape and the Netscape N logo are registered trademarks of Netscape Communications Corporation in the United States and other countries. Microsoft and the Windows logo are registered trademarks and the Microsoft Internet Explorer logo is a trademark of Microsoft Corporation. Other companies and products mentioned herein are trademarks or registered trademarks of their respective trademark owners.

Contents

1 Introducing the Java Viewer 5

- What is the Java Viewer? 6
- Java Viewer file organization 8
- Java Viewer frames 10

2 Customizing the Java Viewer 12

- Changing layers of a MapService 13
- Opening files: changing the order and removing files 14
- Changing title, logo, and background 15
- Loading the MapService, Overview Map, and Toolbar 16
- Setting active and visible layers 18
- Setting the right-click menu 19
- Setting MapTips 20
- Setting visibility and alias for results 21
- Setting the scale bar units 22
- Setting Folders for MapNotes and EditNotes 23
- Setting colors for the applets 24
- Setting stored queries 25
- Adding a tool to the toolbar 26
- Using hyperlinks 28
- Using the sample Java Viewers 30

3 Java Viewer Object Model 31

- CallOutMarkerSymbol 32
- Collection 54
- Color 59
- Extent 60
- GradientFillSymbol 65
- GroupRenderer 76
- HashLineSymbol 84
- IMSMMap 101
- IMSOOverviewMap 244
- IMSScaleBar 260
- IMSToc 270
- IMSToolBar 281
- LabelRenderer 282
- Layer 288
- LineSymbol 325

MarkerSymbol 341
NameValuePair 357
PolygonSymbol 362
RasterFillSymbol 382
RasterMarkerSymbol 391
RasterShieldSymbol 410
Renderer 437
ScaleDependentRenderer 438
ShieldSymbol 445
SimpleRenderer 463
TextSymbol 467
TrueTypeMarkerSymbol 489
ValueMapLabelRenderer 511
ValueMapRenderer 522
ValueRange 531

Introducing the Java Viewer

1

IN THIS CHAPTER

- **What is the Java Viewer?**
- **Java Viewer file organization**
- **Java Viewer frames**

Notes

You can use ArcIMS Designer to create a viewer from the Java Standard or Java Custom template. The Java Standard template is an out-of-the-box solution that cannot be customized and is therefore not discussed in this book. Described here is the Java Custom template, hereafter referred to as the Java Viewer.

ESRI®ArcIMS™ 3 software provides a suite of tools allowing you to create very effective web sites for your mapping and GIS needs. The ArcIMS Viewers provide the foundation for the graphical and functional components of these web sites. You can build on this foundation through customization of the ArcIMS Viewers.

Customizing ArcIMS is a series of books that describes the customization of the ArcIMS Viewers and their programming references. This series covers customization using the HTML and Java™ Viewers and the ActiveX® and ColdFusion® Connectors.

This book explains the foundation for customizing the ArcIMS Java client, or viewer, as it is commonly referred to, as well as provides a complete reference to the Java Viewer Object Model available with ArcIMS.

This book assumes that you have a working knowledge of HTML and familiarity with JavaScript™.

In this chapter, you are introduced to

- Reasons for customizing the Java Viewer
- The file structure and frame layout of the Java Viewer
- The relationship between key HTML and JavaScript files with the Java Viewer

What is the Java Viewer?

If you have used ArcIMS Designer to create a web site, you are probably already familiar with the Java Viewer. The Java Viewer defines the graphical look and functionality of your ArcIMS web site. The default Java Viewer is a set of HTML pages and JavaScript files that reflect the choices you make on the panels of ArcIMS Designer. The HTML files are used to generate each web page component and to interact with the applets. Many of the HTML files have embedded JavaScript code that contains parameters for customization.

The Java Viewer provides a framework for the map, toolbar, legend, overview map, and other graphic portions of the page. Starting with this initial framework, you can quickly customize the web page.

Even with the many choices you have in ArcIMS Designer, you still need to be more flexible in order to implement a more customized look in the design of your web site.

You may want to customize in the following ways to meet your users' needs:

- Changing the frame layout
- Modifying the toolbar
- Adding functionality
- Changing the graphic look
- Inserting your own logo and changing colors

Considerations for choosing the Java Viewer

The Java Viewer can display your map data as images (Image MapService) or streamed features (Feature MapService). The Java Viewer can display one or more Image or Feature MapServices in any combination. In addition, data from local sources (shapefiles or ArcSDE layers) can be added in the same viewer. When using streamed features or local vector data, you can:

- Change the color or style of a layer from the interface. Changing the color or style takes place inside the applet. No additional requests to the ArcIMS Spatial Server are required.
- Add MapTips (small text popups with attribute information).
- Create EditNotes (simple attribute and spatial edits that can be submitted to the ArcIMS Spatial Server and converted to XML or shapefile format).
- Add MapNotes (notes such as text or graphic elements that you add to the map).

The Java Viewer differs from the other viewers in that it uses Java applets to display the map, legend, and scale bar and to send requests to the ArcIMS Spatial Server. To customize, you can communicate with these applets using JavaScript to access methods in the Java Viewer Object Model. The look and feel of the Java Viewer can also be customized using HTML and JavaScript to alter tags and parameters.

The Java Viewer is considered most advantageous for Intranet use. When the Java Viewer is opened, Java applets are sent to the client's web browser. End users interact with the server through these applets. The Java Viewer first processes requests on the server-side. If you are using a Feature MapService, the features are then cached on the client's machine. The caching of features speeds up data retrieval by minimizing requests to the server. But the client machine must be able to handle the processing done by the applets. The difficulties with the heavy nature of the applets are lessened in an Intranet environment.

The Java Viewer is best suited for the Intranet because it requires two downloads. The Java Runtime Environment (JRE) and ArcIMS Java Viewer components are required to run the Java Viewer. Both are onetime downloads, but each is over 5 MB in size. The download is most likely not suitable for the Internet unless you know your user is on a T1 line.

Another consideration for the Java Viewer is the web browser being used. The Java Viewer must be run in a web browser that supports scripting to Java 2 applets. At press time, Internet Explorer™ allows this but Netscape® browsers do not.

Java Viewer file organization

Directory structure

When you create a web site using the Java template through ArcIMS Designer, a hierarchy of directories and files is generated. The web site directory contains a set of HTML files, JavaScript parameter files, and a viewer configuration file (default.axl), along with two subdirectories—images and Meta-inf. The viewer configuration file is introduced in Chapter 2, ‘Customizing the Java Viewer’.

The images subdirectory stores any GIF and JPEG files used for tools icons, cursors, and backgrounds. The Meta-inf subdirectory is created when building a Java Viewer from ArcIMS Designer, but is not needed for any customization of the viewer. It can be deleted to make your Java Viewer lighter.

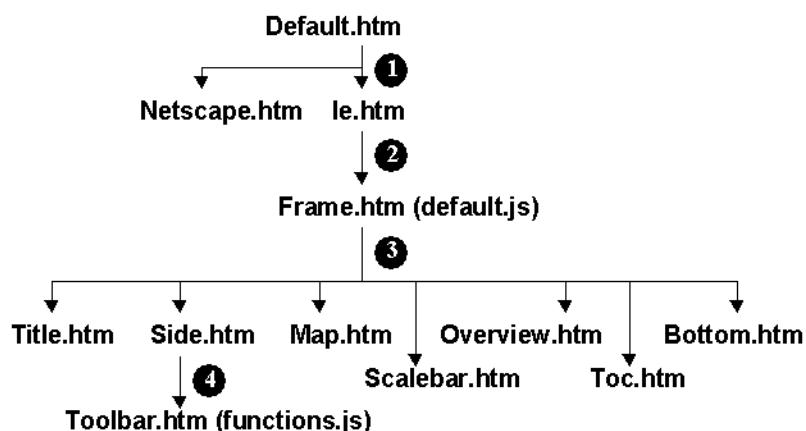
The default.js and functions.js parameters files

The default.js file is a JavaScript file used to define the functionality you created during the ArcIMS Designer process. Some of the functionality defined in this file includes MapTips, active layers, map extents, scale bar units, and query results. These functions are referenced in your web site through the frame.htm file. The functions included in the default.js file are described in Chapter 2, ‘Customizing the Java Viewer’.

The functions.js file is a JavaScript file used to define the functionality of each tool in your toolbar. The tools are used to communicate with the map using methods in the Java Viewer Object Model. These functions are incorporated into your web site through the toolbar.htm file. The ‘Customizing the Java Viewer’ chapter describes the functions included in the functions.js file.

The HTML files

There are approximately 27 HTML files that define the page content of the Java Viewer. When opening a Java Viewer created by ArcIMS Designer, files are opened and accessed in a sequence. The diagram below shows this sequence.



Default.htm is the entry point for your Web site. In step 1, the default.htm is set up to open either the ie.htm or the netscape.htm file. (Note: The file index.htm is also created through ArcIMS Designer and is used to redirect the browser to default.htm if your server is set to index.htm as its default.)

If a Netscape browser is used to open the web site, the netscape.htm file posts a warning that the browser cannot be used as it does not support scripting to Java 2 applets. The ie.htm file checks that the correct Java Runtime Environment is installed on the machine opening the Web site. If the JRE is not detected, the user is prompted to install it from another web page. (The JRE is a necessary component to allow feature streaming with Java clients.)

After the check for the JRE in step 2, the frame.htm file is opened to define the frames for the web site. The frame.htm is created to define the number, content, dimensions, and positions of frames for your web site. This file is customizable, so you can alter the frame layout by changing its parameters. Frame.htm loads default.js, which loads the viewer configuration file (default.axl). The viewer configuration file is described in detail in the next chapter.

In step 3, once the frames are defined, the HTML files for each added frame will be executed. The two essential frames, the map frame and the toolbar frame, are opened by the map.htm and toolbar.htm files.

The map.htm file calls for the map applet, IMSMap. The IMSMap applet is the key for the Java Viewer. Other parameters set in the map.htm file include the code location, applet name, and Java version. A check is made to verify that the ArcIMS Viewer components are on the client machine. If they are not present, the user will be redirected to do a onetime download of the components.

Two additional files are loaded in step 3: title.htm and side.htm. The side.htm file is a placeholder until the JavaScript function loadToolBar() has been loaded from default.js. Once the function is loaded, side.htm is replaced with toolbar.htm.

The map.htm, overview.htm, scalebar.htm, and toc.htm files send and start the Java applets to the client's web browser.

These four files reference the default.js file to get information on:

- Layers to add to the map and how each is rendered
- Layers to add to the overview map
- Units for the scale, screen, and map units
- Layer names and rendering to be listed in the legend

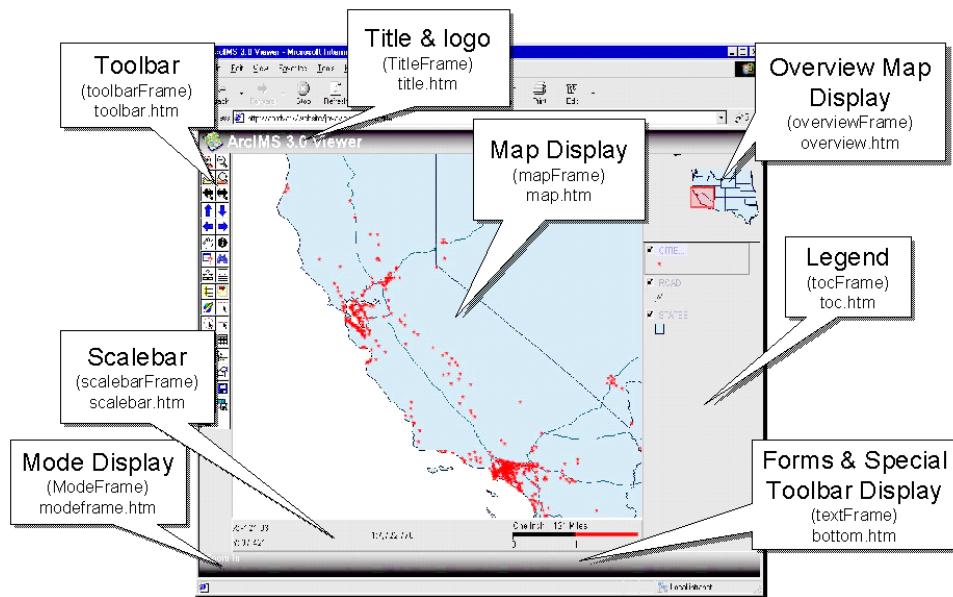
The toolbar.htm file has a table to define the number, order, and position of tools on the toolbar. Each tool has the location of the image, the name, and instructions for the onmousedown() function. Toolbar.htm references the functions.js file, which provides the JavaScript functions for supporting each tool.

The bottom.htm file fills the TextFrame at the bottom of the Java Viewer. Depending on what functionality you have added to your web site, additional HTML files may be used to fill this bottom frame. The TextFrame is described in the following section.

Java Viewer frames

When the Java Viewer is created by ArcIMS Designer, eight frames are created, each containing an important function for your Web site. You can move, resize, or delete any frame created.

This diagram shows the basic frame layout of the Java Viewer.



You can change the dimensions of any of these frames by editing the frame.htm file. Each frame has an associated HTML file that defines the content inside the frame or calls any necessary applets to create the frame.

TitleFrame

Title.htm defines the content of the TitleFrame. This frame includes a logo and title, set as “ArcIMS 3.0 Viewer” by default.

MapFrame

The map.htm file calls for the map applet IMSMap. The MapFrame is the web site’s map display and holds the map applet. It receives parameter information from the default.axl file.

OverviewFrame

The overview.htm file calls the applet to fill the OverviewFrame. To the right of the map and above the legend is the overview map display area.

ToolbarFrame

The ToolbarFrame contains toolbar.htm. The ToolbarFrame displays the tools for the web site. Chapter 2, 'Customizing the Java Viewer', describes how you can add a button to the toolbar.

TocFrame

Toc.htm starts the legend applet to define the content of the TocFrame. The TocFrame displays the legend for the map.

ScalebarFrame

Scalebar.htm starts the scale bar applet to define the content of the ScalebarFrame. The ScalebarFrame displays the scale bar for the map display.

ModeFrame

Modeframe.htm defines the content of the ModeFrame. The ModeFrame displays help descriptions for tools and is updated each time a tool is selected.

TextFrame

The main HTML frame, TextFrame, is bottom.htm. Other HTML files can open to fill the TextFrame, depending on what functionality you have added to the web site. If your web site offers tools to close projects, add MapNotes, use EditNotes, save map images, or set measure units, you can use the following files to fill the TextFrame:

- closeProject.htm
- editnotes.htm
- submitEditNotes.htm
- loadingMapNotes.htm
- mapnotes.htm
- mapnoteslayerlist.htm
- submitmapnotes.htm
- saveMapImage.htm
- setmeasureunits.htm

Customizing the Java Viewer

2

IN THIS CHAPTER

- **Changing layers of a MapService**
- **Opening files: changing the order and removing files**
- **Changing title, logo, and colors**
- **Loading the MapService, Overview Map, and Toolbar**
- **Setting active and visible layers**
- **Setting the right-click menu**
- **Setting MapTips**
- **Setting visibility and alias for results**
- **Setting the scale bar units**
- **Setting Folders for MapNotes and EditNotes**
- **Setting colors for the applets**
- **Setting stored queries**
- **Adding a tool to the toolbar**
- **Using hyperlinks**
- **Using the sample Java Viewers**

The Java Viewer is a template for displaying map data as images or streamed features. The Java Viewer uses JavaScript to access the Viewer Object Model. The viewer can be customized using JavaScript and HTML programming to create a Java web site application that meets your needs.

To appreciate the power and simplicity of the Java Viewer, you must understand the relationship between key HTML and JavaScript files in the viewer.

In this chapter, you learn how to

- Use the viewer configuration file (default.axl) created by ArcIMS Designer to change the Java Viewer's layers.
- Change the color, title, logos, and layout of the Java Viewer.
- Add, remove, or change a button on the toolbar.
- Edit some basic configurations of the Java Viewer's tools.
- Use the sample Java Viewers included with ArcIMS.

Changing layers of a MapService

Using ArcIMS Author, you create a map configuration file (*.axl). This file is used as the input to a MapService.

When you create a Java Viewer through ArcIMS Designer, a viewer configuration file, named default.axl, is created and stored inside the web site's directory. The viewer configuration file is a reference file used to define the map display; it contains references to what layers will be included in the web site and how each layer will be rendered. When a web site is first loaded, the viewer configuration file is sent to one or more ArcIMS Spatial Servers to retrieve information from one or more MapServices.

By default, the information from each MapService listed in the file will be retrieved as is. Rendering and query information can be added to the viewer configuration file to override information in a MapService.

The contents of the viewer configuration file may or may not look similar to the map configuration file depending on the MapService.

Refreshing an Image MapService

For an Image MapService, the map configuration file will contain a reference to one image layer only with a reference to the MapService name. To change an Image MapService, first edit the corresponding map configuration file. Then use ArcIMS Administrator to refresh the MapService. You will see the changes to the MapService when you reopen the web site.

Refreshing a Feature MapService

When a Feature MapService is used, each layer in the map configuration has a corresponding layer in the viewer configuration file. For example, if you have five polygon layers in your map configuration file, you will also have the same five polygon layers defined in the viewer configuration file as feature layers.

If you make a change to the map configuration file, you may need to update the viewer configuration file.

If you change the rendering of a layer in the map configuration file, no changes are needed in the viewer configuration file.

If you delete a layer from the map configuration file, you must delete the layer reference in the viewer configuration file.

If you add a layer to the map configuration file, add the layer reference to the viewer configuration file. Be sure to add a new workspace directory if necessary.

Other ways to change layers

Other ways to access and use the MapService include using the Viewer Object Model to manipulate the map. This chapter contains some examples on using the Viewer Object Model. Also, if a web site is designed to include the Layer Properties tool, a user can manipulate the rendering through the user interface.

Opening files: changing the order and removing files

The sequence in which files are opened is established in ArcIMS Designer. You can alter this sequence to meet your application needs.

Edits to ie.htm

The default.htm file opens either ie.htm or netscape.htm (netscape.htm posts a message that the browser cannot be used as it does not support Java 2 applets).

The purpose of ie.htm is to check for the correct JRE install and open frame.htm.

You can add a parameter to change the location of the ArcIMS plug-in install.

```
<param name="ALTERNATE_URL" value="url">
```

You will not see a parameter in ie.htm that calls frame.htm; it is set up behind the scenes. You can change which file ie.htm opens next by adding a parameter. In this example, the parameter is set to open a file called mypage.htm.

```
<param name="MAIN_PAGE" value="mypage.htm">
```

Removing files and JavaScript functions

Some of the files generated by ArcIMS Designer are optional depending on the needs of your web site. Removing unnecessary files will make your web site easier to manage. You can also remove JavaScript functions that are not being used.

If you remove frames, you should be aware of the following:

- Include onload="javascript:init();" in the main frameset.
- If you remove modeFrame, you will need to remove references to parent.modeFrame.document in default.js and functions.js.
- If you remove textFrame, you will need to remove references to parent.textFrame.document in functions.js.

Changing title, logo, and background

You can change the title logo, and background used in the frames of the Java Viewer.

Changing the title

The title of the Java Viewer can be set when creating the web site with ArcIMS Designer; by default, it is “ArcIMS 3.0 Viewer”.

You can change the title after the Java Viewer is created by editing the frame.htm file. The following variable should be changed:

```
var theTitle = "My very own viewer";
```

Changing the logo and background on the titleFrame

The logo, which appears in the top left corner of the Java Viewer, can be changed by editing the title.htm file. By default, the logo used is *aimslogo1x2.gif* from the images directory in your web site. Edit the location and/or name of the image to change the logo that appears in the titleFrame.

The background for the titleFrame uses *grad_gray.jpg* in the images directory. You can edit the location and/or name of the image to change the background appearance of the titleFrame.

You can also change the colors of the map, overview map, legend, and scale bar with the `setAppletColor()` function in `default.js`. See the “Setting colors for the applets” section later in this chapter for details.

Loading the MapService, Overview Map, and Toolbar

The default.js file is a JavaScript file that defines the functionality of the viewer created with Designer.

Loading the MapService

The MapService is added to the Java Viewer using the loadMapService() function in default.js. The viewer configuration file (default.axl) is loaded using loadMapOnlyProject. Until this function is invoked, the IMSMap applet is empty.

```
function loadMapService () {  
    //find IMSMap Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    var codebase = document.URL;  
    var idx = codebase.lastIndexOf('/');  
    if (idx >= 0)  
        codebase = codebase.substring(0, idx+1);  
    applet.loadMapOnlyProject (codebase + 'default.axl');  
}
```

Loading the Overview Map

Layers are added to the overview map with the setOverviewLayers() function in the default.js. Only layers that are in IMSMap can be in the overview map. For Image MapServices, the entire MapService must appear in the overview map. For Feature MapServices, you can choose which layers to add by specifying the layer name using GetLayer() and adding the layer using addMapLayer(). If two MapServices have the same layer name, ArcIMS will use the first layer it encounters with that name.

```
function setOverviewLayers() {  
    var aeLayer;  
    var subLayer;  
    var overviewApplet = parent.overviewFrame.document.IMSOerviewMap;  
    if (overviewApplet == null)  
        return;  
    var mapApplet = parent.mapFrame.document.IMSMap;  
    if (mapApplet == null)  
        return;  
    setTimeout ('parent.overviewFrame.document.IMSOerviewMap', 1200);  
    aeLayer = mapApplet.getLayer('cities');  
    overviewApplet.addMapLayer (aeLayer);  
    overviewApplet.redraw ();  
}
```

Loading the Toolbar and ModeFrame

The toolbar is added using the loadToolBar() function in default.js. The toolbar frame is the last frame filled. The file side.htm is used as a placeholder until the toolbar is generated. The creation of the toolbar is automatic based on the tools selected in ArcIMS Designer. At the same time, modeframe.htm is added to the ModeFrame. The ModeFrame lets the user know which tool is currently selected. The default is the Zoom In tool, which is set using selectTool (0).

```
function loadToolBar() {  
    parent.toolbarFrame.document.location='toolbar.htm';  
    parent.modeFrame.document.location='modeframe.htm';  
    parent.mapFrame.IMSMap.selectTool (0);  
}
```

Setting active and visible layers

Setting visible layers

Whether a layer is turned on or off when the web site is opened is set by the setVisibleLayers() function in default.js. A value of 1 means the layer will be on, and 0 means it will be turned off. In the following example, only the country layer will be visible when the web site opens. In this function, each layer is first added and then the visibility is set. Edit the visibility by changing the value of setVisibleByInt to 0 or 1.

```
function setVisibleLayers () {  
    var aeLayer;  
    var subLayer;  
    //find IMSMap Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    aeLayer = applet.getLayer ('country');  
    aeLayer.setVisibleByInt (1);  
    aeLayer = applet.getLayer ('LAKES');  
    aeLayer.setVisibleByInt (0);  
    aeLayer = applet.getLayer ('CITIES');  
    aeLayer.setVisibleByInt (0);  
    applet.redraw();  
}
```

Setting active layers

Which layer is selectable (active) by default is set by the setActiveLayer() function in default.js. Setting a layer active is important if you do not include a legend on your web site, as the user will have no way to activate a layer. In this example the country is set as the active layer using set.SelectedLayer(). (Only one layer can be active at a time.)

```
function setActiveLayer () {  
    var aeLayer;  
    //find IMSMap Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    aeLayer = applet.getLayer ('country');  
    aeLayer.setFeaturesSelectable (1);  
    applet.setSelectedLayer ('country');  
}
```

Setting the right-click menu

Right-clicking on a layer in the legend opens a menu of options. When the Java Viewer is created by ArcIMS Designer, the following options are added to the right-click menu: Zoom to Active Layer and Use in Overview Map (if an overview map was added to the web site).

You can use the setupTOCPopupMenuItems() function in the default.js file to set which menu choices will be available when a layer is right-clicked. To change the right-click menu, remove the comments on the function, then remove the comments for the commands you want to add to the right-click menu. The following choices can be added to the right-click menu: Remove Layer, Display Layer Classification, Move Layer, Scale Factors, Clear Selection, Clear Labels, Clear MapTips, and Layer Properties.

Below is a portion of the TOCPopupMenuItems() function with the Remove Layer option enabled.

```
function setupTOCPopupMenuItems() {  
    var applet = parent.tocFrame.document.IMSToc;  
    if (applet == null)  
        return;  
  
    //uncomment the lines you would like to use to  
    //define the IMSToc popup menu items  
  
    //Remove Layer menu item  
    applet.enableRemoveLayer()  
    // applet.disableRemoveLayer()  
    applet.isRemoveLayer()
```

Setting MapTips

MapTips are small popup windows that display the value of a field as you drag your mouse over a map feature. The setMapTipFields() function in the default.js file sets the fields for MapTips. In this function, each layer is listed and setMapTipField is used to set the value of the MapTip.

Each layer in a Feature MapService can have MapTips. MapTips can only be set for one field per layer.

In the example below, the layer ‘country’ has its MapTips field set to ‘FIPS_CNTRY’.

```
function setMapTipFields () {  
    var aeLayer;  
    //find IMSMap Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    aeLayer = applet.getLayer ('country');  
    aeLayer.setMapTipField ('FIPS_CNTRY');  
    aeLayer = applet.getLayer ('LAKES');  
    aeLayer.setMapTipField ('AREA');  
    aeLayer = applet.getLayer ('CITIES');  
    aeLayer.setMapTipField ('NAME');  
}
```

Setting visibility and alias for results

You can control a field's visibility and alias in the Identify Results dialog box with the setQueryResultFields() function in the default.js file. Turn a field off by removing the second field name in the pair. Set an alias by replacing the second value with a new name. In the following example, FIPS_COUNTRY field will not be shown and the field name CNTRY_NAME will appear instead as the alias CountryName.

```
function setQueryResultFields () {  
    var aeLayer;  
    var nameValuePairCollection;  
    //find IMSMap Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    //layer  
    nameValuePairCollection = applet.createCollection();  
    aeLayer = applet.getLayer ('country');  
    //nameValuePairCollection.addNameValuePairElement  
    //    (applet.createNameValuePair('FIPS_CNTRY', ''));  
    nameValuePairCollection.addNameValuePairElement  
    (applet.createNameValuePair('CNTRY_NAME', 'CountryName'));  
}  
}
```

Setting the scale bar units

The setScalebarUnits() function in the default.js file sets the screen units (centimeters or inches) and scale units (miles, meters, feet, or kilometers) for the scale bar and the map units or data source units (decimal degrees, meters, or feet).

```
function setScalebarUnits () {  
    //find scalebar Applet  
    var scalebarApplet = parent.scalebarFrame.document.IMSScaleBar;  
    if (scalebarApplet == null)  
        return;  
    //find map Applet  
    var mapApplet = parent.mapFrame.document.IMSMap;  
    if (mapApplet == null)  
        return;  
    mapApplet.setMapUnit ('Decimal Degrees');  
    scalebarApplet.setScaleUnit ('Miles');  
    scalebarApplet.setScreenUnit ('Inch');  
    scalebarApplet.refresh();  
}
```

Setting Folders for MapNotes and EditNotes

Setting Folders for MapNotes

When a user creates MapNotes or EditNotes, the results are submitted to an ArcIMS Spatial Server and stored in a folder that you can administer with ArcIMS Administrator.

The `setMapNotesFolder()` function in the `default.js` file sets the folder where the notes are stored. You can also use the `setMapNotesFolder()` function to set a limit on the amount of data that can be submitted for each MapNotes session. The limit is in KB and is set to 100 KB by default in ArcIMS Designer (see highlighted code below).

```
function setMapNotesFolder () {  
    //find map Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    applet.setMapNotesFolderOnServer ('myspatialserver', 'MapNotes');  
    applet.setMapNotesSubmitLimit (100);  
}
```

Setting Folders for EditNotes

Similar to MapNotes, the `setEditNotesFolder()` function in the `default.js` file sets the folder where the notes are stored. However, no limit can be set on the amount of data that can be submitted for each EditNotes session.

```
function setEditNotesFolder () {  
    //find map Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    applet.setEditNotesFolder ('myspatialserver', 'EditNotes');  
}
```

Setting colors for the applets

You can use the setAppletColor() function to set the colors for the applets. You can set the background color of the map; the background, rectangular fill, and outline color on the overview map; and the background and text color of the legend and scale bar. Color values are based on RGB values of 0–255 values.

```
function setAppletColor() {  
    //map  
    var mapApplet = parent.mapFrame.document.IMSMap;  
    if (mapApplet == null)  
        return;  
    mapApplet.setBGColor (255,255,255);  
    //overview  
    var overviewApplet = parent.overviewFrame.document.IMSOverviewMap;  
    if (overviewApplet == null)  
        return;  
    overviewApplet.setBGColor (192,192,192);  
    overviewApplet.setRectangleFillColor (255,0,0,80);  
    overviewApplet.setRectangleoutlinecolor (255,0,0,250);  
    //toc  
    var tocApplet = parent.tocFrame.document.IMSToc;  
    if (tocApplet == null)  
        return;  
    tocApplet.setBGColor (192,192,192);  
    tocApplet.setFGColor (0,0,0);  
    tocApplet.refresh();  
    //scalebar  
    var scalebarApplet = parent.scalebarFrame.document.IMSScaleBar;  
    if (scalebarApplet == null)  
        return;  
    scalebarApplet.setBGColor (192,192,192);  
    scalebarApplet.setFGColor (0,0,0);  
    scalebarApplet.refresh();  
}
```

Setting stored queries

If the map file has stored queries, the setStoredQueries() function is included in the default.js file. Stored queries are created during the authoring process. The user can take advantage of stored queries with the Search tool.

In this function, first getLayer is used to identify the layer that has a stored query. Then a collection is made of the field and expression used for the stored query. As the final step, the stored query is set using the setStoredQueries() collection.

```
function setStoredQueries () {  
    var aeLayer;  
    var storedQueryCollection;  
    //find IMSMap Applet  
    var applet = parent.mapFrame.document.IMSMap;  
    if (applet == null)  
        return;  
    //layer  
    aeLayer = applet.getLayer ('country');  
    storedQueryCollection = applet.createCollection();  
    storedQueryCollection.addStringElement('Cntry Name');  
    storedQueryCollection.addStringElement('NamewithLIKE');  
    aeLayer.setStoredQueries (storedQueryCollection);  
}
```

Enabling functions

The enableFunctions() function enables or disables a function specified by its ID. The buttons and menu items on the toolbar, menus, and popup menus are dynamically updated when changes are made to this function. Keep in mind that simply enabling a function will not necessarily make a respective button appear on the interface. See the next chapter for details on this function and the list of functions and their IDs.

Adding a tool to the toolbar

You can add new tools to the Java Viewer's toolbar, but you are limited to using the methods available in the Viewer Object Model. The next chapter details the object model.

To add a button to the Java Viewer, you need to edit the functions.js file to add the button's functionality, the default.htm file to declare the button's use variable, and the toolbar.htm file to add the button to the toolbar.

The following example shows how to add a Remove Layer button to the Java Viewer.

Adding the button's functionality: functions.js

The first step in adding a new button to the toolbar is to use methods in the Viewer Object Model to add functionality in the functions.js file.

In the JavaScript file, functions.js is a function called *clickFunction*. This function is used to define the behavior of all button clicks made to the interface. Each action is defined as a case statement within this function.

The method selectTool selects which tool to use. SetModeDisplay sets the text in the mode frame. For example, the behavior of the Identify tool is defined as:

```
case "identify":  
    if (checkSelectedLayer()) {  
        parent.mapFrame.IMSMap.selectTool ("IDENTIFY_TOOL");  
        setModeDisplay("Identify", "bottom.htm");  
    } else {  
        if (parent.toolbarFrame!=null)  
            parent.toolbarFrame.document.location="toolbar.htm";  
    }  
    break
```

Here is an example of how to implement the remove layer functionality. Add this case for "removelayer" to the functions.js file under the case for "addlayer".

```
case "removelayer":  
    var layer = parent.mapFrame.IMSMap.getSelectedLayer();  
    if(layer == null){  
        alert("No layer selected.");  
    }else if(!(layer.isImageServerSubLayer() || layer.isMOIMSSubLayer()  
    || layer.isAVIMSSubLayer())){  
        parent.mapFrame.IMSMap.removeLayer(layer);  
        parent.mapFrame.IMSMap.redraw();  
    }else{  
        alert("Cannot remove sublayers.");  
    }  
    break
```

The button will only remove layers that are not sublayers of an Image MapService, ArcView IMS service, or MapObjects IMS service. A reference to the selected layer is obtained and tested to ensure that a layer has been selected using GetSelectedLayer(). This layer is also tested to determine if it is a sublayer. If the layer is not a sublayer, it is then removed from the legend and map using removeLayer() and the map is redrawn using redraw(). If it is a sublayer, then a message, “Cannot remove sublayers”, is displayed.

Enabling the button: default.js

The Remove Layer button must now be added to the toolbar. To be consistent with how the other buttons are implemented, a variable, “useRemoveLayer”, should be added to the variable list at the bottom of default.js; its value should be set to *true*.

```
var useRemoveLayer=true;
```

Defining this variable offers a simple solution for enabling (true) and disabling (false) this tool.

Adding the button: toolbar.htm

The button is then generated dynamically within a script in the body of toolbar.htm. To have a logical grouping of buttons, the Remove Layer button should be added after the Add Layer button. You can do this by adding the code for the Remove Layer button after the code for the Add Layer button in toolbar.htm.

The image used for the button’s icon, delete.gif, is included in the images directory and is created when any Java Viewer is created by ArcIMS Designer. The ToolTip for the button is set by the alt parameter. The window.status parameter sets the text to appear in the browser’s status bar when you drag the mouse over the button. In this example, the ToolTip and the status bar text are “Remove Current Layer”.

In toolbar.htm, add the following lines for the Remove Layer button after the lines for the Add Layer button. For onmousedown, clickFunction is set to ‘removelayer’; this activates the ‘removelayer’ case in functions.js.

```
if (parent.useRemoveLayer) {  
    document.write('<td align="center" valign="middle">');  
    document.write('');  
    isSecond = !isSecond;  
    document.writeln('</td>');  
    if (isSecond) document.write('</tr><tr>');  
}
```

Reloading the web page into your browser will give you a new button for removing the active layer.

Removing a button from the toolbar

The variable list at the bottom of the default.js file defines which tools will be included in the toolbar. For example, var usePan=true; means that the Pan tool will be included.

You can set a button’s use variable to false to disable creation of the button and remove it from the toolbar.

Using hyperlinks

You can customize to create hyperlink-like functionality in the Java Viewer. It is referred to as ‘hyperlink-like’ functionality because instead of a single click, it is a three-click process. First, the selection tools are used to select features. Second, a list of the selected features and corresponding attributes, including URLs, is generated when the Attributes button is clicked. (The Attributes button displays the fields and values for selected features.) Third, a click of the hyperlink in the Attributes dialog box will open the corresponding URL.

To add hyperlink-like functionality to the Java Viewer, your dataset must include a field with hyperlinks. (The complete URL, including “[http://](#)”, must be in the hyperlink field.)

You need to make changes to two files to use hyperlinks: toolbar.htm and frame.htm.

Setting up the hyperlinks: editing frame.htm

Open the web site’s frame.htm file to make additions to the SCRIPT tag. First, set the parameters for setting up hyperlinks in the data display. Then define the layer name and field name that has the hyperlinks. List the layer name exactly as it appears in the map configuration file and always capitalize the field name.

In this example, the layer ‘museums’ has a field ‘WEBSITE’ that contains hyperlinks.

```
<html>
<head>
<SCRIPT type="text/javascript" language="JavaScript">
    // Designer will set the next variable - theTitle
    var theTitle = "San Francisco Museums";
    if (theTitle.indexOf("###TITLE##")!=-1) theTitle = "ArcIMS 3.0 Java Viewer";
    document.writeln("<TITLE>" + theTitle + "</TITLE>");
        var hyperLinkLayers = new Array();
        var hyperLinkFields = new Array();
        hyperLinkLayers[0] = "museums";
        hyperLinkFields[0] = "WEBSITE"
</script>
<script language="javascript" src="default.js">
</script>
</head>
```

Setting up the Attributes dialog: editing toolbar.htm

The toolbar.htm file has two options for displaying the dialog box of the Attributes button. The current option calls the showSelect script and uses a Java table to display attributes. The commented option calls the displayAttributes script and uses an HTML table. You will need to display the attributes in an HTML table. (The hyperlinks will be shown in the displayed attributes.)

Open the web site's toolbar.htm file to make changes to the useShowSelect function. In the example below, the parameters for showing the attributes in an HTML table have been uncommented and the Java table option has been commented.

```
if (parent.useShowSelect) {  
    document.write('<td align="center" valign="middle">');  
    // choose between the following two calls  
    // the next line will display attributes of selected features in HTML table.  
    // . . slower but can be customized easily  
    //     document.write('');  
    // the next line will display attributes of selected features in Java table.  
    // . . faster but cannot be customized  
    //     document.write('');  
    //  
    isSecond = !isSecond;  
    document.writeln('</td>');  
    if (isSecond) document.write('</tr><tr>');  
}  
}
```

Testing the hyperlink

To test the hyperlink, select some features, then click the Attributes button. The generated list will contain the hyperlink.

Using the sample Java Viewers

Two sample implementations of the Java Viewer have been provided with ArcIMS. They demonstrate a variety of functions and interface designs. A readme file with a general set of instructions, along with a brief description and requirements for running each sample, is provided.

If you chose all the default locations when you installed ArcIMS, the readme file can be found at <ArcIMS installation directory>\Samples\Viewers. For UNIX, the directory structure is the same, except <ArcIMS installation directory> is changed to \$AIMSHOME.

Generic Map

This sample viewer opens with a blank map and allows you to add a MapService or local data to the map and MapService layers to the overview map. This provides a way to test the Java Viewer without creating a MapService and a web site. It also has a tool for interactively adding the active layer to the overview map.

To use this sample:

1. Create any map configuration file (*.axl).
2. In the browser, type in the URL to your host web site Java Viewer directory (for example, http:\<ArcIMS host>\website\javaviewer).
3. Click Generic Map, then click the Open Project tool. Choose a map configuration file, then click Open AXL file.

Black Tie

This sample viewer presents a different look for the Java Viewer. The interface is black with tools along the bottom of the screen.

To use this sample:

1. Create a Feature MapService named SanFranFeature (case-sensitive) from sf.axl.
2. Go to /javaviewer/Black and open the default.axl file in a text editor. Look for the <FEATUREWORKSPACE> tag and change the ArcIMS host name in the url attribute.
3. In the browser, type in the URL to your host web site Java Viewer directory (for example, http:\<ArcIMS host>\website\javaviewer).
4. Click Black Tie. All functions work the same as the default Java Custom Viewer.

In \javaviewer\Black, the frame.htm file defines each of the files that creates the entire web page, for example, map.htm, bottom.htm, and overview.htm. Look at each of these HTML files and notice that each one has the background color set to black.

Java Viewer Object Model

3

IN THIS CHAPTER

This chapter references the 31 Java Viewer Object Model classes, listing them and their associated public methods in alphabetical order.

The Java Viewer uses Java applets to display the map, legend, and scalebar, and to send requests to the ArcIMS Spatial Server. You can communicate with these applets using JavaScript to access methods in the Viewer Object Model.

The Viewer Object Model allows developers to customize clients for browsers. This includes both functionality and appearance of the client.

A series of objects are available for use with the JavaScript functions. This chapter describes those objects, their methods, and gives examples.

IMSMMap is the interface through which all customizing is done. Any functionality must be based off obtaining a reference to or creating objects via method calls from IMSMMap.

CallOutMarkerSymbol extends Symbol

A CallOutMarkerSymbol draws a label and a callout box around it. This symbol works with point features only. Drawing it allows special effects like glowing, shadows, outline and antialiasing. You can create an instance of the class by calling `IMSSymbol.createSymbol` method.

See Also

[IMSSymbol](#)
[ArcXML Programmer's Reference](#)
[Symbol](#)

CallOutMarkerSymbol.getAntialiasing

Description

Retrieves the current antialiasing value of the symbol label. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a CallOutMarkerSymbol by default.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean true if antialiasing is to be used, false otherwise

See Also

[CallOutMarkerSymbol.setAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default value of antialiasing for CallOutMarkerSymbol is " + symbol.getAntialiasing());
```

CallOutMarkerSymbol.getBackColor

Description

Retrieves the background color of the box around the CallOutMarkerSymbol label. Default value of the color is white.

Syntax

```
String getBackColor()
```

Argument

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[CallOutMarkerSymbol.setBackColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default value of the background color for CallOutMarkerSymbol is " +
symbol.getBackColor());
```

CallOutMarkerSymbol.getBoundaryColor

Description

Retrieves the color of the box boundary around the CallOutMarkerSymbol label. Default value of the color is black.

Syntax

```
String getBoundaryColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[CallOutMarkerSymbol.setBoundaryColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default value of the background color for CallOutMarkerSymbol is " +
symbol.getBoundaryColor());
```

CallOutMarkerSymbol.getFontColor

Description

The getFontColor returns the current color used to draw the font of the label. The default value is black.

Syntax

```
String getFontColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[CallOutMarkerSymbol.setFontColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default value of the font color for CallOutMarkerSymbol is " + symbol.getFontColor());
```

CallOutMarkerSymbol.getFontName

Description

Returns the current name of the font used by the CallOutMarker symbol object.

Syntax

String getFontName()

Arguments

None

Returned Value

String the name of the font used

See Also

[CallOutMarkerSymbol.setFontName](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default name of the font name for CallOutMarkerSymbol is "+ symbol.getFontName());
```

CallOutMarkerSymbol.getFontSize

Description

Retrieves the current size of the font used by the CallOutMarkerSymbol.

Syntax

```
int getFontSize()
```

Arguments

None

Returned Value

The size (in points) of the font specified

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("the size of the font used by CallOutMarkerSymbol is " + symbol.getFontSize());
```

CallOutMarkerSymbol.getGlowColor

Description

Returns the current color of the glow effect around the CallOutMarker symbol label. The color value must be set up by the setGlowColor method first.

Syntax

```
String getGlowColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[CallOutMarkerSymbol.setGlowColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color = mapFrame.IMSMap.createColor(0,255,0);
alert("set color of the glowing effect to green" + symbol.setGlowColor(color2));
alert("current color of the glowing effect is " + symbol.getGlowColor());
```

CallOutMarkerSymbol.getInterval

Description

Returns distance between the callout box and the point feature to be labeled. The default interval value is 10.

Syntax

```
int getInterval()
```

Arguments

None

Returned Value

The distance in screen pixels

See Also

[CallOutMarkerSymbol.setInterval](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default value of the interval property is " + symbol.getInterval());
```

CallOutMarkerSymbol.getOutlineColor

Description

Returns the color of the Outline effect around the CallOutMarker symbol label. The color value must be set up by the setOutlineColor method first.

Syntax

```
String getOutlineColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[CallOutMarkerSymbol.setOutlineColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color = mapFrame.IMSMap.createColor(0,255,0);
alert("set color of the glowing effect to green" + symbol.setOutlineColor(color2));
alert("current color of the outline effect is " + symbol.getOutlineColor());
```

CallOutMarkerSymbol.getShadowColor

Description

Returns the color of the shadow effect around the CallOutMarker symbol label. The color value must be set up by the setShadowColor method first.

Syntax

```
String getShadowColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[CallOutMarkerSymbol.setShadowColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color = mapFrame.IMSMap.createColor(0,255,0);
alert("set color of the shadow effect to green" + symbol.setShadowColor(color2));
alert("current color of the shadow effect is " + symbol.getShadowColor());
```

CallOutMarkerSymbol.getTransparency

Description

The getTransparency method retrieves the current transparency value set on the object. The default value is 1.0 with a valid range of 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

Double value ranging from 0.0 to 1.0

See Also

[CallOutMarkerSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
alert("default value of the transparency property is " + symbol.getTransparency());
```

CallOutMarkerSymbol.setAntialiasing

Description

Sets the current antialiasing value of the label for the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a CallOutMarkerSymbol by default.

Syntax

```
boolean setAntialiasing( String enabled)
```

Arguments

enabled	“true” is to set antialiasing on, “false” is to set it off.
---------	--

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
symbol.setAntialiasing("true");
```

CallOutMarkerSymbol.setBackColor

Description

Sets a color of the callout box background for the CallOutMarker.

Syntax

```
boolean setBackColor( Color color)
```

Arguments

color an instance of the Color class symbol that defines the color to be set.

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getBackColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setBackColor(color1)
```

CallOutMarkerSymbol.setBoundaryColor

Description

Sets a color of the boundary of the callout box around the CallOutMarker symbol.

Syntax

```
boolean setBoundaryColor( Color color)
```

Arguments

color an instance of the Color class that defines the color to be set.\

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getBoundaryColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setBoundaryColor(color1)
```

CallOutMarkerSymbol.setFont

Description

Sets particular font and its size for the CallOutMarker symbol label

Syntax

```
boolean setFont( String fontName, int size)
```

Arguments

fontName	the name of the font to be set
size	size of the font in points

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getFontSize](#)
[CallOutMarkerSymbol.getFontName](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
symbol.setFont("Arial", 12)
```

CallOutMarkerSymbol.setFontColor

Description

Sets a color of the CallOutMarkerSymbol label font

Syntax

```
boolean setFontColor( Color color)
```

Arguments

color an instance of the Color class that defines the color to be set.

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getFontColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setFontColor(color1)
```

CallOutMarkerSymbol.setGlowColor

Description

Sets a color of the glowing effect around the CallOutMarkerSymbol label.

Syntax

```
boolean setGlowColor( Color color)
```

Arguments

color an instance of the Color class that defines the color to be set.

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getGlowColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setGlowColor(color1)
```

CallOutMarkerSymbol.setInterval

Description

Sets distance between the callout box and the point feature to be labelled.

Syntax

```
boolean setInterval ( double intervalValue)
```

Arguments

intervalValue defines distance value to be set. It is measured in screen points and must be nonnegative.

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getInterval](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
symbol.setInterval(2);
```

CallOutMarkerSymbol.setOutlineColor

Description

Sets a color of the outline effect around the CallOutMarkerSymbol label

Syntax

```
boolean setOutlineColor( Color color)
```

Arguments

color an instance of the Color class that defines the color to be set.

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getOutlineColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setOutlineColor(color1)
```

CallOutMarkerSymbol.setShadowColor

Description

Sets a color of the shadow effect around the CallOutMarkerSymbol label. The shadow will be drawn using a 0.5 transparency

Syntax

```
boolean setShadowColor( Color color)
```

Arguments

color an instance of the Color class that defines the color to be set.

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getShadowColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setShadowColor(color1)
```

CallOutMarkerSymbol.setTransparency

Description

The setTransparency method allows you to set the level of transparency on the label. Valid values are 0.0 (Transparent) to 1.0 (Opaque).

Syntax

```
boolean setTransparency( double transparencyValue)
```

Arguments

transparencyValue	defines transparency value to be set. It must be any value between 0.0 and 1.0.
-------------------	---

Returned Value

returns true if the method was successful, false otherwise

See Also

[CallOutMarkerSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("CALLOUT_MARKER_SYMBOL");
symbol.setTransparency(0.5);
```

Collection

Description

A Collection is a very simple stack implementation for multiple String and/or NameValuePair elements. It has utilities for putting and retrieving elements in the Collection, getting the size of the Collection, and testing for containment. Once elements are added to the Collection, they cannot be removed. It is very useful for passing data between Java and JavaScript.

See Also

[IMSMMap.createCollection](#)
[NameValuePair](#)

Collection.addNameValuePairElement

Description

Adds a NameValuePair element to the end (index is equal to Collection.getSize()) of the Collection.

Syntax

```
boolean addNameValuePairElement(NameValuePair pair)
```

Arguments

pair The NameValuePair to be added to the end of the Collection.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.getCollectionElementAt](#)
[NameValuePair](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var theCollection = imsMap.createCollection();
var newPair = imsMap.createNameValuePair("President", "John Smith");
var newPair2 = imsMap.createNameValuePair("VicePresident", "Mary Smith");
theCollection.addNameValuePairElement(newPair);
theCollection.addNameValuePairElement(newPair2);
var pairFromCollection = imsMap.getCollectionElementAt(theCollection,1);
alert(pairFromCollection.getValue());
```

Collection.addStringElement

Description

Adds a String element to the end (index is equal to Collection.getSize()) of the Collection.

Syntax

```
boolean addString(String string)
```

Arguments

String The String to be added to the end of the Collection.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var theCollection = imsMap.createCollection();
var theString = "Mary Smith";
theCollection.addStringElement("John Smith");
theCollection.addStringElement(theString);
var theString2 = theCollection.getStringElement(1);
alert(theString2);
...
```

Collection.getSize

Description

Returns the number objects in the collection.

Syntax

```
int getSize()
```

Arguments

None.

Returned Value

int The number of objects contained in the Collection.

Example

```
var imsMap= parent.mapFrame.document.IMSMap  
var theLayers = imsMap.getLayerNames();  
var theSize = theLayers.getSize();  
var theObject = theLayers.getStringElement(1);  
alert("The Map contains " + theSize + " layer(s).");  
alert("Layer 0 is " + theObject);
```

Collection.getStringElement

Description

Returns an element as a String.

Syntax

```
String getStringElement(int index)
```

Arguments

index The index of the desired.

Returned Value

String The String at the given index.

See Also

[Collection.addStringElement\(\)](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMap  
var theLayers = imsMap.getLayerNames();  
var theSize = theLayers.getSize();  
var theObject = theLayers.getStringElement(0);  
alert("The Map contains " + theSize + " layer(s).");  
alert("Layer 0 is " + theObject);
```

Color extends java.awt.Color

This class is a wrapper for the java.awt.Color. It should be used to set up color properties for instances of other classes, for symbols in particular. You can create an instance of the color class by calling IMSMap.createColor method.

See Also

[IMSMap](#)
[Symbol](#)

Example

```
var color = mapFrame.IMSMap.createColor(0,255,0);
```

Extent

An Extent is used to define a rectangle on a map, using map units. It includes methods to get the minimum and maximum x and y values. It is used in many methods such as `IMSMMap.setExtent()` and `IMSMMap.setExtentLimit()`. An Extent can be constructed through `IMSMMap.createExtent()`.

See Also

[IMSMMap](#)

Extent.getXMax

Description

Returns the maximum x value of the Extent.

Syntax

```
double getXMax()
```

Arguments

None

Returned Value

double The maximum x value of the Extent.

See Also

[Extent.getXMin](#)

[Extent.getYMax](#)

[Extent.getYMin](#)

Example

```
var MINWIDTH = 10;
var MINHEIGHT = 10;

var currentExtent = imsMap.getExtent();
var width = currentExtent.getXMax() - currentExtent.getXMin();
var height = currentExtent.getYMax() - currentExtent.getYMin();
if((width < MINWIDTH) || (height < MINHEIGHT)){
    var newExtent = imsMap.createExtent(currentExtent.getXMin(), currentExtent.getYMin(),
        MINWIDTH, MINHEIGHT);
    imsMap.setExtent(newExtent);
}
```

Extent.getXMin

Description

Returns the minimum x value of the Extent.

Syntax

```
double getXMin()
```

Arguments

None

Returned Value

double The minimum x value of the Extent.

See Also

[Extent.getXMax](#)

[Extent.getYMax](#)

[Extent.getYMin](#)

Example

```
var MINWIDTH = 10;  
var MINHEIGHT = 10;  
  
var currentExtent = imsMap.getExtent();  
var width = currentExtent.getXMax() - currentExtent.getXMin();  
var height = currentExtent.getYMax() - currentExtent.getYMin();  
if((width < MINWIDTH) || (height < MINHEIGHT)){  
    var newExtent = imsMap.createExtent(currentExtent.getXMin(), currentExtent.getYMin(),  
        MINWIDTH, MINHEIGHT);  
    imsMap.setExtent(newExtent);  
}
```

Extent.getYMax

Description

Returns the maximum y value of the Extent.

Syntax

```
double getYMax()
```

Arguments

None

Returned Value

double The maximum y value of the Extent.

See Also

[Extent.getXMax](#)
[Extent.getXMin](#)
[Extent.getYMin](#)

Example

```
var MINWIDTH = 10;  
var MINHEIGHT = 10;  
  
var currentExtent = imsMap.getExtent();  
var width = currentExtent.getXMax() - currentExtent.getXMin();  
var height = currentExtent.getYMax() - currentExtent.getYMin();  
if((width < MINWIDTH) || (height < MINHEIGHT)){  
    var newExtent = imsMap.createExtent(currentExtent.getXMin(), currentExtent.getYMin(),  
        MINWIDTH, MINHEIGHT);  
    imsMap.setExtent(newExtent);  
}
```

Extent.getYMin

Description

Returns the minimum y value of the Extent.

Syntax

```
double getYMin()
```

Arguments

None

Returned Value

double The minimum y value of the Extent.

See Also

[Extent.getXMax](#)

[Extent.getXMin](#)

[Extent.getYMax](#)

Example

```
var MINWIDTH = 10;  
var MINHEIGHT = 10;  
  
var currentExtent = imsMap.getExtent();  
var width = currentExtent.getXMax() - currentExtent.getXMin();  
var height = currentExtent.getYMax() - currentExtent.getYMin();  
if((width < MINWIDTH) || (height < MINHEIGHT)){  
    var newExtent = imsMap.createExtent(currentExtent.getXMin(), currentExtent.getYMin(),  
        MINWIDTH, MINHEIGHT);  
    imsMap.setExtent(newExtent);  
}
```

GradientFillSymbol extends Symbol

This symbol fills a polygon feature with a gradual variation of colors ranging from start color to finish color. Drawing it allows such special effect as antialiasing but boundaries will not be drawn. You can create an instance of the symbol by the IMSMap.createSymbol method.

See Also

[ArcXML Programmer's Reference](#)

[IMSMap](#)

[Symbol](#)

GradientFillSymbol.getAntialiasing

Description

Retrieves the current antialiasing value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a GradientFillSymbol by default.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean true if antialiasing is to be used, false otherwise

See Also

[GradientFillSymbol.setAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
alert("default value of antialiasing for GradientFillSymbol is " + symbol.getAntialiasing());
```

GradientFillSymbol.getEndColor

Description

Returns the end color in the gradial variation that uses to fill a polygon feature. The default value of the color is green.

Syntax

String getEndColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[GradientFillSymbol.getStartTime](#)
[GradientFillSymbol.setEndColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
alert("the default end color for the GradientFillSymbol is " + symbol.getEndColor());
```

GradientFillSymbol.getStartColor

Description

Returns the start color in the gradual variation that used to fill a polygon feature. The default value of the color is red.

Syntax

```
String getStartColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[GradientFillSymbol.setStartColor](#)
[GradientFillSymbol.getEndColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
alert("the default start color for the GradientFillSymbol is " + symbol.getStartColor());
```

GradientFillSymbol.getStyle

Description

Returns the current value of the style property of the gradient fill symbol. The property defines a direction of boundaries between different colors in the gradual variation that is used to fill a polygon feature. The default direction is backward diagonal.

Syntax

```
int getStyle()
```

Arguments

None

Returned Value

- 0 – forward diagonal
- 1 – backward diagonal
- 2 – horizontal
- 3 – vertical

See Also

[GradientFillSymbol.setStyle](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
alert("the default style for the GradientFillSymbol is " + symbol.getStyle());
```

GradientFillSymbol.getTransparency

Description

The getTransparency method retrieves the current transparency value set on the object. The default value is 1.0 with a valid range of 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

Returns the transparency value. It must be between 0.0 and 1.0

See Also

[GradientFillSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
alert("default value of the transparency property is " + symbol.getTransparency());
```

GradientFillSymbol.setAntialiasing

Description

Sets the current antialiasing value of the label for the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a CallOutMarkerSymbol by default.

Syntax

```
boolean setAntialiasing( String enabled)
```

Arguments

enabled “true” is to set antialiasing on,
“false” is to set it off.

Returned Value

true if the method was successful, false otherwise

See Also

[GradientFillSymbol.getAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
symbol.setAntialiasing("true")
```

GradientFillSymbol.setEndColor

Description

Sets the end color in the gradual variation that is used to fill a polygon feature. The default value of the color is green.

Syntax

```
boolean setEndColor(Color color)
```

Arguments

color – an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[GradientFillSymbol.getEndColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setEndColor(color1)
```

GradientFillSymbol.setStartColor

Description

Sets the start color in the gradual variation that uses to fill a polygon feature. The default value of the color is red.

Syntax

```
boolean setStartColor( Color color)
```

Arguments

color an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[GradientFillSymbol.getStartColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(0,0,255);
symbol.setStartColor(color1)
```

GradientFillSymbol.setStyle

Description

Sets the current value of the style property of the gradient fill symbol. The property defines a direction of boundaries between different colors in the gradual variation that is used to fill a polygon feature. The default direction is backward diagonal.

Syntax

```
boolean setStyle( int style)
```

Arguments

style: 0 – forward diagonal
1 – backward diagonal
2 – horizontal
3 – vertical

Returned Value

true if the method was successful, false otherwise

See Also

[GradientFillSymbol.getStyle](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
symbol.setStyle(3);
```

GradientFillSymbol.setTransparency

Description

The setTransparency method sets the current transparency value set on the object. The default value is 1.0 with a valid range of 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
boolean setTransparency( double transparencyValue)
```

Arguments

transparencyValue transparency to be set. It must have a value between 0 and 1.0

Returned Value

true if the method was successful, false otherwise

See Also

[GradientFillSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("GRADIENT_FILL_SYMBOL");
symbol.setTransparency(0.5);
```

GroupRenderer extends Renderer

A Group Renderer is used to draw a Layer using a collection of Renderers. The renderers are drawn in the order that they are added.

GroupRenderer.addRenderer

Description

Adds a renderer to the collection of renderers. The renderer is added to the end of collection, thus it will be drawn on top.

Syntax

boolean addRenderer(Renderer)

Arguments

Renderer a valid Renderer object to be added

Returned Value

boolean returns true if the renderer object was added, false otherwise.

See Also

Renderer
IMSMMap

Example

```
if( grouprenderer.addRenderer(myRenderer) ) {  
    alert( " renderer was added" );  
}
```

GroupRenderer.clearRenderers

Description

Remove all renderers from the collection.

Syntax

```
boolean clearRenderers()
```

Arguments

None

Returned Value

boolean	returns true if method returns successfully and all renderers are cleared, false otherwise.
---------	---

See Also

[Renderer](#)
[IMSMMap](#)

Example

```
If( Grouprenderer.clearRenderers() ) {  
    alert( "all renderers are cleared" );  
}
```

GroupRenderer.getSize

Description

Retrieves the number of renderers in the collection.

Syntax

`int getSize()`

Arguments

None

Returned Value

`int` The total number of renderers currently help in the Grouprenderer collection.

See Also

[Renderer](#)
[IMSMMap](#)

Example

```
alert ("the total number of renderers is: " + Grouprenderer.getSize());
```

GroupRenderer.indexOf

Description

Gets the index of a specified renderer within the collection.

Syntax

boolean indexOf(Renderer)

Arguments

Renderer the renderer object of the Layer

Returned Value

boolean true if method returns the index successfully, false otherwise

See Also

Renderer
IMSMMap

Example

```
If( renderer.indexOf(rendererObject)>1 ) {  
    alert( "Renderer is in the right order" );  
}
```

GroupRenderer.moveRenderer

Description

Moves a Renderer specified by the “fromPosition” index to a new position within the collection specified by “toPosition”. This causes the map to be redrawn.

Syntax

```
boolean moveRenderer(int fromPosition, int toPosition)
```

Arguments

int	fromPosition - the index of the renderer to be moved
int	toPosition - the new renderer index

Returned Value

boolean	the lower value, “None” if not set or no field specified.
---------	---

See Also

Renderer
ArcXML Programmer's Reference

Example

```
If( grouprenderer.moveRenderer( 2, 0 ) ) {  
    alert( "Renderer was moved" );  
}
```

GroupRenderer.removeRenderer

Description

Removes the renderer that matches the renderer object from the collection.

Syntax

```
boolean removeRenderer(Renderer)
```

Arguments

Renderer a valid renderer object that inherits from the Renderer

Returned Value

boolean returns true if the method was successful in removing the renderer specified, false otherwise

See Also

Renderer
IMSMMap

Example

```
If( removeRenderer( rendererObject ) ) {  
    alert( "Renderer was removed" );  
}
```

GroupRenderer.removeRendererAt

Description

The method removes the renderer, specified by the given index, from the collection.

Syntax

```
boolean removeRendererAt(int index)
```

Arguments

int index - value specifying the position of the renderer to be removed.

Returned Value

boolean returns true if the method was successful in removing the renderer specified, false otherwise

See Also

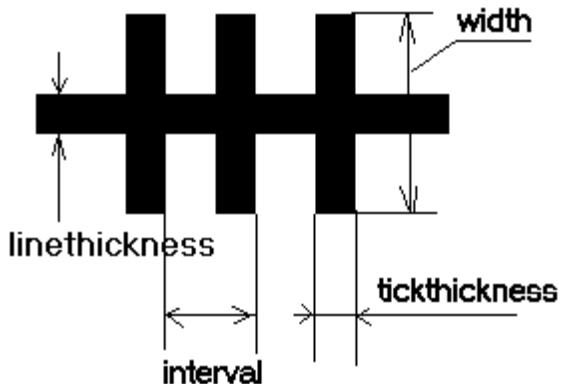
Renderer
ArcXML Programmer's Reference

Example

```
If( grouprenderer.removeRendererAt(1) ) {  
    // show popup message to reflect the renderer was removed
```

HashLineSymbol extends Symbol

The hash line symbol can be applied to draw line features. There are two styles (or types) of the hash line symbol—background and foreground. A particular style can be chosen by calling `setStyle` method. If the style is background, the symbol is drawn as a line symbol. If the style is foreground, the symbol is drawn as a railroad. There are four properties, namely width, line thickness, interval and tick thickness, which define the railroad size. These properties are shown below:



You can create an instance of the class by the `IMSMMap.createSymbol` method.

See Also

ArcXML Programmer's Reference
IMSMMap
Symbol

HashLineSymbol.getAntialiasing

Description

Retrieves the current antialiasing property value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a HashLineSymbol by default.

Syntax

boolean getAntialiasing()

Arguments

None

Returned Value

boolean true if antialiasing is to be used, false otherwise

See Also

[HashLineSymbol.setAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
alert("default value of antialiasing property for HashLineSymbol is "+ symbol.getAntialiasing());
```

HashLineSymbol.getColor

Description

Returns the current color of the hash line symbol. The default color is black.

Syntax

`String getColor()`

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[HashLineSymbol.setColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
alert("the default color of the LineSymbol is " + symbol.getColor());
```

HashLineSymbol.getInterval

Description

Returns the current interval property value of the hash line symbol in screen pixels. The default value is 8.

Syntax

```
double getInterval()
```

Arguments

None

Returned Value

the current interval value of the hash line symbol in screen points

See Also

[HashLineSymbol.setInterval](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
alert("the default value of the interval property is " + symbol.getInterval());
```

HashLineSymbol.getLineThickness

Description

Returns the current value of line thickness property in screen pixels. The default value is 1.

Syntax

```
int getLineThickness()
```

Arguments

None

Returned Value

the current value of line thickness property in screen pixels

See Also

[HashLineSymbol.setLineThickness](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
alert("the default value of the line thickness property is " + symbol.getLineThickness());
```

HashLineSymbol.getStyle

Description

Returns the current style of the hash line symbol. The default style is foreground.

Syntax

`int getStyle()`

Arguments

None

Returned Value

`int : 0 – foreground
1 - background`

See Also

[HashLineSymbol.setStyle](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");  
alert("the default style of the hash line symbol is " + symbol.getStyle());
```

HashLineSymbol.getTickThickness

Description

Returns the current value of thick thickness property in screen pixels. The default value is 1.

Syntax

```
int getThickThickness()
```

Arguments

None

Returned Value

the current value of thick thickness property in screen pixels

See Also

[HashLineSymbol.setThickThickness](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
alert("the default value of the thick thickness property is "+ symbol.getThickThickness());
double number in-between.
```

HashLineSymbol.getTransparency

Description

The getTransparency method retrieves the current transparency value set on the object. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

current hash line transparency. It must have a value between 0 and 1.0

See Also

[HashLineSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("the default transparency value of the HashLineSymbol is " + symbol.getTransparency());
```

HashLineSymbol.getWidth

Description

Returns the current width of the hash line symbol in screen pixels. The default value is 6.

Syntax

```
int getWidth()
```

Arguments

None

Returned Value

the current width of the hash line symbol in screen pixels

See Also

[HashLineSymbol.setWidth](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
alert("the default width of the HashLineSymbol is " + symbol.getWidth());
```

HashLineSymbol.setAntialiasing

Description

Sets the current antialiasing property value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a HashLineSymbol by default.

Syntax

```
boolean setAntialiasing( String enabled)
```

Arguments

enabled “true” is to set antialiasing on,
“false” is to set it off.

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setAntialiasing("true")
```

HashLineSymbol.setColor

Description

Sets the current color of the hash line symbol. The default color is black.

Syntax

```
boolean setColor( Color color)
```

Arguments

color – an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setColor(color1)
```

HashLineSymbol.setInterval

Description

Sets the current interval property value of the hash line symbol in screen pixels. The default value is 8.

Syntax

```
boolean setInterval( double intervalValue)
```

Arguments

intervalValue – the interval value in screen pixels

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getInterval](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setInterval(5);
```

HashLineSymbol.setLineThickness

Description:

Sets the current value of line thickness property in screen pixels. The default value is 1.

Syntax

```
boolean setLineThickness( int lineThickness)
```

Arguments

lineThickness – the line thickness value in false otherwise

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getLineThickness](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setLineThickness(3);
```

HashLineSymbol.setStyle

Description

Sets the current style of the hash line symbol. The default style is foreground.

Syntax

```
boolean setStyle( int style)
```

Arguments

style: 0 – foreground
1 - background

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getStyle](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setStyle(1);
```

HashLineSymbol.setTickThickness

Description

Sets the current value of thick thickness property in screen pixels. The default value is 1.

Syntax

```
boolean setTickThickness( int tickThickness)
```

Arguments

tickThickness – the tick thickness value in screen pixels

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getTickThickness](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setTickThickness(3);
```

HashLineSymbol.setTransparency

Description

Sets the current transparency value set on the object. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double number in-between.

Syntax

```
boolean setTransparency( double transparencyValue)
```

Arguments

transparencyValue – the transparency to be set. It must have a value between 0 and 1.0

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setTransparency(0.5);
```

HashLineSymbol.setWidth

Description

Sets the current width of the hash line symbol in screen pixels. The default value is 6.

Syntax

```
boolean setWidth( double width)
```

Arguments

width – the width is in screen pixels

Returned Value

true if the method was successful, false otherwise

See Also

[HashLineSymbol.getWidth](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("HASH_LINE_SYMBOL");
symbol.setWidth(3)
```

IMSMAP

Description

The applet which displays the map and functions also as “container” to other applets. IMSMAP extends javax.swing.JApplet and thus inherits all methods defined in JApplet. A table of contents, an overview map, a toolbar and a scalebar can all be displayed in the same applet next to the map. Use setBuiltInXXXX methods to enable/disable the display of these components. It is also possible to deploy these components as separate applets (IMSToc, IMSOverviewMap, IMSToolBar and IMSScaleBar). In such cases such applets must register themselves with the IMSMAP applet in order to function together. Use setExternalXXXX to register such applets.

See Also

[IMSOverviewMap](#)

[IMSScaleBar](#)

[IMSToolBar](#)

[IMSToc](#)

IMSMAP.addAVServiceLayer

Description

Adds a layer being served from an ArcViewIMS site.

Syntax

boolean addAVServiceLayer(String URL, String mapName, String viewName, boolean visibility)

Arguments

URL	The URL serving the service layer.
mapName	The name of the AVIMS map being served.
viewName	The name of the AVIMS view being served.
visibility	True if the layer is to be made visible, false if the layer is to be hidden.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addServiceLayers](#)
[IMSMAP.removeServiceLayers](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMAP;  
imsMap.addAVServiceLayer("http://spacecowboy/", "ESRICampus", "view1", display);
```

IMSMMap.addLabel

Description

Adds a label, using map units, into the acetate layer.

Syntax

boolean addLayer(String text, double x, double y)

Arguments

text The text to be labeled.

x,y The x and y coordinates for placement of the label.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.getLabelRenderingFields](#)

[IMSMMap.createAcetateLayer](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;  
var x = 711.0;  
var y = 21.0;  
imsMap.addPoint(x,y);  
imsMap.addLabel("The hideout", x, y);
```

IMSMMap.addLayer

Description

Adds a layer to the map.

Syntax

```
boolean addLayer(Layer layer)
```

Arguments

layer The Layer to be added to the map.

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;  
var newLayer = imsMap.createSDELayer("http://entropy/", "esri_sde", "hsimpson", "doh!_nuts", "lots");  
imsMap.addLayer(newLayer);
```

IMSMMap.addMapNotesLayer

Description

Adds a new MapNotes Layer to the map. The name of the layer is passed as an argument and enables the MapNotes Tool.

Syntax

```
boolean addMapNotesLayer(String name)
```

Arguments

name The name of the MapNotes layer to be added.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.startMapNotesTool](#)

Example

See next page

IMSMAP.addMapNotesLayer

Example

```
function addNewMapNotesLayer(layerName){  
    var imsMap= parent.mapFrame.document.IMSMAP;  
    var mapNotesLayers = new Array();  
    var theString = imsMap.getMapNotesLayers();  
    var theList=theString.split("|");  
    if(theList.length>0) {  
        for(var i=0;i<theList.length;i++) {  
            if(layerName==theList[I]){  
                canUse = false;  
            }  
        }  
    }  
    if(canUse){  
        imsMap.addLayer(layerName);  
    }  
}
```

IMSMMap.addMOIMSServiceLayer

Description

Adds a layer being served from a MapObjectIMS site.

Syntax

```
boolean addMOIMSServiceLayer(String URL, String serviceName, String visibility)
```

Arguments

URL The URL serving the service layer.

serviceName The name of the AVIMS service being served.

visibility True if the layer is to be made visible, false otherwise.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.addServiceLayers](#)

[IMSMMap.removeServiceLayers](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.addAVServiceLayer("http://opu/", "bookstores", "true");
```

IMSMMap.addPoint

Description

Adds a point into the acetate layer using map units.

Syntax

boolean addPoint(double x, double y)

Arguments

x,y The x and y coordinates for placement of the point.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.createAcetateLayer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var x = 3.1415;  
var y = 2.7182;  
imsmap.addPoint(x,y);
```

IMSMMap.addServiceLayers

Description

Adds all the layers from a defined ArcIMS service to the map.

Syntax

boolean addServiceLayers(String URL, String name, int serviceType, boolean visibility)

Arguments

URL	The URL serving the service layer.
name	The name of the AVIMS serivce being served.
serviceType	0 for Feature Service, 1 for Image Service.
visibility	True if the layer is to be made visible, false otherwise.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.removeServiceLayers](#)

Example

```
var serviceType = 0;  
var imsMap= parent.mapFrame.document.IMSMMap;  
imsMap.addServiceLayers("http://twoball/", "FrenchCompanies", serviceType, isVisible);
```

IMSMAP.cancel

Description

Cancels all currently occurring retrievals.

Syntax

boolean cancel()

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap= parent.mapFrame.document.IMSMAP;  
if(!imsMap.isConstructionFinished){  
    imsMap.cancel();  
}
```

IMSMAP.clearFeatureCache

Description

Clears the local cache of all feature server data layers.

Syntax

```
void clearFeatureCache()
```

Arguments

None

Returned Value

```
void
```

Example

```
function clearSelectionsAndCache() {  
    var imsMap= parent.mapFrame.document.IMSMAP;  
    imsMap.clearSelections();  
    imsMap.clearFeatureCache();  
}
```

IMSMAP.clearSelections

Description

Clears all selections made on the map. Features are no longer highlighted and the selection set is made null.

Syntax

boolean clearSelections()

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
function clearSelectionsAndCache() {  
    var imsMap= parent.mapFrame.document.IMSMAP;  
    imsMap.clearSelections();  
    imsMap.clearFeatureCache();  
}
```

IMSMMap.closeProject

Description

Closes the current project.

Syntax

boolean closeProject()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.loadProject](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;  
imsMap.closeProject();
```

IMSMMap.copyMapImageToFile

Description

Copies the current map as an image to a file in JPEG format.

Syntax

```
boolean copyMapImageToFile(String filename)
```

Arguments

filename Pathname of the file to be written.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.copyTOCImageToFile](#)

Example

```
var filePath = “/spacecowboy1/pub/images/niftymap.jpg”;  
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.copyMapImageToFile(filePath);
```

IMSMMap.copyTOCImageToFile

Description

Copies the current TOC as an image to a file in JPEG format.

Syntax

boolean copyTOCImageToFile(String filename)

Arguments

filename Pathname of the file to be written.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.copyMapImageToFile](#)

Example

```
var filePath = “/spacecowboy1/pub/images/niftytoc.jpg”;  
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.copyMapImageToFile(filePath);
```

IMSMMap.createAcetateLayer

Description

Creates an acetate layer on the map. If the acetate layer already exists, this method does nothing.

Syntax

```
boolean createAcetateLayer()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.removeAcetateLayer](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;  
if(!hasAcetate) {  
    imsMap.createAcetateLayer();  
    hasAcetate=true;  
}
```

IMSMMap.createCollection

Description

Creates an empty Collection.

Syntax

Collection createCollection()

Arguments

None

Returned Value

Collection Returns a new, empty Collection.

See Also

[IMSMMap.getCollectionElementAt](#)
[Collection](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var theCollection = imsMap.createCollection();  
theCollection.addStringElement("John Smith");  
theCollection.addStringElement("Mary Smith");
```

IMSMMap.createColor

Description

Creates a Color defined by the RGB arguments.

Syntax

Color createColor(int r, int g, int b)

Arguments

r,g,b Values for red, green, and blue using standard RGB form (all values must be between 0-255).

Returned Value

Color Returns a new Color defined by the given RGB values.

See Also

Color

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;
var layer =imsMap.getSelectedLayer();
var renderer =imsMap.createRenderer("SIMPLE_RENDERER");
var symbol =imsMap.createSymbol("MARKER_SYMBOL");
var newColor =imsMap.createColor(0,255,255);
symbol.setColor(newColor);
symbol.setSize(9);
symbol.setStyle(1);
renderer.setSymbol(symbol);
imsMap.setLayerRenderer(layer, renderer);
imsMap.redraw();
```

IMSMMap.createExtent

Description

Creates an Extent with the coordinate parameters (x and y) and size parameters (height and width).

Syntax

Extent createExtent(double x, double y, double width, double height)

Arguments

x,y The x and y coordinates for south-west corner of desired extent.
width, height The width and height of the desired extend.

Returned Value

Extent Returns a new Extent defined by the given values.

See Also

[IMSMMap.setExtent](#)
[IMSMMap.setExtentLimit](#)
[Extent](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;  
var newExtent = imsMap.createExtent(-110.456, 33.765, 1.5, 0.75);  
imsMap.setExtent(newExtent);
```

IMSMMap.createNameValuePair

Description

Creates and returns a NameValuePair.

Syntax

`NameValuePair createNameValuePair(String name, String value)`

Arguments

name Name
value Value

Returned Value

`NameValuePair` A NameValuePair with the given name and value.

See Also

`NameValuePair`

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;  
var theCollection = imsMap.createCollection();  
var newPair = imsMap.createNameValuePair("John Smith", "President");  
theCollection.addNameValuePairElement(newPair);
```

IMSMAP.createRenderer

Description

Creates a Renderer of the type defined in the parameter. If parameter is not recognized then null is returned.

Syntax

Renderer createRenderer(String type)

Arguments

type The type of renderer to be created. See the list of returned values for acceptable arguments.

Returned Value

Renderer	The Renderer of the type defined in the parameter. The following is a list of the object types returned by each:
“SIMPLE_RENDERER”	SimpleRenderer
“VALUemap_RENDERER”	ValueMapRenderer
“SCALE_DEPENDENT_RENDERER”	ScaleDependentRenderer
“GROUP_RENDERER”	GroupRenderer
“LABEL_RENDERER”	LabelRenderer
“VALUemap_LABEL_RENDERER”	ValueMapLabelRenderer

See Also

Renderer

Example

see the next page

IMSMAP.createRenderer

Example

```
var imsMap= parent.mapFrame.document.IMSMAP;
var layer = imsMap.getSelectedLayer();
var renderer = imsMap.createRenderer("SIMPLE_RENDERER");
var symbol = imsMap.createSymbol("MARKER_SYMBOL");
var newColor = imsMap.createColor(0,255,255);
symbol.setColor(newColor);
symbol.setSize(9);
symbol.setStyle(1);
renderer.setSymbol(symbol);
imsMap.setLayerRenderer(layer, renderer);
imsMap.redraw();
```

IMSMMap.createSDELayer

Description

Returns a Layer for the layer specified by the parameters or null if the connection fails or other errors occur.

Syntax

```
Layer createSDELayer(String server, String instance, String userName, String pswd, String  
layerName)
```

Arguments

server	The name of the server.
instance	The instance of the SDE server.
userName	Username.
pswd	Password.
layerName	Name of the layer to be created.

Returned Value

Layer The requested SDE layer as a Layer. Null is returned if connection failed or an error occurred.

See Also

[Layer](#)

Example

See next page

IMSMAP.createSDELayer

Example

```
var imsMap= parent.mapFrame.document.IMSMAP;  
var newLayer = imsMap.createSDELayer("http://entropy/", "esri_sde", "hsimpson", "doh!_nuts", "lots");  
if(newLayer != null){  
    imsMap.addLayer(newLayer);  
}else{  
    alert("Layer is null.");  
}
```

IMSMAP.createSymbol

Description

Creates a Symbol object of the type defined in the parameter. If parameter is not recognized then null is returned.

Syntax

```
Symbol createSymbol(String type)
```

Arguments

type The type of symbol to be created. See the list of returned values for acceptable arguments.

Returned Value

Symbol	The Symbol object of the type defined in the parameter. The following is a list of what type of object is returned by each:
“MARKER_SYMBOL”	MarkerSymbol
“LINE_SYMBOL”	LineSymbol
“POLYGON_SYMBOL”	PolygonSymbol
“TRUETYPE_MARKER_SYMBOL”	TrueTypeMarkerSymbol
“RASTER_MARKER_SYMBOL”	RasterMarkerSymbol
“HASH_LINE_SYMBOL”	HashLineSymbol
“RASTER_FILL_SYMBOL”	RasterFillSymbol
“GRADIENT_FILL_SYMBOL”	GradientFillSymbol
“SHIELD_SYMBOL”	ShieldSymbol
“TEXT_SYMBOL”	TextSymbol
“RASTER_SHIELD_SYMBOL”	RasterShieldSymbol
“CALLOUT_MARKER_SYMBOL”	CalloutMarkerSymbol
“FILL_SYMBOL”	FillSymbol
	If parameter is invalid, null is returned.

IMSMAP.createSymbol

See Also

[Symbol](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMAP;
var layer = imsMap.getSelectedLayer();
var renderer = imsMap.createRenderer("SIMPLE_RENDERER");
var symbol = imsMap.createSymbol("MARKER_SYMBOL");
var newColor = imsMap.createColor(0,255,255);
symbol.setColor(newColor);
symbol.setSize(9);
symbol.setStyle(1);
renderer.setSymbol(symbol);
imsMap.setLayerRenderer(layer, renderer);
imsMap.redraw();
var newColor = imsMap.createColor(179,137,191);
symbol.setColor(newColor);
symbol.setSize(9);
```

IMSMAP.createValueRange

Description

Creates a ValueRange given the parameters.

Syntax

ValueRange createValueRange(Layer layer, String fieldName, String lower, String upper)

Arguments

layer	The layer for which the ValueRange is created.
fieldName	The name of the field of interest.
lower, upper	The lower and upper values of the range.

Returned Value

ValueRange A ValueRange defined by the parameters.

See Also

ValueRange

Example

See next page

IMSMAP.createValueRange

Example

```
var imsMap= parent.mapFrame.document.IMSMAP;
var selectedLayer = imsMap.getSelectedLayer();
var fieldString = selectedLayer.getFieldNames();
var nameField = null;
if(fieldString!="") {
    var theList = fieldString.split("|");
    if(theList.length>0) {
        for(var i=0;i<theList.length;i++) {
            if(theList[i].toUpperCase().indexOf("NAME") != -1){
                nameField = theList[i];
                break;
            }
        }
    }
}
if(nameField != null){
    var valueRange = imsMap.createValueRange(selectedLayer, nameField, "alpha", "omega");
}
```

IMSMAP.displayAttributesUI

Description

Displays the AttributesUI. This UI is used after features have been selected to display the attributes of those features.

Syntax

boolean displayAttributesUI()

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
function queryStringAlways(query) {  
    if(checkSelectedLayer()) {  
        var layer = parent.mapFrame.IMSMAP.getSelectedLayer();  
        var result = layer.select(query);  
        displayAttributesUI();  
    }  
}
```

IMSMAP.displayBufferUI

Description

Displays the BufferUI. This UI is used to buffer the selected features.

Syntax

```
boolean displayBufferUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
if(checkSelectedLayer()) {  
    var returnString = parent.mapFrame.IMSMAP.getSelectedLayer().getSelectionSet();  
    if(returnString!="") {  
        parent.mapFrame.IMSMAP.displayBufferUI();  
    } else {  
        alert("No features selected in Active Layer");  
    }  
}
```

IMSMMap.displayCatalogUI

Description

Displays the CatalogUI. This UI allows users to add layers to map.

Syntax

```
boolean displayCatalogUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.displayCatalogUI();
```

IMSMAP.displayFindUI

Description

Displays the FindUI. This UI allows users to perform a simple query on a layer.

Syntax

```
boolean displayFindUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.displayFindUI();IMSMAP.displayGeocodeUI
```

IMSMAP.displayGeocodeUI

Description

Displays the GeocodeUI. This UI allows users to locate addresses on a layer which is geocoded.

Syntax

boolean displayGeocodeUI()

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.displayGeocodeUI();
```

IMSMMap.displayLayerPropertiesUI

Description

Displays the LayerPropertiesUI. This UI allows user to change the properties of the selected layer.

Syntax

```
boolean displayLayerPropertiesUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.displayLayerPropertiesUI();
```

IMSMAP.displayMapTipsUI

Description

Displays the MapTipsUI. This UI allows users to define layers on which map tips are displayed.

Syntax

```
boolean displayMapTipsUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.displayMapTipsUI();
```

IMSSMap.displayOpenProjectUI

Description

Displays the OpenProjectUI. This UI allows user to open a project file from the local system.

Syntax

```
boolean displayOpenProjectUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSSMap;  
imsMap.displayOpenProjectUI();
```

IMSMAP.displayPrintUI

Description

Displays the PrintUI. This UI allows user to print the current map and TOC.

Syntax

boolean displayPrintUI()

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.displayPrintUI();
```

IMSMAP.displayQueryBuilderUI

Description

Displays the QueryBuilderUI. This UI allows a user to precisely query on the map layers using SQL strings.

Syntax

```
boolean displayQueryBuilderUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
if(checkSelectedLayer()) {  
    parent.mapFrame.IMSMAP.displayQueryBuilderUI();  
}
```

IMSMMap.displaySaveAsProjectUI

Description

Displays the SaveAsProjectUI. This UI allows user to save current project to an AXL file.

Syntax

```
boolean displaySaveAsProjectUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.displaySaveAsProjectUI();
```

IMSMMap.displayStoredQueryUI

Description

Displays the StoredQueryUI. This UI allows users to perform a search on a predefined query.

Syntax

```
boolean displayStoredQueryUI()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
if(checkSelectedLayer()) {  
    parent.mapFrame.IMSMMap.displayStoredQueryUI();  
}
```

IMSMAP.enableFunction

Description

Enables/disables a function specified by its function ID. The tools and menu items in the toolbar, menu bar, and popup menus will also be dynamically updated to reflect the changes. However, simply enabling a function will not necessarily make the respective tool appear on the interface. An example as to how to add a tool to the interface can be found in Chapter 2 in the section, “Adding a tool to the toolbar.”

Some of the function IDs reflect groups of functions rather than single functions. The following is a list of all the functions and function groups with their respective IDs:

- 10 Project Functions (11, 12, 13, 14)
- 11 Open project
- 12 Save Project and Save Project As
- 13 Close Project
- 14 Print

- 20 All Zoom Functions(21, 22, 23, 24, 25, 26, 27, 28)
- 21 Go Back To Extent
- 22 Go Forward To Extent
- 23 Full Extent
- 24 Zoom To Active Layer
- 25 Zoom In
- 26 Zoom Out
- 27 Pan
- 28 Direction – Pan North, Pan South, Pan East, Pan West

- 30 All Query Functions(31, 32, 33, 34, 36, 37)
- 31 Identify
- 32 Measure (Feet, Miles, Meters, Kilometers)
- 33 Select (Circle, Rectangle, Line, Polygon)
- 34 Find and Query Builder
- 36 Clear Selections
- 37 Stored Query

IMSMMap.enableFunction

- 40 All Layer Actions(41, 42, 43, 44, 45, 46, 47, 48)
- 41 Layer Properties and Clear Labels
- 42 Edit Notes Tool
- 43 Layer Classification
- 44 Add Layer, Remove Layer, and Move Layer
- 45 Toggle Legend
- 46 Map Notes Tool
- 47 Geocoding
- 48 Map Tips

Syntax

boolean enableFunction(int functionID, boolean value)

Arguments

- functionID Integer value is the ID of the function to enable (see above). Non-recognized values will have no effect, but will not return a false.
- value True enables, false disables.

Returned Value

- boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.enableFunction(10, true); //Enables all Project Functions  
imsMap.enableFunction(37, false); //Disables StoredQueries
```

IMSMMap.getAppletInfo

Description

Returns information about the applet being used.

Syntax

`String getAppletInfo()`

Arguments

None

Returned Value

`String` Information about the applet being used.

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
alert(imsMap.getAppletInfo());
```

IMSMMap.getBGColor

Description

Returns a String of the RGB values of the background color delimited by commas without spaces.

Syntax

String getBGColor()

Arguments

None

Returned Value

String String of the RGB values for the background color (e.g. “170,150,200”).

See Also

[IMSMMap.setBGColor](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var theToc = imsMap.getToc();
var mapBGColor = imsMap.getBGColor();
var rgb = mapBGColor.split(",");
if(rgb.length == 3){
    theToc.setBGColor(rgb[0],rgb[1],rgb[2]);
} else {
    alert("Error in obtaining Background Color of Map");
}
```

IMSMMap.getCartScale

Description

Returns the RF (Representative Fraction) for the map at its current scale.

Syntax

double getCartScale()

Arguments

None

Returned Value

double The current map scale. If current Extent is null, 1 is returned.

See Also

[IMSMMap.setMapUnit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var cscale = imsMap.getCartScale();  
alert("The current cartographic scale is: " + cscale);
```

IMSMMap.getCollectionElementAt

Description

Returns the element of a Collection at a given index. Note that Collections can only contain Strings and NameValuePair objects.

Syntax

Object getCollectionElementAt(Collection collection, int index)

Arguments

collection The Collection containing the String or NameValuePair desired.
index The index of the desired object.

Returned Value

Object The String or NameValuePair at the given index. If this value is invalid, then null is returned.

See Also

[IMSMMap.createCollection](#) Collection
[NameValuePair](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var theCollection = imsMap.createCollection();
var newPair = imsMap.createNameValuePair("President", "John Smith");
var newPair2 = imsMap.createNameValuePair("VicePresident", "Mary Smith");
theCollection.addNameValuePairElement(newPair);
theCollection.addNameValuePairElement(newPair2);
var pairFromCollection = imsMap.getCollectionElementAt(theCollection,1);
alert(pairFromCollection.getValue());
```

IMSMMap.getExtent

Description

Returns the current extent of the map as an Extent.

Syntax

Extent getExtent()

Arguments

None

Returned Value

Extent Returns the current extent of the map as an Extent. If current Extent is null or if error occurs then null is returned.

See Also

[IMSMMap.getFullExtent](#)
[IMSMMap.setExtent](#)
[IMSMMap.setExtentLimit](#)
[Extent](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var currentExtent = imsMap.getExtent();
var theExtentString = "Current Extent:\n";
theExtentString += currentExtent.getXMin() + "," + currentExtent.getYMin() + "- ";
theExtentString += currentExtent.getXMax() + "," + currentExtent.getYMax() + "\n";
alert(theExtentString);
```

IMSMMap.getFeatureLayerNames

Description

Returns the feature layer names that are available in the map as a Collection of String elements. The names can be accessed by first finding how many elements there are using the Collection's getSize() method, then access each String using the Collection's getStringElement() method.

Syntax

Collection getFeatureLayerNames()

Arguments

None

Returned Value

Collection Returns the names of the feature layers as a Collection. If no feature layers exist, an empty Collection is returned.

See Also

Collection

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var layerNames = imsMap.getFeatureLayerNames();  
var numLayers = layerNames.getSize();  
alert("There are " + numLayers + " Feature Layers.");
```

IMSMMap.getFullExtent

Description

Returns the full extent of the map as an Extent.

Syntax

Extent getFullExtent()

Arguments

None

Returned Value

Extent Returns the full extent of the map as an Extent.

See Also

[IMSMMap.getExtent](#)
[IMSMMap.setExtent](#)
[IMSMMap.setExtentLimit](#)
[Extent](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var fullExtent = imsMap.getFullExtent();  
imsMap.setExtent(fullExtent);
```

IMSMMap.getIdentifiableFields

Description

Returns all of the fields that will be displayed when an identify operation is performed on the given Layer, and returns the field names String elements within a Collection.

Syntax

Collection getIdentifiableFields(Layer layer)

Arguments

layer The Layer to find the identifiable fields on.

Returned Value

Collection Returns a Collection which contains all of the identifiable fields in the layer passed to this function. If none exist, an empty Collection element is returned.

See Also

Collection
Layer

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var selectedLayer = imsMap.getSelectedLayer();
var iFields = imsMap.getIdentifiableFields(selectedLayer);
var numberofFields = iFields.getSize();
alert("This layer has " + numberofFields + " identifiable fields.");
```

IMSMMap.getLabelRenderingFields

Description

Returns a Collection consisting of all of the fields of the given layer which are used for labeling. The Collection consists of field names as String elements within a Collection. This method cannot be used inside a Scale Dependent Renderer.

Syntax

```
Collection getLabelRenderingFields(Layer layer, Renderer renderer)
```

Arguments

Layer	The Layer searched for fields which are rendered by a LabelRenderer.
Renderer	The renderer which is rendering the labels on the layer parameter.

Returned Value

Collection	A collection of the field names of the fields which have labels rendered on them as a Collection. If none exist, an empty Collection is returned.
------------	---

See Also

Collection
LabelRenderer

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var selectedLayer = imsMap.getSelectedLayer();
var labelRenderer = imsMap.getLayerLabelRenderer(selectedLayer);
if(labelRenderer != null){
    var lrFields = imsMap.getLabelRenderingFields(selectedLayer, labelRenderer);
    var numberofLrFields = lrFields.getSize();
    alert("This layer has " + numberofLrFields + " label rendering fields.");
}
```

IMSMMap.getLayer

Description

Returns the layer with the given name or null if no such layer exists.

Syntax

Layer getLayer(String layerName)

Arguments

LayerName String, the name of the layer to be searched.

Returned Value

Layer Returns the matching Layer or null if no such layer is found.

See Also

[IMSMMap.getLayerCount](#)
[IMSMMap.getLayerNames](#)
[IMSMMap.getLayerNamesFromService](#)

Example

```
var theNameToLookFor = "Internet_Cafes";
var imsMap = parent.mapFrame.document.IMSMMap;
var layer = imsMapGetLayer(theNameToLookFor);
if(layer == null){
    alert("Layer not found.");
}
```

IMSMMap.getLayerCount

Description

Returns the current number of layers. Note that a group layer is considered one layer and its sublayers are not counted individually. (See the documentation for Layer for more information on sublayers.)

Syntax

```
int getLayerCount();
```

Arguments

None

Returned Value

int Returns the number of layers in the map excluding any sublayers.

See Also

[IMSMMap.getLayerNames](#)
[Layer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var numberofLayers = imsMap.getLayerCount();  
for(int i=0; i < numberofLayers; i++){  
    imsMap.removeLayerByIndex(i);  
}
```

IMSMMap.getLayerExtent

Description

Returns the extent of the layer defined by the parameter.

Syntax

Extent getLayerExtent(String layerName)

Arguments

layerName The layer name as a String.

Returned Value

Extent Returns an Extent which is the current extent of the defined layer. If Extent is null or if an error occurs, null is returned.

See Also

[IMSMMap.getExtent](#)
[IMSMMap.getFullExtent](#)
[IMSMMap.setExtent](#)
[IMSMMap.setExtentLimit](#)
[Extent](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var layer = imsMap.getSelectedLayer();  
var extent = imsMap.getLayerExtent(layer.getName());  
imsMap.setExtent(extent);
```

IMSMMap.getLayerLabelRenderer

Description

Returns the Renderer used to draw the Labels for the layer defined in the parameter. This method cannot be used inside a Scale Dependent Renderer.

Syntax

Renderer getLayerLabelRenderer(Layer layer)

Arguments

Layer The Layer from which to get the renderer.

Returned Value

Renderer Returns a Renderer which does the feature label rendering for that layer. Null is returned if there is no label renderer on the layer.

See Also

[IMSMMap.getLabelRenderingFields](#)
[IMSMMap.getLayerLabelRendererType](#)
[IMSMMap.setLayerLabelRenderer](#)
[LabelRenderer](#)
[ValueMapLabelRenderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var selectedLayer = imsMap.getSelectedLayer();
var labelRenderer = imsMap.getLayerLabelRenderer(selectedLayer);
if(labelRenderer != null){
    var lrFields = imsMap.getLabelRenderingFields(selectedLayer, labelRenderer);
    var numberOfLrFields = lrFields.getSize();
    alert("This layer has " + numberOfLrFields + " label rendering fields.");
}
```

IMSMMap.getLayerLabelRendererType

Description

Returns the type of label renderer that is used by this layer. This method cannot be used inside a Scale Dependent Renderer.

Syntax

```
String getLayerLabelRendererType(Layer layer)
```

Arguments

layer The Layer to find the renderer type on.

Returned Value

String Returns string with the type of Renderer used to render labels on the defined layer.
Known return String values: LABEL_RENDERER, VALUEMAP_LABEL_RENDERER, null
(if no label renderer exists for this layer).

See Also

[IMSMMap.getLabelRenderingFields](#)
[IMSMMap.getLayerLabelRendererType](#)
[IMSMMap.setLayerLabelRenderer](#)
[LabelRenderer](#)
[ValueMapLabelRenderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var labelRendererType = imsMap.getLayerLabelRendererType(imsMap.getSelectedLayer());
```

IMSMMap.getLayerNames

Description

Returns the names of the current map layers as Strings in a Collection. Note that a group layer is considered one layer and its sublayers are not counted individually. (See the documentation for Layer for more information on sublayers.)

Syntax

```
Collection getLayerNames();
```

Arguments

None

Returned Value

Collection	Returns a Collection which contains the names of all the layers excluding sublayers. If none exist then an empty Collection is returned.
------------	--

See Also

[IMSMMap.getLayerCount](#)
[IMSMMap.getLayerNamesFromService](#)
[Layer](#)

Example

See next page

IMSMAP.getLayerNames

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;
var numberOfLayers = imsMap.getLayerCount();
var namesOfLayers = imsMap.getLayerNames();
var nameList = "";
for (var i=0; i < numberOfLayers; i++){
    nameList = nameList + namesOfLayers.getStringElement(i);
    if(i < (numberOfLayers-1)){
        nameList = nameList + ",";
        if(i == (numberOfLayers-2)){
            nameList = nameList + "and ";
        }
    }
}
alert("The layers that are included in this map are " + nameList + ".");
```

IMSMMap.getLayerNamesFromService

Description

Returns the names of the layers from the defined service. Note that a group layer is considered one layer and its sublayers are not counted individually. (See the documentation for Layer for more information on sublayers.)

Syntax

```
Collection getLayerNamesFromService(String url, String service);
```

Arguments

url	The URL of the server on which the service resides.
Service	The name of the service from which the layer names will be returned.

Returned Value

Collection	Returns a Collection which contains the names of all the layers excluding sublayers of the defined service. If the service info given is null, then null is returned.
------------	---

See Also

[IMSMMap.getLayerCount](#)
[IMSMMap.getLayerNames](#)
[Layer](#)

Example

see the next page

IMSSMap.getLayerNamesFromService

Example

```
var SERVICE = "SanFranFeature";
var URL = "http://kat/";
var imsMap = parent.mapFrame.document.IMSSMap;
var namesOfLayers = imsMap.getLayerNamesFromService(URL, SERVICE);
if(namesOfLayers!=null) {
    var numberOfLayers = namesOfLayers.getSize();
    var nameList = "";
    for (var i=0; i < numberOfLayers; i++){
        nameList = nameList + namesOfLayers.getStringElement(i);
        if(i <(numberOfLayers-1)){
            nameList = nameList + ", ";
            if(i == (numberOfLayers-2)){
                nameList = nameList + "and ";
            }
        }
    }
    alert("The layers that are included in this on " + SERVICE + " on " + URL + " are " +
nameList + ". ");
}
} else {
    alert("No layers from " + SERVICE + " on " + URL + ".");
}
```

IMSMMap.getLayerRenderer

Description

Returns the renderer assigned to this layer as a Renderer.

Syntax

Renderer getLayerRenderer(Layer layer)

Arguments

layer This is the layer which is being rendered.

Returned Value

Renderer Returns a Renderer which is the renderer for the layer defined. If the layer is invalid, null is returned.

See Also

[IMSMMap.setLayerRenderer](#)
[Renderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var layer = imsMap.getSelectedLayer();  
var renderer = imsMap.getLayerRenderer(layer);  
if(renderer == null){  
    alert("No renderer for layer: " + layer);  
}
```

IMSMMap.getLayerRendererAt

Description

Returns the renderer at a given index within a group renderer.

Syntax

Renderer getLayerRendererAt(GroupRenderer groupRenderer, int index)

Arguments

groupRenderer The GroupRenderer to be indexed.
index The index of renderer to be obtained.

Returned Value

Renderer Returns the renderer at the given index. If the renderer passed to this method is not a GroupRenderer or if the index is invalid, null is returned.

See Also

[IMSMMap.getLayerRenderer](#)
[IMSMMap.getLayerRendererAtType](#)
[IMSMMap.setLayerRenderer](#)
[GroupRenderer](#)

Example

See next page

IMSMAP.getLayerRendererAt

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;
var INDEX = 0;
var layer = imsMap.getSelectedLayer();
var gRenderer = imsMap.getLayerRenderer(layer);
if(gRenderer==null){
    alert("ERROR: No renderer for layer: " + layer);
} else{
    var type = imsMap.getLayerRendererAtType(gRenderer, INDEX);
    if(type==null){
        alert("Defined layer is not rendered by a group renderer or index is invalid.");
    } else if(type=="VALUemap_LABEL_RENDERER"){
        var renderer = imsMap.getLayerRendererAt(gRenderer, INDEX);
        var symbol = imsMap.getValueMapLabelDefaultSymbol(renderer);
        ...
    }
}
```

IMSMMap.getLayerRendererAtType

Description

Returns the type of the renderer at a given index within a group renderer.

Syntax

```
String getLayerRendererAt(GroupRenderer groupRenderer, int index)
```

Arguments

groupRenderer The GroupRenderer to be indexed.
index The index of renderer to be obtained.

Returned Value

String Returns the type of renderer at the given index. If the renderer passed to this method is not a GroupRenderer or if the index is invalid, null is returned. Otherwise, one of the following six Strings is returned: “SIMPLE_RENDERERT”, “VALUemap_RENDERERT”, “SCALE_DEPENDENT_RENDERERT”, “GROUP_RENDERERT”, “LABEL_RENDERERT”, “VALUemap_LABEL_RENDERERT”.

See Also

[IMSMMap.getLayerRenderer](#)
[IMSMMap.getLayerRendererAt](#)
[IMSMMap.getLayerRendererType](#)

Example

See next page

IMSMAP.getLayerRendererAtType

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;
var INDEX = 0;
var layer = imsMap.getSelectedLayer();
var gRenderer = imsMap.getLayerRenderer(layer);
if(gRenderer==null){
    alert("ERROR: No renderer for layer: " + layer);
}else{
    var type = imsMap.getLayerRendererAtType(gRenderer, INDEX);
    if(type==null){
        alert("Defined layer is not rendered by a group renderer or index is invalid.");
    }else if(type=="VALUemap_LABEL_RENDERER"){
        var renderer = imsMap.getLayerRendererAt(gRenderer, INDEX);
        var symbol = imsMap.getValueMapLabelDefaultSymbol(renderer);
        ...
    }
}
```

IMSMMap.getLayerRendererType

Description

Returns the type of renderer for the given layer as a String.

Syntax

String getLayerRendererType(Layer layer)

Arguments

layer The Layer for which information is desired..

Returned Value

String Returns the type of renderer used on the given layer. If the layer is invalid null is returned. Otherwise one of the following four Strings is returned: “SIMPLE_RENDERER”, “VALUemap_RENDERER”, “SCALE_DEPENDENT_RENDERER”, “GROUP_RENDERER”.

See Also

[IMSMMap.getLayerRenderer](#)
[IMSMMap.getLayerRendererAt](#)
[IMSMMap.getLayerRendererAtType](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var layer = imsMap.getSelectedLayer();
if(layer != null){
  var type = imsMap.getLayerRendererType(layer);
  alert("The layer is rendererd by a " + type);
}
```

IMSMMap.getMapNotesLayers

Description

Returns all the available MapNotes layers as a String with the names delimited by a pipe. If no layer exists then an empty string is returned.

Syntax

```
String getMapNotesLayers()
```

Arguments

None

Returned Value

String Returns the names of the MapNotes layers delimited by a pipe (“|”). An empty String is returned if no layers exist.

See Also

[IMSMMap.addMapNotesLayer](#)
[IMSMMap.selectMapNotesLayer](#)
[IMSMMap.setMapNotesFolder](#)
[IMSMMap.setMapNotesFolderOnServer](#)
[IMSMMap.setMapNotesSubmitLimit](#)
[IMSMMap.startMapNotesTool](#)
[IMSMMap.stopMapNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var mnLayers = imsMap.getMapNotesLayers();  
var layArray = mnLayers.split("|");  
var mnLayerCount = layArray.length;  
if(mnLayerCount == 0){  
    alert("There are no MapNotes layers.");  
}
```

IMSMMap.getMapServiceExtent

Description

Returns the extent of the given service as an Extent. If either of the parameters is invalid or if an error occurs then null is returned.

Syntax

```
Extent getMapServiceExtent(String url, String service)
```

Arguments

url The URL of the service.
service The name of the service.

Returned Value

Extent Returns the extent of the given service as an Extent. Null is returned if parameters are invalid or if an error occurs.

See Also

[IMSMMap.setExtent](#)
[IMSMMap.setExtentLimit](#)
[Extent](#)

Example

```
var SERVICE = "theatres";
var URL = "http://decartes/";
var imsMap = parent.mapFrame.document.IMSMMap;
var extent = imsMap.getMapServiceExtent(URL, SERVICE);
imsMap.setExtent(extent);
alert("Extent reset.");
```

IMSMMap.getMapUnit

Description

Returns the map unit of the current map.

Syntax

```
int getMapUnit()
```

Arguments

None

Returned Value

int	Returns the current map unit as an int. The following are all the known values:
Decimal Degrees	0
Centimeters	1
Inches	2
Feet	3
Miles	4
Meters	5
Kilometers	6
Unknown	7

See Also

[IMSMMap.setMapUnit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var MAP_UNIT = imsMap.getMapUnit();
```

IMSMMap.getMeasureUnit

Description

Returns the measure unit.

Syntax

```
int getMeasureUnit()
```

Arguments

None

Returned Value

int Returns the measure unit as an int. If error occurs, -1 is returned.

See Also

[IMSMMap.setMeasureUnit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var MEASURE_UNIT = imsMap.getMeasureUnit();
```

IMSMMap.getOverviewMap

Description

Returns the IMSOverviewMap associated with the instance of IMSMap calling this method.

Syntax

IMSOverviewMap getOverviewMap()

Arguments

None

Returned Value

IMSOverviewMap Returns the IMSOverviewMap associated with this IMSMap. If no IMSOverviewMap exists, then one is created.

See Also

[IMSOverviewMap](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var ovMap = imsMap.getOverviewMap();  
ovMap.redraw();
```

IMSMMap.getRendererSymbol

Description

Returns a new Symbol for the given renderer. If that Renderer is a ValueMapRenderer or a ValueMapLabelRenderer then a non-null value is needed to produce the Symbol corresponding to that value. Otherwise, the value is ignored.

Syntax

```
Symbol getRendererSymbol(Renderer renderer, String value)
```

Arguments

Renderer	The Renderer used to create the symbol.
Value	Value used to generate the Symbol. Only need be non-null if the Renderer is a ValueMapRenderer or a ValueMapLabelRenderer. If null for these types, then null is returned. Only SimpleRenderer, ValueMapRenderer, LabelRenderer, and ValueMapLabel Renderer are recognized by this method.

Returned Value

Symbol A Symbol for the given Renderer and value (if applicable). If error occurs null is returned.

See Also

[IMSMMap.createSymbol](#)
[IMSMMap.getRendererSymbolType](#)
[Symbol](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var labelName = "Burried Treasure.";
var renderer = imsMap.getLayerRenderer(imsMap.getSelectedLayer());
if(renderer != null){
    var XmarksTheSpot = imsMap.getRendererSymbol(renderer, labelName);
    ...
}
```

IMSMMap.getRendererSymbolType

Description

Returns a new Symbol for the given renderer. If that Renderer is a ValueMapRenderer or a ValueMapLabelRenderer then a non-null value is needed to produce the Symbol corresponding to that value. Otherwise, the value is ignored.

Syntax

```
String getRendererSymbolType(Renderer renderer, String value)
```

Arguments

Renderer	The Renderer to be used to create the symbol.
Value	Value used to generate the Symbol object. Only need be non-null if the Renderer is a ValueMapRenderer or a ValueMapLabelRenderer. If null for these types, then null is returned. Only SimpleRenderer, ValueMapRenderer, LabelRenderer, and ValueMapLabel Renderer are recognized by this method.

Returned Value

String	Returns a String for the symbol defined by the Renderer and the value parameters. If no match is found, the String “None” is returned. Otherwise, one the following 13 Strings will be returned: “MARKER_SYMBOL”, “LINE_SYMBOL”, “POLYGON_SYMBOL”, “TRUETYPE_MARKER_SYMBOL”, “RASTER_MARKER_SYMBOL”, “HASH_LINE_SYMBOL”, “RASTER_FILL_SYMBOL”, “GRADIENT_FILL_SYMBOL”, “SHIELD_SYMBOL”, “TEXT_SYMBOL”, RASTER_SHIELD_SYMBOL”, “CALLOUT_MARKER_SYMBOL”, FILL_SYMBOL”.
--------	--

See Also

[IMSMMap.createSymbol](#)
[IMSMMap.getRendererSymbol](#)
Symbol

IMSMAP.getRendererSymbolType

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
var renderer = imsMap.getLayerRenderer(imsMap.getSelectedLayer());  
if(renderer != null) alert(imsMap.getRendererSymbolType(renderer, labelName));
```

IMSMMap.getScale

Description

Returns the current scale (map units per pixel) of the map.

Syntax

```
double getScale()
```

Arguments

None

Returned Value

double The current scale of the map.

See Also

[IMSMMap.getMapUnit](#)
[IMSMMap.setMapUnit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var scale = imsMap.getScale();
```

IMSScaleBar

Description

Returns the current IMSScaleBar. If it does not exist, then one is created.

Syntax

IMSScaleBar getScaleBar()

Arguments

None

Returned Value

IMSScaleBar Returns the current IMSScaleBar item. If it does not exist, then one is created.

See Also

IMSScaleBar

Example

```
var imsMap = parent.mapFrame.document.IMSSMap;  
var scaleBar = imsMap.getScaleBar();  
scaleBar.setBGColor(30, 60, 90);  
scaleBar.refresh();
```

IMSMMap.getSelectedLayer

Description

Returns the currently selected Layer. If no layer is selected or if an error occurs then null is returned.

Syntax

Layer getSelectedLayer()

Arguments

None

Returned Value

Layer Returns the currently selected Layer. If no layer is selected or if an error occurs then null is returned

See Also

[IMSMMap.setSelectedLayer](#)

[IMSMMap.getLayer](#)

[Layer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var layer = imsMap.getSelectedLayer();  
if(layer != null){  
    var type = imsMap.getLayerRendererType(layer);  
    alert("The layer is rendererd by a " + type);  
}
```

IMSMMap.getSelectedTool

Description

Returns the name of currently selected tool from the map. If no tool is selected, then null is returned.

Syntax

`String getSelectedTool()`

Arguments

None

Returned Value

`String` Returns the name of the currently selected tool. If no tool is selected, null is returned.
Otherwise, one of the following 14 Strings is returned: “ZOOM_IN_TOOL”,
“ZOOM_OUT_TOOL”, “IDENTIFY_TOOL”, “MEASURE_TOOL”, “PAN_TOOL”,
“PERSISTENT_SELECT_CIRCLE”, “PERSISTENT_SELECT_POLYGON”,
“PERSISTENT_SELECT_POLYLINE”, “EDITNOTES_TOOL”,
“MAPNOTES_TOOL”, “SELECT_RECTANGLE_TOOL”,
“SELECT_CIRCLE_TOOL”, “SELECT_LINE_TOOL”,
“SELECT_POLYGON_TOOL”.

See Also

`IMSMMap.selectTool`

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var selectedTool = imsMap.getSelectedTool();  
if(selectedTool == "ZOOM_IN") {  
    alert("Zoom in tool is active.");  
}
```

IMSMMap.getShowStackTrace

Description

Returns the current level stack trace being displayed in the console if an exception is thrown in the viewer.

Syntax

```
int getShowStackTrace()
```

Arguments

None

Returned Value

`int` Returns the level at which the messages are being displayed. These values are defined in `IMSMMap` as the constants: `SHOW_NO_STACK_TRACES`, `SHOW_RUNTIME_STACK_TRACES`, `SHOW_ALL_STACK_TRACES`.

See Also

`IMSMMap.setShowStackTrace`

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var traceLevel = imsMap.getShowStackTrace();
```

IMSMMap.getStoredQueries

Description

Returns all the stored queries on a given layer as a Collection.

Syntax

Collection getStoredQueries(Layer layer)

Arguments

Layer The layer to search for stored queries on.

Returned Value

Collection Returns a Collection with all the names of the stored queries on this layer as Strings. If none exist, then an empty Collection is returned.

See Also

Collection

Example

See next page

IMSMAP.getStoredQueries

Example

```
function getLayerStoredQueries() {  
    var imsMap = parent.mapFrame.document.IMSMAP;  
    var layer = imsMap.getSelectedLayer();  
    var stoQueries = imsMap.getStoredQueries(layer);  
    var nameList = “”;  
    if(stoQueries.getSize()>0) {  
        for(var i=0; i < stoQueries.getSize(); i++) {  
            nameList = nameList + stoQueries.getStringElement(i);  
            if(i < (numberOfLayers-1)) {  
                nameList = nameList + “, “;  
                if(i == (numberOfLayers-2)) {  
                    nameList = nameList + “and “;  
                }  
            }  
        }  
        alert(“The StoredQueries on this layer are “ + nameList + “.”);  
    } else {  
        alert(“No StoredQueries on this layer.”);  
    }  
}
```

IMSMAP.getSubLayer

Description

Returns the defined sublayer of a given layer as a Layer.

Syntax

Layer getSubLayer(String groupLayer, String subLayer)

Arguments

groupLayer The name of the group layer for the sublayer on the map.
subLayer The name of the sublayer to be returned.

Returned Value

Layer Returns the Layer defined by the parent and child arguments. If the layer does not exist, null is returned.

See Also

[IMSMAP.getSubLayerNames](#)

Example

See next page

IMSMAP.getSubLayer

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;
var selectedLayer = imsMap.getSelectedLayer();
var layerRendererType = imsMap.getLayerRendererType(selectedLayer);
if(layerRendererType == "GROUP_RENDERER"){
    var names = imsMap.getSubLayerNames(selectedLayer);
    var subLayerCount = names.getSize();
    if(subLayerCount > 0) {
        for(int i=0; i < subLayerCount; i++){
            var theSubLayerName = names.getStringElement(i);
            var layer =
imsMap.getSubLayer(selectedLayer.getName(),theSubLayerName);
            ...
        }
    }
}
```

IMSMMap.getSubLayerNames

Description

Returns the names of all the sublayers for a given layer as a Collection.

Syntax

Collection getSubLayerNames(Layer layer)

Arguments

layer The Layer to be searched for sublayers.

Returned Value

Collection Returns a Collection containing the names of all the sublayers of the layer argument. If no such layers exist, then an empty Collection is returned.

See Also

[IMSMMap.getSubLayer](#)
[Collection](#)

Example

See next page

IMSMAP.getSubLayerNames

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;
var selectedLayer = imsMap.getSelectedLayer();
var layerRendererType = imsMap.getLayerRendererType(selectedLayer);
if(layerRendererType == "GROUP_RENDERER"){
    var names = imsMap.getSubLayerNames(selectedLayer);
    var subLayerCount = names.getSize();
    if(subLayerCount > 0) {
        for(int i=0; i < subLayerCount; i++){
            var theSubLayerName = names.getStringElement(i);
            var layer =
imsMap.getSubLayer(selectedLayer.getName(),theSubLayerName);
            ...
        }
    }
}
```

IMSMMap.getToc

Description

Returns the IMSToc associated with this map.

Syntax

IMSToc getToc()

Arguments

None

Returned Value

IMSToc Returns the IMSToc associated with this IMSMap instance. If no IMSToc exists, one is created.

See Also

IMSToc

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var theToc = imsMap.getToc();
var mapBGColor = imsMap.getBGColor();
var rgb = mapBGColor.split(",");
if(rgb.length == 3){
    theToc.setBGColor(rgb[0],rgb[1],rgb[2]);
} else {
    alert("Error in obtaining Background Color of Map");
}
```

IMSMAP.getValueMapDefaultSymbol

Description

Returns the default Symbol used by the ValueMapRenderer passed as the argument.

Syntax

```
Symbol getValueMapDefaultSymbol(ValueMapRenderer vmrenderer)
```

Arguments

vmrenderer The ValueMapRenderer to find the default symbol for.

Returned Value

Symbol Returns the Symbol which is the default symbol for the ValueMapRenderer argument. If error occurs null is returned.

See Also

[IMSMAP.getValueMapDefaultSymbolType](#)
[IMSMAP.getValueMapLabelDefaultSymbol](#)
[IMSMAP.getValueMapLabelDefaultSymbolType](#)
[ValueMapRenderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerRendererType(layer);

if(rendererType == "VALUemap_RENDERER"){
    var renderer = imsMap.getLayerRenderer(layer);
    var symbol = imsMap.getValueMapDefaultSymbol(renderer);
    ...
}
```

IMSMMap.getValueMapDefaultSymbolType

Description

Returns the type of the default symbol used by the ValueMapRenderer passed as the argument.

Syntax

```
Symbol getValueMapDefaultSymbolType(ValueMapRenderer vmrenderer)
```

Arguments

vmrenderer The ValueMapRenderer to find the default symbol type for.

Returned Value

String Returns a String for the symbol defined by the ValueMapRenderer. If no match is found, the String “None” is returned. Otherwise, one the following 13 constants from Constants is returned: “MARKER_SYMBOL”, “LINE_SYMBOL”, “POLYGON_SYMBOL”, “TRUETYPE_MARKER_SYMBOL”, “RASTER_MARKER_SYMBOL”, “HASH_LINE_SYMBOL”, “RASTER_FILL_SYMBOL”, “GRADIENT_FILL_SYMBOL”, “SHIELD_SYMBOL”, “TEXT_SYMBOL”, “RASTER_SHIELD_SYMBOL”, “CALLOUT_MARKER_SYMBOL”, “FILL_SYMBOL”..

See Also

[IMSMMap.getValueMapDefaultSymbol](#)
[IMSMMap.getValueMapLabelDefaultSymbol](#)
[IMSMMap.getValueMapLabelDefaultSymbolType](#)
[ValueMapRenderer](#)

Example

See next page

IMSMAP.getValueMapDefaultSymbolType

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
var layer = imsMap.getSelectedLayer();  
var rendererType = imsMap.getLayerRendererType(layer);  
if(rendererType == "VALUemap_RENDERER"){  
    var renderer = imsMap.getLayerRenderer(layer);  
    var symbolType = imsMap.getValueMapDefaultSymbolType(renderer);  
    if(symbolType == "HASH_LINE_SYMBOL"){  
        ...  
    }  
}
```

IMSMMap.getValueMapLabelDefaultSymbol

Description

Returns the default Symbol used by the ValueMapLabelRenderer passed as the argument.

Syntax

Symbol getValueMapLabelDefaultSymbol(ValueMapLabelRenderer vmrenderer)

Arguments

vmlrenderer The ValueMapLabelRenderer to find the default symbol for.

Returned Value

Symbol Returns the Symbol which is the default symbol for the ValueMapLabelRenderer argument. If error occurs null is returned.

See Also

[IMSMMap.getValueMapDefaultSymbol](#)
[IMSMMap.getValueMapDefaultSymbolType](#)
[IMSMMap.getValueMapLabelDefaultSymbolType](#)
[ValueMapRenderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerLabelRendererType(layer);

if(rendererType == "VALUemap_LABEL_RENDERER"){
    var renderer = imsMap.getLayerLabelRenderer(layer);
    var symbol = imsMap.getValueMapLabelDefaultSymbol(renderer);
    ...
}
```

IMSMMap.getValueMapLabelDefaultSymbolType

Description

Returns the type of the default symbol used by the ValueMapLabelRenderer passed as the argument.

Syntax

Symbol getValueMapLabelDefaultSymbolType(ValueMapLabelRenderer vmlrenderer)

Arguments

vmlrenderer The ValueMapLabelRenderer to find the default symbol type for.

Returned Value

String Returns a String for the symbol defined by the ValueMapLabelRenderer. If no match is found, the String “None” is returned. Otherwise, one the following 13 constants from Constants is returned: “MARKER_SYMBOL”, “LINE_SYMBOL”, “POLYGON_SYMBOL”, “TRUETYPE_MARKER_SYMBOL”, “RASTER_MARKER_SYMBOL”, “HASH_LINE_SYMBOL”, “RASTER_FILL_SYMBOL”, “GRADIENT_FILL_SYMBOL”, “SHIELD_SYMBOL”, TEXT_SYMBOL, “RASTER_SHIELD_SYMBOL”, “CALLOUT_MARKER_SYMBOL”, “FILL_SYMBOL”..

See Also

[IMSMMap.getValueMapDefaultSymbol](#)
[IMSMMap.getValueMapDefaultSymbolType](#)
[IMSMMap.getValueMapLabelDefaultSymbol](#)
[ValueMapRenderer](#)

Example

See next page

IMSSMap.getValueMapLabelDefaultSymbolType

Example

```
var imsMap = parent.mapFrame.document.IMSSMap;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerLabelRendererType(layer);
if(rendererType == "VALUEMAP_LABEL_RENDERER"){
    var renderer = imsMap.getLayerLabelRenderer(layer);
    var symbolType = imsMap.getValueMapLabelDefaultSymbolType(renderer);
    ...
}
```

IMSMMap.getWrappedRenderer

Description

Returns the Renderer that is being wrapped by a ScaleDependentRenderer.

Syntax

```
Renderer getWrappedRenderer(Layer layer, ScaleDependentRenderer scaleRenderer)
```

Arguments

layer The Layer being rendered.

scaleRenderer The ScaleDependentRenderer which is wrapping another Renderer.

Returned Value

Renderer The Renderer for the given layer that was wrapped by a ScaleDependentRenderer. If either argument is invalid, null is returned.

See Also

[IMSMMap.getWrappedRendererType](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerRendererType(layer);

if(rendererType == "SCALE_DEPENDENT_RENDERER"){
    var sdRend = imsMap.getLayerRender(layer);
    var wrapRend = imsMap.getWrappedRenderer(layer, sdRend);
    ...
}
```

IMSMMap.getWrappedRendererType

Description

Returns the type of Renderer that is being wrapped by a ScaleDependentRenderer.

Syntax

```
String getWrappedRendererType(Layer layer, ScaleDependentRenderer scaleRenderer)
```

Arguments

layer The Layer being rendered.

scaleRenderer The ScaleDependentRenderer which is wrapping another Renderer.

Returned Value

String	The type of Renderer being wrapped by the ScaleDependentRenderer. The following are the Renderer types defined as Strings: “SIMPLE_RENDERER”, “VALUemap_RENDERER”, “SCALE_DEPENDENT_RENDERER”, “GROUP_RENDERER”, “LABEL_RENDERER”, “VALUemap_LABEL_RENDERER”.
--------	---

See Also

[IMSMMap.getWrappedRenderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerRendererType(layer);

if(rendererType == "SCALE_DEPENDENT_RENDERER"){
    var sdRend = imsMap.getLayerRender(layer);
    var wrapRendType = imsMap.getWrappedRendererType(layer, sdRend);
    if(wrapRendType == "SIMPLE_RENDERER"){
        ...
    }
}
```

IMSMMap.goBackToExtent

Description

Changes the current extent of the map to the previous extent if one exists in the current session history.

Syntax

boolean goBackToExtent()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.goBackForwardToExtent](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.goBackToExtent();
```

IMSMMap.goForwardToExtent

Description

Changes the current extent of the map to the next extent if one exists in the current session history.

Syntax

boolean goForwardToExtent()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.goBackToExtent](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.goBackToExtent();
```

IMSMMap.isEditNotesModified

Description

Returns true if the edit notes tool is open and a change has been made since the last time the session was submitted.

Syntax

boolean isEditNotesModified()

Arguments

None

Returned Value

boolean True if modified, false otherwise.

See Also

[IMSMMap.selectEditNotesToolFunction](#)
[IMSMMap.selectTool](#)
[IMSMMap.startEditNotesTool](#)
[IMSMMap.stopEditNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
if(imsMap.isEditNotesModified()) {  
    alert("EditNote has been modified.");  
}
```

IMSMMap.isMapNotesEditable

Description

Returns true if the current MapNotes session is editable, false otherwise.

Syntax

boolean isMapNotesEditable()

Arguments

None

Returned Value

boolean True if editable, false otherwise.

See Also

[IMSMMap.addMapNotesLayer](#)
[IMSMMap.selectMapNotesLayer](#)
[IMSMMap.setMapNotesEditable](#)
[IMSMMap.setMapNotesFolder](#)
[IMSMMap.setMapNotesFolderOnServer](#)
[IMSMMap.setMapNotesSubmitLimit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
if(!imsMap.isMapNotesEditable()) {  
    alert("Cannot edit the MapNotes.");  
}
```

IMSMMap.isProjectModified

Description

Returns true if the project has been modified since the last time it was saved, otherwise false is returned.

Syntax

boolean isProjectModified()

Arguments

None

Returned Value

boolean True if project has been modified since the last save, false otherwise.

See Also

[IMSMMap.closeProject](#)
[IMSMMap.loadMapOnlyProject](#)
[IMSMMap.loadProject](#)
[IMSMMap.saveProject](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
if(parent.mapFrame.IMSMMap.isProjectModified()) {
if(confirm("Save project before closing?")) {
    imsMap.displaySaveAsProjectUI();
} else {
    imsMap Map.closeProject();
}
} else {
    imsMap.closeProject();
}
```

IMSMAP.loadMapOnlyProject

Description

Loads a map only project file (AXL file) from a given URL. Only the map information is retained. Information on the overview map, toolbar, toc, and scalebar is not saved.

Syntax

boolean loadMapOnlyProject(String url)

Arguments

url The URL of the map only project to load.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.closeProject](#)

[IMSMAP.loadProject](#)

[IMSMAP.saveProject](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.loadMapOnlyProject("http://headhunter/projects/survival.axl");
```

IMSMAP.loadProject

Description

Loads a project file (AXL file) from a given URL.

Syntax

boolean loadProject(String url)

Arguments

url The URL of the project to load.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.closeProject](#)
[IMSMAP.loadMapOnlyProject](#)
[IMSMAP.saveProject](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.loadProject("http://nebula/projects/gases.axl");
```

IMSMAP.moveSelectedLayer

Description

Moves the selected layer in the direction specified by the direction argument. If there is no selected layer then there is no effect. Note that IMSMAP.redraw() must be called for effects to take place on the map.

Syntax

```
boolean moveSelecteLayer(String direction)
```

Arguments

direction The direction in which to move the currently selected layer. Known values are defined by the following Strings: “MOVE_UP”, “MOVE_DOWN”, “MOVE_TOP”, “MOVE_BOTTOM”.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addLayer](#)
[IMSMAP.removeLayer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.moveSelectedLayer("MOVE_UP");  
imsMap.redraw();
```

IMSMAP.pan

Description

Pans the map to the North, South, East, or West.

Syntax

```
boolean pan(String direction)
```

Arguments

Direction The direction to pan the map. Known values are defined by the following Strings: “PAN_NORTH”, “PAN_EAST”, “PAN_SOUTH”, “PAN_WEST”.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.zoom](#)

[IMSMAP.zoomIn](#)

[IMSMAP.zoomOut](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.pan("PAN_EAST");
```

IMSMAP.redraw

Description

This method is used to redraw the map. This must be called after many of the methods in IMSMap for the effects to be displayed on the map (removeLayer() and removeSelectedLayer() for example).

Syntax

boolean redraw()

Arguments

None

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.removeSelectedLayer();  
imsMap.redraw();
```

IMSMMap.removeAcetateLayer

Description

Removes the acetate layer if it exists. Otherwise, there is no effect.

Syntax

boolean removeAcetateLayer()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.createAcetateLayer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.removeAcetateLayer();
```

IMSMAP.removeLayer

Description

Removes the defined layer from the map if it exists. Otherwise, there is no effect.

Syntax

boolean removeLayer(Layer layer)

Arguments

layer Is the Layer to be removed from the map.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.addLayer](#)
[IMSMAP.removeLayerByIndex](#)
[IMSMAP.removeServiceLayers](#)

Example

```
function removeAndTell(){
    var imsMap = parent.mapFrame.document.IMSMAP;
    var selectedLayer = imsMap.getSelectedLayer();
    imsMap.removeLayer(selectedLayer);
    imsMap.redraw();

    alert("Selected Layer removed.");
}
```

IMSMAP.removeLayerByIndex

Description

Removes the layer at the defined index from the map if it exists. Otherwise, there is no effect.

Syntax

```
boolean removeLayerByIndex(int index)
```

Arguments

index The index of the layer to be removed.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.addLayer](#)
[IMSMAP.removeLayer](#)
[IMSMAP.removeServiceLayers](#)

Example

```
function removeAndTell(index){  
    var imsMap = parent.mapFrame.document.IMSMAP;  
    imsMap.removeLayerByIndex(index);  
    imsMap.redraw();  
  
    alert("Layer at index " + index + " removed.");  
}
```

IMSMMap.removeServiceLayers

Description

Removes all the layers on the map associated with this service.

Syntax

```
boolean removeServiceLayers(String url, String service, int serviceType)
```

Arguments

url	The URL of the service to be removed.
service	The name of the service to be removed.
serviceType	0 for Feature Service, 1 for Image Service.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMMap.addLayer](#)
[IMSMMap.removeLayer](#)
[IMSMMap.removeServiceLayers](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.removeServiceLayers("http://lola/", "RunningWater", 0);
```

IMSMMap.saveMapOnlyProject

Description

Saves the current project to an AXL file defined in the argument. Only the map information is retained. Information on the overview map, toolbar, toc, and scalebar is not saved.

Syntax

boolean saveMapOnlyProject(String path)

Arguments

path The path of the file to be saved.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.loadProject](#)
[IMSMMap.loadMapOnlyProject](#)
[IMSMMap.saveProject](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var DEFAULT_FILE = "D:\\ArcIMS\\Projects\\MapOnly\\default.axl";  
imsMap.saveMapOnlyProject(DEFAULT_FILE);
```

IMSMMap.saveProject

Description

Saves the project to an AXL file.

Syntax

boolean saveProject(String path)

Arguments

path The path of the file to be saved.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.loadProject](#)
[IMSMMap.loadMapOnlyProject](#)
[IMSMMap.saveMapOnlyProject](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var DEFAULT_FILE = "D:\\ArcIMS\\Projects\\projects\\temp.axl";  
imsMap.saveProject(DEFAULT_FILE);
```

IMSMMap.selectEditNotesToolFunction

Description

Select the mode of the EditNotes tool.

Syntax

```
boolean selectEditNotesToolFunction(String mode)
```

Arguments

mode	The EditNotes function to have enabled. Known values are defined by the following Strings: “EDITNOTES_TOOL_SELECT_FEATURES”, “EDITNOTES_TOOL_ADD_FEATURES”, “EDITNOTES_TOOL_MODIFY_FEATURES”, “EDITNOTES_TOOL_DELETE_FEATURES”, “EDITNOTES_TOOL_MODIFY_ATTRIBUTES”, “EDITNOTES_TOOL_SUBMIT, EDITNOTES_TOOL_STOP”, “EDITNOTES_TOOL_UNDO, EDITNOTES_TOOL_REDO”, “EDITNOTES_TOOL_CLEAR_SELECTION”, “EDITNOTES_TOOL_SELECT_TOOL_ENVELOPE_ACTION”, “EDITNOTES_TOOL_SELECT_TOOL_CIRCLE_ACTION”, “EDITNOTES_TOOL_SELECT_TOOL_LINE_ACTION”, “EDITNOTES_TOOL_SELECT_TOOL_POLYGON_ACTION”.
------	--

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.selectEditNotesToolFunction](#)

[IMSMMap.selectTool](#)

[IMSMMap.startEditNotesTool](#)

[IMSMMap.stopEditNotesTool](#)

IMSMAP.selectEditNotesToolFunction

Example

```
function selectENTool(mode){  
    var imsMap = parent.mapFrame.document.IMSMAP;  
    if(mode==0) {  
        imsMap.selectEditNotesToolFunction("EDITNOTES_TOOL_SELECT_FEATURES");  
    } else if (mode==1) {  
        ...  
    }  
}
```

IMSMAP.selectMapNotesLayer

Description

Select an active MapNotes layer.

Syntax

```
boolean selectMapNotesLayer(String name)
```

Arguments

name The name of the MapNotes layer to be selected.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addMapNotesLayer](#)
[IMSMAP.getMapNotesLayer](#)
[IMSMAP.setMapNotesFolder](#)
[IMSMAP.setMapNotesFolderOnServer](#)
[IMSMAP.setMapNotesSubmitLimit](#)
[IMSMAP.startMapNotesTool](#)
[IMSMAP.stopMapNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.selectMapNotesLayer("GarageSaleAt_380NewYork_Redlands");
```

IMSSelectMapNotesToolFunction

Description

Select the mode of the MapNotes tool.

Syntax

```
boolean selectMapNotesToolFunction(String mode)
```

Arguments

mode The MapNotes function to have enabled. Known values are defined by the following

Strings: “MAPNOTES_TOOL_SELECT_LAYER”,
“MAPNOTES_TOOL_NEW”, “MAPNOTES_TOOL MODIFY”,
“MAPNOTES_TOOL_DELETE_LAYER”,
“MAPNOTES_TOOL_DETAILS”, “MAPNOTES_TOOL_SUBMIT”,
“MAPNOTES_TOOL_STOP”,
“MAPNOTES_TOOL_SELECT_ELEMENT”,
“MAPNOTES_TOOL_POINT”, “MAPNOTES_TOOL_LINE”,
“MAPNOTES_TOOL_POLYGON”, “MAPNOTES_TOOL_RECTANGLE”,
“MAPNOTES_TOOL_CIRCLE”, “MAPNOTES_TOOL_TEXT”,
“MAPNOTES_TOOL_IMAGE”, “MAPNOTES_TOOL_FREEHAND”,
“MAPNOTES_TOOL_DELETE_ELEMENT”,
“MAPNOTES_TOOL_MODIFY_ELEMENT”,
“MAPNOTES_TOOL_UNDO”, “MAPNOTES_TOOL_REDO”.

Returned Value

boolean True for success, false otherwise.

IMSMAP.selectMapNotesToolFunction

See Also

[IMSMAP.addMapNotesLayer](#)
[IMSMAP.getMapNotesLayers](#)
[IMSMAP.selectMapNotesLayer](#)
[IMSMAP.setMapNotesFolder](#)
[IMSMAP.setMapNotesFolderOnServer](#)
[IMSMAP.setMapNotesSubmitLimit](#)
[IMSMAP.startMapNotesTool](#)
[IMSMAP.stopMapNotesTool](#)

Example

```
function selectMNTTool(mode){  
    var imsMap = parent.mapFrame.document.IMSMAP;  
    if(mode==0) {  
        imsMap.selectMapNotesToolFunction("MAPNOTES_TOOL_SELECT_FEATURES");  
    } else if(mode==1) {  
        ...  
    }  
}
```

IMSMAP.selectTool

Description

Select a predefined tool as the active tool.

Syntax

```
boolean selectTool(String tool)
```

Arguments

tool	The tool to be made active. Known values are defined by the following Strings: “ZOOM_IN_TOOL”, “ZOOM_OUT_TOOL”, “PAN_TOOL”, “IDENTIFY_TOOL”, “MEASURE_TOOL”, “EDITNOTES_TOOL”, “SELECT_RECTANGLE_TOOL”, “SELECT_CIRCLE_TOOL”, “SELECT_POLYGON_TOOL”, “SELECT_LINE_TOOL”, “MAPNOTES_TOOL”, “PERSISTENT_SELECT_POLYGON”, “PERSISTENT_SELECT_POLYLINE”, “PERSISTENT_SELECT_CIRCLE”.
------	--

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.selectEditNotesToolFunction](#)
[IMSMAP.selectMapNotesToolFunction](#)

Example

```
function selectTool(mode){  
    var imsMap = parent.mapFrame.document.IMSMAP;  
    switch(mode) {  
        case 0:  
            imsMap.selectTool("ZOOM_IN");  
            break  
        case1:  
            ...  
    }  
}
```

IMSMMap.setBGColor

Description

Sets the background color using RGB values.

Syntax

boolean setBGColor(int r, int g, int b)

Arguments

r,g,b Values for red, green, and blue using standard RGB form (all values must be between 0-255).

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.getBGColor](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
var RED = 112;  
var GREEN = 155;  
var BLUE = 30;  
imsMap.setBGColor(RED, GREEN, BLUE);
```

IMSMMap.setDefaultProjectDir

Description

Set the default directory to be used for opening and closing of a project.

Syntax

```
boolean setDefaultProjectDir(String path)
```

Arguments

path The path to the directory that is to be used as a project directory.

Returned Value

boolean True for success, false otherwise.

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.setDefaultProjectDir("d:/myAXLs");
```

IMSMMap.setEditNotesDescription

Description

Submits the current EditNotes Session setting the EditNotes description to the argument passed to it. This does not stop the EditNotes session, but submits the session in its current state.

Syntax

```
boolean setEditNotesDescription(String description)
```

Arguments

description A description of the EditNotes session being submitted.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.selectEditNotesToolFunction](#)
[IMSMMap.selectTool](#)
[IMSMMap.startEditNotesTool](#)
[IMSMMap.stopEditNotesTool](#)
[IMSMMap.setEditNotesFolder](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.setEditNotesDescription("Florida after 10% polar cap melt.");
```

IMSMMap.setEditNotesFolder

Description

Sets the EditNotes folder for this session. The folder is a virtual folder on an ArcIMS Server (there is no directory created for it). An EditNotes folder can be managed through ArcIMS Administrator.

Syntax

```
boolean setEditNotesFolder(String server, String name)
```

Arguments

server The name of the web server to connect to.

name The name of the folder to which EditNotes sessions will be submitted.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMMap.selectEditNotesToolFunction](#)

[IMSMMap.selectTool](#)

[IMSMMap.startEditNotesTool](#)

[IMSMMap.stopEditNotesTool](#)

[IMSMMap.setEditNotesDescription](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.setEditNotesFolder("http://mozart/", "Edits");
```

IMSMAP.setExtent

Description

Sets the current Extent of the map.

Syntax

boolean setExtent(Extent extent)

Arguments

extent The Extent the map is to be set to.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.getExtent](#)
[IMSMAP.getFullExtent](#)
[IMSMAP.setExtentLimit](#)
[Extent](#)

Example

```
var MINWIDTH = 10;  
var MINHEIGHT = 10;  
var imsMap = parent.mapFrame.document.IMSMAP;  
var currentExtent = imsMap.getExtent();  
var width = currentExtent.getXMax() - currentExtent.getXMin();  
var height = currentExtent.getYMax() - currentExtent.getYMin();  
if((width < MINWIDTH) || (height < MINHEIGHT)){  
    var newExtent = imsMap.createExtent(currentExtent.getXMin(), currentExtent.getYMin(),  
        MINWIDTH, MINHEIGHT);  
    imsMap.setExtent(newExtent);  
}
```

IMSMMap.setExtentLimit

Description

Sets the extent limit of the map.

Syntax

```
boolean setExtentLimit(Extent extentLimit)
```

Arguments

extentLimit The Extent to be used as the extent limit for this map.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.getExtent](#)
[IMSMMap.getFullExtent](#)
[IMSMMap.setExtent](#)
[Extent](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;
var currentExtent = imsMap.getExtent();
var width = currentExtent.getXMax() - currentExtent.getXMin();
var height = currentExtent.getYMax() - currentExtent.getYMin();
var newExtent = imsMap.createExtent(currentExtent.getXMin(), currentExtent.getYMin(), width,
height);
imsMap.setExtentLimit(newExtent);
```

IMSMAP.setLayerLabelRenderer

Description

Defines the label renderer for a given layer.

Syntax

```
boolean setLayerLabelRenderer(Layer layer, Renderer renderer)
```

Arguments

layer Is the Layer to have the labels rendered on.

renderer Is the Renderer doing the label rendering on the given layer.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.createRenderer](#)

[IMSMAP.setLayerRenderer](#)

[LabelRenderer](#)

[Layer](#)

[Renderer](#)

[ValueMapLabelRenderer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
var layer = imsMap.getSelectedLayer();  
var labelRenderer = imsMap.createRenderer("LABEL_RENDERER");  
labelRenderer.setSeparator("##");  
imsMap.setLayerLabelRenderer(layer, labelRenderer);
```

IMSMMap.setLayerRenderer

Description

Defines the renderer for a given layer.

Syntax

boolean setLayerRenderer(Layer layer, Renderer renderer)

Arguments

layer Is the Layer to be rendered.
renderer Is the Renderer doing the layer rendering.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.setLayerLabelRenderer](#)
[Layer](#)
[Renderer](#)

Example

```
var imsMap= parent.mapFrame.document.IMSMMap;
var layer = imsMap.getSelectedLayer();
var renderer = imsMap.createRenderer("SIMPLE_RENDERER");
var symbol = imsMap.createSymbol("MARKER_SYMBOL");
var newColor = imsMap.createColor(0,255,255);
symbol.setColor(newColor);
symbol.setSize(9);
symbol.setStyle(1);
renderer.setSymbol(symbol);
imsMap.setLayerRenderer(layer, renderer);
imsMap.redraw();
```

IMSMMap.setMapNotesEditable

Description

Enables and disables the ability for the user to edit the current MapNotes session. This does not change the editable property of the MapNotes folder. If the MapNotes folder is not editable, then this call will not allow it to be edited.

Syntax

boolean setMapNotesEditable(boolean editable)

Arguments

Editable True sets the MapNotes session editable, false disables the editing tools of MapNotes.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.isMapNotesEditable](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.setMapNotesEditable(true);
```

IMSMAP.setMapNotesFolder

Description

Sets the MapNotes folder by name.

Syntax

boolean setMapNotesFolder(String name)

Arguments

name Is the name of the MapNotes folder to be used for the current session.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addMapNotesLayer](#)
[IMSMAP.getMapNotesLayer](#)
[IMSMAP.selectMapNotesLayer](#)
[IMSMAP.setMapNotesToolFunction](#)
[IMSMAP.setMapNotesFolderOnServer](#)
[IMSMAP.setMapNotesSubmitLimit](#)
[IMSMAP.startMapNotesTool](#)
[IMSMAP.stopMapNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.setMapNotesFolder("myMapNotes");
```

IMSMAP.setMapNotesFolderOnServer

Description

Sets the MapNotes folder on the server, for this session.

Syntax

boolean setMapNotesFolderOnServer(String server, String name)

Arguments

server The name of the web server to connect to.

name The name of the folder to which MapNotes sessions will be submitted.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addMapNotesLayer](#)

[IMSMAP.getMapNotesLayer](#)

[IMSMAP.selectMapNotesLayer](#)

[IMSMAP.setMapNotesToolFunction](#)

[IMSMAP.setMapNotesFolder](#)

[IMSMAP.setMapNotesSubmitLimit](#)

[IMSMAP.startMapNotesTool](#)

[IMSMAP.stopMapNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.setMapNotesFolderOnServer("http://escher/", "UFO_Sightings");
```

IMSMAP.setMapNotesSubmitLimit

Description

Sets the maximum submission size for images on a MapNote in kilobytes. This does not take into account the submission size attributed to non image edits to a MapNote.

Syntax

```
void    setMapNoteSubmitLimit(int kilobytes)
```

Arguments

kilobytes The maximum number of kilobytes attributed to images per submission.

Returned Value

void

See Also

[IMSMAP.addMapNotesLayer](#)
[IMSMAP.getMapNotesLayer](#)
[IMSMAP.selectMapNotesLayer](#)
[IMSMAP.setMapNotesToolFunction](#)
[IMSMAP.setMapNotesFolder](#)
[IMSMAP.setMapNotesFolderOnServer](#)
[IMSMAP.startMapNotesTool](#)
[IMSMAP.stopMapNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
var SUBMIT_LIMIT = 20; //in kilobytes  
imsMap.setMapNotesSubmitLimit(SUBMIT_LIMIT);
```

IMSMAP.setMapUnit

Description

Sets the map unit.

Syntax

boolean setMapUnit(String unit)

Arguments

unit	The following integer Strings are all the known values with their corresponding units (invalid values are ignored):
Decimal Degrees	“0”
Centimeters	“1”
Inches	“2”
Feet	“3”
Miles	“4”
Meters	“5”
Kilometers	“6”
Unknown	“7”

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.getMapUnit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.setMapUnit("5");
```

IMSMAP.setMeasureUnit

Description

Sets the measure unit for the map.

Syntax

boolean setMeasureUnit(String unit)

Arguments

unit	The following integer Strings are all the known values with their corresponding units (invalid values are ignored):
Feet	“3”
Miles	“4”
Meters	“5”
Kilometers	“6”
Unknown	“7”

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.getMeasureUnit](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.setMeasureUnit("3");
```

IMSMMap.setSelectedLayer

Description

Sets the selected / active map layer.

Syntax

```
boolean setSelectedLayer(String layerName)
```

Arguments

layerName The name of the layer to be selected / made active. Null to deselect all.

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.getSelectedLayer](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.setSelectedLayer("redlands_pools");
```

IMSMMap.setShowStackTrace

Description

Sets the current level stack trace to be outputted to the console.

Syntax

```
void setShowStackTrace(int level)
```

Arguments

level	The level at which the messages are being displayed. These values are defined in IMSMap as the constants: SHOW_NO_STACK_TRACES, SHOW_RUNTIME_STACK_TRACES, SHOW_ALL_STACK_TRACES. The default level is SHOW_RUNTIME_STACK_TRACES.
-------	---

Returned Value

void

See Also

[IMSMMap.getShowStackTrace](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.setShowStackTrace(imsMap.SHOW_NO_STACK_TRACES);
```

IMSMMap.startEditNotesTool

Description

Starts the EditNotes tool and does any of the necessary initializations.

Syntax

```
boolean startEditNotesTool()
```

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMMap.selectEditNotesToolFunction](#)
[IMSMMap.selectTool](#)
[IMSMMap.setEditNotesFolder](#)
[IMSMMap.stopEditNotesTool](#)
[IMSMMap.setEditNotesDescription](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.startEditNotesTool();
```

IMSMAP.startMapNotesTool

Description

Starts the MapNotes tool and does any of the necessary initializations.

Syntax

boolean startMapNotesTool()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addMapNotesLayer](#)
[IMSMAP.getMapNotesLayers](#)
[IMSMAP.selectMapNotesToolFunction](#)
[IMSMAP.setMapNotesFolder](#)
[IMSMAP.setMapNotesFolderOnServer](#)
[IMSMAP.setMapNotesSubmitLimit](#)
[IMSMAP.stopMapNotesTool](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.startMapNotesTool();
```

IMSMAP.stopEditNotesTool

Description

Stops the EditNotes tool and does any of the necessary cleanup then selects the zoom in tool.

Syntax

boolean stopEditNotesTool()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.selectEditNotesToolFunction](#)
[IMSMAP.selectTool](#)
[IMSMAP.setEditNotesFolder](#)
[IMSMAP.startEditNotesTool](#)
[IMSMAP.setEditNotesDescription](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.stopEditNotesTool();
```

IMSMAP.stopMapNotesTool

Description

Stops the MapNotes tool and does any of the necessary cleanup.

Syntax

boolean stopMapNotesTool()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[IMSMAP.addMapNotesLayer](#)
[IMSMAP.getMapNotesLayers](#)
[IMSMAP.selectMapNotesToolFunction](#)
[IMSMAP.setMapNotesFolder](#)
[IMSMAP.setMapNotesFolderOnServer](#)
[IMSMAP.setMapNotesSubmitLimit](#)
[IMSMAP.startMapNotesTool](#)

Example

```
Var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.stopMapNotesTool();
```

IMSMAP.ZOOM

Description

Zooms in or out by a factor of the parameter.

Syntax

boolean zoom(double factor)

Arguments

factor Factor to zoom in or out by. If zoom factor is greater than 1, this method zooms out. If zoom factor is less than 1, this method zooms in. If zoom factor equal to 1, there is no change. If zoom factor is equal to or less than 0, there is no effect.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.ZOOMIN](#)
[IMSMAP.ZOOMOUT](#)
[IMSMAP.ZOOMTOCARSCALE](#)
[IMSMAP.ZOOMTOSCALE](#)
[IMSMAP.ZOOMToselectedlayer](#)
[IMSMAP.ZOOMTOSELECTION](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.zoom(1.6180339887);
```

IMSMMap.zoomIn

Description

Zooms in by a factor of the parameter. Same effect as zoom(1.0/factor).

Syntax

boolean zoomIn(int factor)

Arguments

factor Factor to zoom in by.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMMap.zoom](#)
[IMSMMap.zoomOut](#)
[IMSMMap.zoomToCartScale](#)
[IMSMMap.zoomToScale](#)
[IMSMMap.zoomToSelectedLayer](#)
[IMSMMap.zoomToSelection](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMMap;  
imsMap.zoomIn(3);
```

IMSMAP.zoomOut

Description

Zooms out by a factor of the parameter. Same effect as zoom(factor).

Syntax

boolean zoomOut(int factor)

Arguments

factor Factor to zoom out by.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.zoom](#)
[IMSMAP.zoomIn](#)
[IMSMAP.zoomToCartScale](#)
[IMSMAP.zoomToScale](#)
[IMSMAP.zoomToSelectedLayer](#)
[IMSMAP.zoomToSelection](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.zoomOut(5);
```

IMSMAP.zoomToCartScale

Description

Zoom to the given scale where the scale is a cartographic scale defined by the RF (Representative Fraction).

Syntax

boolean zoomToCartScale(int factor)

Arguments

factor Factor to zoom by.

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.zoom](#)
[IMSMAP.zoomIn](#)
[IMSMAP.zoomOut](#)
[IMSMAP.zoomToScale](#)
[IMSMAP.zoomToSelectedLayer](#)
[IMSMAP.zoomToSelection](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.zoomToCartScale(12000);
```

IMSMAP.zoomToScale

Description

Zoom to scale where scale is map units per pixel.

Syntax

boolean zoomToScale(double scale)

Arguments

scale Scale to zoom to

Returned Value

boolean True for success, false for failure

See Also

[IMSMAP.zoom](#)
[IMSMAP.zoomIn](#)
[IMSMAP.zoomOut](#)
[IMSMAP.zoomToCartScale](#)
[IMSMAP.zoomToSelectedLayer](#)
[IMSMAP.zoomToSelection](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.zoomToScale(1.570796);
```

IMSMAP.zoomToSelectedLayer

Description

Zooms to the selected layer. If no layer is selected, then there is no effect.

Syntax

boolean zoomToSelectedLayer()

Arguments

None

Returned Value

boolean True for success, false for failure.

See Also

[IMSMAP.zoom](#)
[IMSMAP.zoomIn](#)
[IMSMAP.zoomOut](#)
[IMSMAP.zoomToCartScale](#)
[IMSMAP.zoomToScale](#)
[IMSMAP.zoomToSelection](#)

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.zoomToSelectedLayer();
```

IMSMAP.zoomToSelection

Description

Highlights and zooms to an extent including all of the selected features.

Syntax

boolean zoomToSelection(String layer, String query)

Arguments

Layer The name of the layer the search is being done on.

Query The query string to search on.

Returned Value

boolean True for success, false if no values are returned or if an error occurred.

See Also

IMSMAP.zoom

IMSMAP.zoomIn

IMSMAP.zoomOut

IMSMAP.zoomToCartScale

IMSMAP.zoomToScale

IMSMAP.zoomToSelectedLayer

Example

```
var imsMap = parent.mapFrame.document.IMSMAP;  
imsMap.zoomToSelection("Cities","POPULATION > 9000000");
```

IMSOerviewMap

Description

The IMSOverviewMap class provides methods to customize the overview map applet. It extends javax.swing.Japplet class. The size of the overview map window is defined by APPLET_DIM Constant inside the class as the following:

```
public static java.awt.Dimension APPLET_DIM= new java.awt.Dimension(120,120);
```

This class displays one or more layers on a map at a larger extent compared with the map. You can choose which map layers to show in the IMSOverviewMap by using one of the IMSOverviewMap's addLayer methods. These layers must already be in the map. If you remove them from the map, they are automatically removed from the IMSOverviewMap. However you can still remove particular layers from the IMSOverviewMap by calling removeLayer method. The current extent of the map is normally shown as a half transparent red colored rectangular frame. You can change the color using the setRectangleFillColor and setRectangleOutlineColor methods. The setBGCOLOR method allows you to change the background color of the IMSOverview window. Before you can use an IMSOverviewMap object, you need to call the setMap method to associate it with an IMSMap object. Usually it is performed by the environment when an ArcIMS Java Custom Applet is initialized.

A new instance of the IMSOverviewMap can be created by calling IMSMap.getOverviewMap().

See Also

[IMSMap](#)

IMSOerviewMap.addLayer

Description

Adds a layer to the IMSOverviewMap layer set. The picture changes in the window after the IMSOverviewMap.redraw() method has been executed. If the layer is already in the set, then nothing is changed.

Syntax

```
boolean addLayer(String layerName);
```

Arguments

layerName the name of the layer being added

Returned Value

true if the method was successful, false otherwise

See Also

[IMSOerviewMap.addMapLayer](#)
[IMSOerviewMap.removeLayer](#)
[IMSOerviewMap.layerExists](#)

Example

```
var layerNames = mapFrame.IMSMap.getLayerNames();
var layerName = layerNames.get(1);
alert(overviewFrame.IMSOverviewMap.addLayer(layerName));
alert(overviewFrame.IMSOverviewMap.redraw());
```

IMSOerviewMap.addMapLayer

Description

Adds a layer to the IMSOverviewMap layer set. The picture changes in the window after the IMSOverviewMap.redraw() method has been executed. If the layer is already in the set, then nothing is changed.

Syntax

```
boolean addLayer(Layer layer);
```

Arguments

layer a layer being added

Returned Value

true if the method was successful, false otherwise

See Also

[IMSOerviewMap.addLayer](#)
[IMSOerviewMap.removeLayer](#)
[IMSOerviewMap.layerExists](#)

Example

```
var layerNames = mapFrame.IMSMap.getLayerNames();
var layerName = layerNames.get(1);
var layer = mapFrame.IMSMap.getLayer(layerName);
alert(overviewFrame.IMSOverviewMap.addMapLayer(layer));
alert(overviewFrame.IMSOverviewMap.redraw());
```

IMSOerviewMap.getBGColor

Description

Returns the current background color of the IMSOverviewMap window.

Syntax

String getBGColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[IMSOerviewMap.setBGColor](#)

Example

```
alert("the current background color of the OverviewMap is "+  
overviewFrame.IMSOerviewMap.getBGColor());
```

IMSOerviewMap.getRectangleFillColor

Description

Returns the current fill color of the colored rectangular frame on the IMSOverviewMap window.

Syntax

`String getRectangleFillColor()`

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[IMSOerviewMap.setRectangleFillColor](#)

Example

```
alert("the current fill color of the rectangular frame is " +  
overviewFrame.IMSOerviewMap.getBGColor());
```

IMSOerviewMap.getRectangleOutlineColor

Description

Returns the current outline color of the colored rectangular frame on the IMSOverviewMap window.

Syntax

```
String getRectangleOutlineColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[IMSOerviewMap.setRectangleOutlineColor](#)

Example

```
alert(" the current outline color of the rectangular frame is " +  
overviewFrame.IMSOerviewMap.getBGColor());
```

IMSOerviewMap.layerExists

Description

Checks to see if a particular layer is in the IMSOverviewMap layer set.

Syntax

boolean layerExists(Layer layer)

Arguments

layer a layer being checked

Returned Value

true if the layer is in the set , false otherwise

See Also

[IMSOerviewMap.addMapLayer](#)
[IMSOerviewMap.removeLayer](#)
[IMSOerviewMap.addLayer](#)

Example

```
var layerNames = mapFrame.IMSMap.getLayerNames();
var layerName = layerNames.get(1);
var layer = mapFrame.IMSMap.getLayer(layerName);
if(overviewFrame.IMSOverviewMap.layerExists(layer))
    alert ("The layer of " +layerName+ " is in the layer set");
else
    alert ("The layer of " +layerName+ " is not in the layer set");
```

IMSOerviewMap.redraw

Description

Redraws the IMSOverviewMap applet window. Any changes made to the IMSOverviewMap will be reflected.

Syntax

```
boolean redraw();
```

Arguments

None

Returned Value

true if the method was successful, false otherwise

See Also

[IMSOerviewMap](#)

Example

```
alert(overviewFrame.IMSOerviewMap.redraw());
```

IMSOerviewMap.removeLayer

Description:

Removes a layer from the IMSOverviewMap layer set. The picture changes in the IMSOverviewMap window after the IMSOverviewMap.redraw() method has been executed.

Syntax:

```
boolean removeLayer(Layer layer);
```

Arguments:

layer a layer being removed

Returned Value:

true if the method was successful, false otherwise

See Also

[IMSOerviewMap.addMapLayer](#)

[IMSOerviewMap.addLayer](#)

[IMSOerviewMap.layerExists](#)

Example

see the next page

IMSOerviewMap.removeLayer

Example

```
var layerNames = mapFrame.IMSMap.getLayerNames();
var layerName = layerNames.get(1);
var layer = mapFrame.IMSMap.getLayer(layerName);
if(overviewFrame.IMSOerviewMap.layerExists(layer))
    alert ("The layer of " +layerName+ " is in the layer set");
else
    alert ("The layer of " +layerName+ " is not in the layer set");
alert(overviewFrame.IMSOerviewMap.removeLayer(layer));
if(overviewFrame.IMSOerviewMap.layerExists(layer))
    alert ("The layer of " +layerName+ " is in the layer set");
else
    alert ("The layer of " +layerName+ " is not in the layer set");
alert(overviewFrame.IMSOerviewMap.redraw());
```

IMSOerviewMap.setBColor

Description

Sets the background color of the IMSOverviewMap window. The color is changed in the window after the IMSOverviewMap.redraw() method has been executed.

Syntax

boolean setBColor(int R, int G, int B)

Arguments

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue part of the RGB color value, should be between 0 and 255

Returned Value

true if the method was successful, false otherwise

See Also

[IMSOerviewMap.getBColor](#)
[IMSOerviewMap.redraw](#)

Example

```
alert("set the white color as the background IMSOverviewMap color");  
overviewFrame.IMSOerviewMap.setBColor(255,255,255);  
alert(" the current background color of the IMSOverviewMap is "+  
overviewFrame.IMSOerviewMap.getBColor());  
overviewFrame.IMSOerviewMap.redraw();
```

IMSOerviewMap.setMap

Description

Sets an IMSMap to be used with the IMSOverviewMap applet. Normally, you will not use this method as it is called when the applet is initialized.

Syntax

```
boolean setMap( IMSMap imsMap);
```

Arguments

imsMap an instance of the IMSMap class

Returned Value

true if the method was successful, false otherwise

Example

```
alert("reset the same IMSMap object that is already used");
alert(overviewFrame.IMSOerviewMap.setMap(mapFrame.IMSMap));
overviewFrame.IMSOerviewMap.redraw();
```

IMSOerviewMap.setRectangleFillColor

Description

Sets a fill color of the colored rectangular frame on the IMSOverviewMap window. The color is changed in the window after the IMSOverviewMap.redraw() method has been executed.

Syntax

```
boolean setRectangleFillColor( int R, int G, int B, int A)
```

Arguments

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue part of the RGB color value, should be between 0 and 255
- A defines the alpha part of the RGB color value, should be between 0 and 255

Returned Value

true if the method was successful, false otherwise

See Also

[IMSOerviewMap.getRectangleFillColor](#)

Example

see the next page

IMSOerviewMap.setRectangleFillColor

Example

```
alert("set the white color as the fill color of the rectangular frame");
overviewFrame.IMSOerviewMap.setRectangleFillColor(255,255,255,0);
alert("the current rectangle fill color is " +
overviewFrame.IMSOerviewMap.getRectangleFillColor());
overviewFrame.IMSOerviewMap.redraw();
```

IMSOerviewMap.setRectangleOutlineColor

Description

Sets an outline color of the colored rectangular frame on the IMSOverviewMap window. The color is changed in the window after the IMSOverviewMap.redraw() method having been executed

Syntax

```
boolean setRectangleOutlineColor( int R, int G, int B, int A)
```

Arguments

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue part of the RGB color value, should be between 0 and 255
- A defines the alpha part of the RGB color value, should be between 0 and 255

Returned Value

true if the method was successful, false otherwise

See Also

[IMSOerviewMap.getRectangleOutlineColor](#)

Example

see the next page

IMSOerviewMap.setRectangleOutlineColor

Example

```
alert("set the white color as the outline color of the rectangular frame");
overviewFrame.IMSOerviewMap.setRectangleOutlineColor(255,255,255,0);
alert("the current rectangle outline color is "+
overviewFrame.IMSOerviewMap.getRectangleOutlineColor());
overviewFrame.IMSOerviewMap.redraw();
```

IMSScaleBar

The IMSScaleBar provides methods to customize a scale bar panel. It extends javax.swing.Japplet. The scale bar panel is used to display information about the current scale used by the IMSMap applet. It is placed below the IMSMap window on the ArcIMS Java Custom Applet page. The environment creates a IMSScaleBar instance when an ArcIMS Java Custom Applet is initialized. You can get access to the instance through the IMSMap.getScaleBar() method.

See Also

[IMSMap](#)

IMSScaleBar.getBGColor

Description

Returns the current background color of the IMSScalebar panel.

Syntax

String getBGColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines the red part of the RGB color value, should be between 0 and 255

G defines the green part of the RGB color value, should be between 0 and 255

B defines the blue part of the RGB color value, should be between 0 and 255

See Also

[IMSScaleBar.setBGColor](#)

Example

```
alert(" the current background color of the IMSScaleBar panel is " +  
scalebarFrame.IMSScaleBar.getBGColor());
```

IMSScaleBar.getFGColor

Description

Returns the current foreground color of the IMSScalebar panel.

Syntax

`String getFGColor()`

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue part of the RGB color value, should be between 0 and 255

See Also

[IMSScaleBar.setFGColor](#)

Example

```
alert(" the current foreground color of the IMSScaleBar panel is " +  
scalebarFrame.IMSScaleBar.getFGColor());
```

IMSScaleBar.getScaleUnit

Description

Returns the current scale unit used by the IMSScaleBar panel.

Syntax

`int getScaleUnit()`

Arguments

None

Returned Value

- 3 - Feet
- 4 - Miles
- 5 - Meters
- 6 - Kilometers

See Also

[IMSScaleBar.getScaleUnit](#)

Example

```
alert("the current scale unit of the IMSScaleBar panel is "+  
scalebarFrame.IMSScaleBar.getScaleUnit());
```

IMSScaleBar.getScreenUnit

Description

Returns the current screen unit used by the IMSScaleBar panel.

Syntax

```
int getScreenUnit()
```

Arguments

None

Returned Value

1 – centimeters
2 - inches

See Also

[IMSScaleBar.setScreenUnit](#)

Example

```
alert(" the current screen unit of the IMSScaleBar panel is "+  
scalebarFrame.IMSScaleBar.getScreenUnit());
```

IMSScaleBar.refresh

Description

Refreshes the picture on the IMSScaleBar panel.

Syntax

```
boolean refresh();
```

Arguments

None

Returned Value

true if the method was successful, false otherwise

See Also

IMSScaleBar

Example

```
alert(scalebarFrame.IMSScaleBar.refresh());
```

IMSScaleBar.setBColor

Description

Sets the current background color of the IMSScalebar panel.

Syntax

boolean setBColor(int R, int G, int B)

Arguments

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue of the RGB color value, should be between 0 and 255

Returned Value

true if the method was successful, false otherwise

See Also

[IMSScaleBar.getBColor](#)

Example

```
alert(" set the background color to white");
scalebarFrame.IMSScaleBar.setBColor(255,255,255));
alert(" the current background color is " + scalebarFrame.IMSScaleBar.getBColor());
```

IMSScaleBar.setFGColor

Description

Sets the current foreground color of the IMSScalebar panel

Syntax

boolean setFGColor(int R, int G, int B)

Arguments

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue part of the RGB color value, should be between 0 and 255

Returned Value

true if the method is successful, false otherwise

See Also

[IMSScaleBar.getFGColor](#)

Example

```
alert(" set the foreground color to white");
scalebarFrame.IMSScaleBar.setFGColor(255,255,255));
alert(" the current foreground color is " + scalebarFrame.IMSScaleBar.getFGColor());
```

IMSScaleBar.setScaleUnit

Description

Sets the current scale unit used by the IMSScaleBar panel.

Syntax

```
boolean setScaleUnit(String scaleUnit)
```

Arguments

scaleUnit: “FEET”, “METERS”, “KILOMETERS”, “MILES”

Returned Value

true if the method was successful, false otherwise

See Also

[IMSScaleBar.getScaleUnit\(\)](#)

Example

```
alert(scalebarFrame.IMSScaleBar.setScaleUnit("METERS"));
```

IMSScaleBar.setScreenUnit

Description

Gets screen unit setting used by the IMSScaleBar.

Syntax

```
boolean setScreenUnit(String screenUnit)
```

Arguments

screenUnit: “INCHES”, “CENTIMETERS”

Returned Value

true if the method was successful, false otherwise

See Also

IMSScaleBar.setScreenUnit

Example

```
alert(scalebarFrame.IMSScaleBar.setScreenUnit("CENTIMETERS"));
```

IMSToc

The IMSToc applet extends javax.swing.Japplet and contains methods to work with a Table of contents (Toc) that is a visual representation of the layer set of a map. Each layer is represented by a separate legend contained within the Toc. Legends can be “selected” by clicking on them. A layer becomes the active layer if its legend is selected. You can customize Toc by setting its background color and some other features.

To make the applet work automatically with an IMSMap instance on the same document page you must:

- set a link to the IMSMap by using the IMSToc.setMap method
- register it using the setExternalToc method in IMSMap.

Usually the procedure is performed by the ArcIMS Java custom applet environment when the applet is initialized. The current instance of the IMSToc can be accessed through the IMSMap.getToc() method.

See Also

[IMSMap](#)
[Layer](#)

IMSToc.getBGColor

Description

Returns the current background color of the IMSToc panel.

Syntax

String getBGColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines the red part of the RGB color value, should be between 0 and 255

G defines the green part of the RGB color value, should be between 0 and 255

B defines the blue part of the RGB color value, should be between 0 and 255

See Also

[IMSToc.setBGColor](#)

Example

```
alert("the current background color of the ToC panel is "+  
tocFrame.IMSToc.getBGColor());
```

IMSToc.getSelectedLayer

Description

Returns the layer selected on the IMSToc panel.

Syntax

Layer getSelectedLayer()

Arguments

None

Returned Value

the layer selected or null if no layer selected.

Example

```
var layer = tocFrame.IMSToc.getSelectedLayer();
if( layer == null)
    alert(" no layer selected");
else
    alert(" the selected layer is " +layer.getName());
```

IMSToc.isDrawBorderEnabled

Description

Checks if the border is drawn around the ToC panel.

Syntax

boolean isDrawBorderEnabled()

Arguments

None

Returned Value

true if the ToC border is drawn, false otherwise

See Also

[IMSToc.setDrawBorderEnabled](#)

Example

```
var IMSToc = mapFrame.IMSMap.getToc();
if( true== IMSToc.isDrawBorderEnabled())
    alert(" the ToC border is drawn");
else
    alert(" the ToC border is not drawn");
```

IMSToc.isLayerExpanded

Description

Checks whether a layer has a classification legend

Syntax

boolean isLayerExpanded(Layer layer)

Arguments

layer is the layer being checked

Returned Value

True if layer is expanded, False, otherwise

See Also

[IMSToc.setLayerExpanded](#)

Example

```
var IMSToc = mapFrame.IMSMap.getToc();
var layer = mapFrame.IMSMap.getLayer("County");
if( true == IMSToc.isLayerExpanded(layer))
    alert (" The layer of County is having a classification legend");
else
    alert (" The layer of County is not having a classification legend");
```

IMSToc.refresh

Description

Refreshes the IMSToc applet panel.

Syntax

```
boolean refresh();
```

Arguments

None

Returned Value

true if the method was successful, false otherwise

Example

```
alert(tocFrame.IMSToc.refresh());
```

IMSToc.setBGColor

Description:

Sets the background color of the IMSToc panel.

Syntax:

boolean setBGColor(int R, int G, int B)

Arguments:

- R defines the red part of the RGB color value, should be between 0 and 255
- G defines the green part of the RGB color value, should be between 0 and 255
- B defines the blue part of the RGB color value, should be between 0 and 255

Returned Value:

true if the method was successful, false otherwise

See Also

[IMSToc.getBGColor](#)

Example

```
alert(" set the background color of the IMSToc panel to white");  
alert(tocFrame.IMSToc.setBGColor(255,255,255));
```

IMSToc.setDrawBorderEnabled

Description

Controls whether the Toc border is drawn. By default, it is not drawn.

Syntax

```
boolean setDrawBorderEnabled( String enabled)
```

Arguments

enabled “true” is for TOC border being drawn, any other string is for TOC border not being drawn

Returned Value

true if the method was successful, false otherwise

See Also

[IMSToc.isDrawBorderEnabled](#)

Example

```
alert(" force the border to be drawn");
var IMSToc = mapFrame.IMSMap.getToc();
alert(IMSToc.setDrawBorderEnabled("true"));
```

IMSToc.setLayerExpanded

Description

Specifies whether a layer classification legend is displayed. It is displayed by default.

Syntax

```
boolean setLayerExpanded( Layer layer, String expanded)
```

Arguments

Layer is the name of the specified layer

Expanded : “true”, if the layer classification legend is displayed, “false”, if not.

Returned Value

true if the method was successful, false otherwise

Example

```
alert(" force that the classification legend not to be displayed for the county layer");
var IMSToc = mapFrame.IMSMap.getToc();
var layer = mapFrame.IMSMap.getLayer("County");
alert(IMSToc.setLayerExpanded(layer, "false"));
```

IMSToc.setMap

Description

Associates the IMSToc with a specified IMSMap.

Syntax

```
boolean setMap(IMSMap imsMap);
```

Arguments

imsMap is an IMSMap applet

Returned Value

true if the method was successful, false otherwise

See Also

IMSMap

Example

```
alert(tocFrame.IMSToc.setMap(mapFrame.IMSMap));
```

IMSToc.updateLegend

Description

Updates the legend of a particular layer. You need to invoke this method if you change any aspect of the layer outside the context of the Toc. For instance if you manually set the Visible property of the layer, then want the Toc to reflect this change, or, more importantly, if you change any aspect of the layer renderer.

Syntax

```
boolean updateLegend( Layer layer)
```

Arguments

layer - the layer being chosen

Returned Value

true if the method was successful, false otherwise

See Also

[Layer](#)

Example

```
var layer = mapFrame.IMSMap.getLayer("County");
alert( layer.setVisible("false"));
alert(tocFrame.IMSToc.updateLegend(layer));
```

IMSToolBar

Description

The IMSToolBar applet is used as a container for tools available to the end user. Although the IMSToolBar is used by IMSMap, it is currently not available to developers for customization and thus has no method or field definitions. It defines the same functionality that is found in the java standard viewer toolbar. It extends javax.swing.JApplet and inherits all methods defined there.

See Also

[IMSMap](#)

LabelRenderer extends Renderer

A LabelRenderer is an object that represents a way of symbolizing features by drawing text on a feature. Field property is the name of the field in the layer that stores the text values to use as labels. The Symbol property defines how the text is drawn on the feature.

With the LabelRenderer, and using the setFields and setSeparator methods, you can set one or more Fields that will be used to provide text for labeling.

LabelRenderer.getSeparator

Description

The getSeparator method retrieves the current value specified to separate the display of multiple field values used to display labels on a feature. The separator can be any valid string character.

Syntax

```
String getSeparator()
```

Arguments

None

Returned Value

String returns the string character specified in the setSeparator method.

See Also

Renderer
setSeparator
IMSMMap

Example

```
var labelrenderer;  
labelrenderer = imsmap.getLayerLabelRenderer(myLayer);  
var separatorChar = labelrenderer.getSeparator();  
alert( "The separator used is: " + separatorChar );
```

LabelRenderer.setField

Description

The setField method allows you to use a single field to draw text on a feature. Any value set in the setSeparator method will be ignored.

Syntax

```
boolean setField(Layer, String)
```

Arguments

None

Returned Value

Layer	the layer object to apply the field to
String	The Field name used to store values used to draw text on a feature

See Also

Renderer	Layer
IMSMMap	

Example

```
var sym, labelrenderer;  
sym = imsmmap.createSymbol();  
labelrenderer = imsmmap.createRenderer("LABEL_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
labelrenderer.setSymbol(sym);  
labelrenderer.setField(myLayer, "field1");  
labelrenderer = imsmmap.getLayerLabelRenderer(myLayer);  
imsmmap.setLayerLabelRenderer(myLayer, labelrenderer );
```

LabelRenderer.setFields

Description

The setFields method allows you to use multiple fields to draw text on a feature. Be sure to use the setSeparator method when using this method to identify the string to use in the display between fields.

Syntax

```
boolean setFields(Layer layerObject, Collection col)
```

Arguments

Layer	layerObject – the layer object to apply the fields on
Collection	col – The collection object that contains the field names to use

Returned Value

boolean	returns true if the method was successful setting the separator value specified, false otherwise
---------	--

See Also

Renderer	Collection	Layer	ArcXML Programmer's Reference
----------	------------	-------	-------------------------------

Example

```
var col
var sym, labelrenderer;
sym = imsmap.createSymbol();
labelrenderer = imsmap.createRenderer("LABEL_RENDERERT");
col = imsmap.createCollection();
col.addStringElement("field1");
col.addStringElement("field2");
sym.setFont("times", 12);
sym.setAntialiasing("true");
labelrenderer.setSymbol(sym);
labelrenderer.setSeparator(":");
labelrenderer.setFields(myLayer, col);
labelrenderer = imsmap.getLayerLabelRenderer(myLayer);
```

LabelRenderer.setSeparator

Description

The setSeparator method allows you to set a separator character when multiple fields are used to display labels on a feature. The separator can be any valid string character.

Syntax

```
boolean setSeparator(String)
```

Arguments

String any valid string character that can be used to separate multiple field value displays when the text is drawn on a feature.

Returned Value

boolean returns true if the method was successful setting the separator value specified, false otherwise

See Also

Renderer IMSMap

Example

```
var col
var sym, labelrenderer;
sym = imsmap.createSymbol();
labelrenderer = imsmap.createRenderer("LABEL_RENDERER");
col = imsmap.createCollection();
col.addStringElement("field1");
col.addStringElement("field2");
sym.setFont("times", 12);
sym.setAntialiasing("true");
labelrenderer.setSymbol(sym);
labelrenderer.setSeparator(":");
labelrenderer.setFields(myLayer, col);
labelrenderer = imsmap.getLayerLabelRenderer(myLayer);
```

LabelRenderer.setSymbol

Description

Sets the Symbol to be used by the LabelRenderer to draw the text.

Syntax

```
boolean setSymbol(Symbol)
```

Arguments

Symbol A text symbol object that controls how text is rendered.

Returned Value

boolean returns true if the method was successful in setting the symbol specified,
 false otherwise

See Also

Renderer Symbol
ArcXML Programmer's Reference

Example

```
var sym, labelrenderer;  
sym = imsmap.createSymbol();  
labelrenderer = imsmap.createRenderer("LABEL_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
labelrenderer.setSymbol(sym);  
labelrenderer.setField("myfieldname");  
labelrenderer = imsmap.getLayerLabelRenderer(myLayer);  
imsmap.setLayerLabelRenderer(myLayer, labelrenderer);
```

Layer

Description

A Layer is a visual representation of data obtained from a spatial dataset. It contains methods which allow for getting and setting display properties as well as methods which help in distinguishing the type of data source being used to produce this layer. Construction of layers can be done through `IMSMMap.createXXXLayer()` methods

See Also

[IMSMMap](#)

Layer.clearScales

Description

Clears the existing scales limitations of the layer, allowing it to be drawn at all scales.

Syntax

boolean clearScales()

Arguments

None

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.getMaxScale](#)
[Layer.getMinScale](#)
[Layer.setMaxScale](#)
[Layer.setMinScale](#)

Example

```
layer.clearScales();
```

Layer.clearSelection

Description

Clears all features which are currently selected.

Syntax

`boolean clearSelection()`

Arguments

None

Returned Value

`boolean` True for success, false otherwise.

See Also

[Layer.setSelectable](#)
[Layer.setSelectionSymbol](#)

Example

```
layer.clearSelection();
```

Layer.getFieldNames

Description

Returns all the fields name in the Layer as a String with fields delimited by a pipe (|).

Syntax

```
String getFieldNames()
```

Arguments

None

Returned Value

String Field names delimited by a pipe (|). If layer is not a feature layer, then an empty String (“”) is returned.

See Also

[Layer.getMapTipField](#)
[Layer.setMapTipField](#)

[Layer.getQueryResultsFields](#)
[Layer.setQueryResultsFields](#)

Example

```
var names = layer.getFieldNames();
var nameArray = names.split("|");
```

Layer.getMapTipField

Description

Returns the name of field that is being used for map tips.

Syntax

```
boolean getMapTipField()
```

Arguments

None

Returned Value

String The name of the field being used for map tips. If no field has been selected, then null is returned.

See Also

[Layer.setMapTipField](#)

Example

```
var mtField = layer.getMapTipField();
```

Layer.getMaxScale

Description

Returns the maximum scale at which this layer is displayed. If the current scale value is greater than the maximum scale, the Layer will not be shown.

Syntax

```
double getMaxScale()
```

Arguments

None

Returned Value

double The maximum scale factor for this layer.

See Also

[Layer.getMinScale](#)
[Layer.setMaxScale](#)

Example

```
var max = layer.getMaxrScale();
```

Layer.getMinScale

Description

Returns the minimum scale factor at which this layer is displayed. If the current scale value is less than the minimum scale, the Layer will not be shown.

Syntax

```
double getMinScale()
```

Arguments

None

Returned Value

double The minimum scale factor for this layer.

See Also

[Layer.getMaxScale](#)
[Layer.setMinScale](#)

Example

```
var min = layer.getMinrScale();
```

Layer.getName

Description

Returns the name of the layer.

Syntax

`String getName()`

Arguments

None

Returned Value

`String` The name of the layer. If layer does not have a name or if one cannot be found, an empty String (“”) is returned.

See Also

[Layer.setName](#)

Example

```
var name = layer.getName();
```

Layer.getQueryResultsFields

Description

Returns a Collection containing the names of all the fields which are displayed when a identify is performed. Default is that all fields will be identifiable.

Syntax

Collection getQueryResultsFields()

Arguments

None

Returned Value

Collection The names that are displayed when an identify is made. Null is returned if no fields have been set as identifiable.

See Also

[Layer.setQueryResultsFields](#)

Example

```
var qrFields = layer.getQueryResultsFields();
```

Layer.getSelectionSet

Description

Returns the selection set of this layer as a String. The fields are delimited by a double colon and the features are delimited by a pipe.

Syntax

```
String getSelectionSet()
```

Arguments

None

Returned Value

String The selection set of this layer as a String. The fields are delimited by a double colon (::) and the features are delimited by a pipe (|). If none is found an empty String (“ ”) is returned.

See Also

[Layer.clearSelection](#)
[Layer.setSelectable](#)
[Layer.setSelectionSymbol](#)

Example

```
var selectionSet = layer.getSelectionSet();
```

Layer.hasGeocodeExtension

Description

Used to determine if the layer can be geocoded (has a Geocode Extension).

Syntax

```
boolean hasGeocodeExtension()
```

Arguments

None

Returned Value

boolean True if the layer can be geocoded, false otherwise.

Example

```
var isLayerGeocodable = layer.hasGeocodeExtension();
```

Layer.isAVIMSGroupLayer

Description

Used to determine if the layer is an ArcView IMS group layer.

Syntax

boolean isAVIMSGroupLayer()

Arguments

None

Returned Value

boolean True if the layer is an ArcView IMS group layer, false otherwise.

See Also

[Layer.isAVIMSSubLayer](#)

Example

```
var layer = imsMap.getSelectedLayer();
if(layer.isAVIMSGroupLayer() || layer.isMOIMSGroupLayer() || layer.isImageServerLayer()){
    alert("Current layer is a group layer.");
}
```

Layer.isAVIMSSubLayer

Description

Used to determine if the layer is an ArcView IMS sublayer.

Syntax

boolean isAVIMSSubLayer()

Arguments

None

Returned Value

boolean True if the layer is an ArcView IMS sublayer, false otherwise.

See Also

[Layer.isAVIMSGroupLayer](#)

Example

```
var layer=imsMap.getSelectedLayer();
if(layer.isAVIMSSubLayer() || layer.isMOIMSSubLayer() || layer.isImageServerSubLayer()){
    alert("Current layer is a sublayer.");
}
```

Layer.isFeatureLayer

Description

Used to determine if the layer is a feature layer. Note that a layer of an ArcIMS feature service, a feature layer which is a sublayer from an Image service (ArcIMS, MOIMS, and ArcView IMS), and layer from a local data source will all return true.

Syntax

boolean isFeatureLayer()

Arguments

None

Returned Value

boolean True if layer is a feature layer, false otherwise.

Example

```
var layer = imsMap.getSelectedLayer();
if(layer.isFeatureLayer()){
    alert("The current layer is a feature layer.");
}
```

Layer.isIdentifiable

Description

Used to determine if features on this layer can be identified.

Syntax

```
boolean isIdentifiable()
```

Arguments

None

Returned Value

boolean True if the layer is identifiable, false otherwise.

Example

```
var layer = imsMap.getSelectedLayer();
if(layer.isIdentifiable()){
    alert("The current layer is identifiable.");
}
```

Layer.isImageLayer

Description

Used to determine if the layer is an image layer.

Syntax

```
boolean isImageLayer()
```

Arguments

None

Returned Value

boolean True if the layer is an image layer, false otherwise.

Example

```
var layer = imsMap.getSelectedLayer();
if(layer.isImageLayer){
    alert("The current layer is an image layer.");
}
```

Layer.isImageServerLayer

Description

Used to determine if the layer is an image service (ArcIMS, MOIMS, or ArcView IMS) layer.

Syntax

```
boolean isImageServerLayer()
```

Arguments

None

Returned Value

boolean	True if the layer is an image service (ArcIMS, MOIMS, or ArcView IMS) layer, false otherwise.
---------	---

See Also

[Layer.isImageServerSubLayer](#)

Example

```
var layer = imsMap.getCurrentLayer();
if(layer.isAVIMSGroupLayer() || layer.isMOIMSGroupLayer() || layer.isImageServerLayer()){
    alert("Current layer is a group layer.");
}
```

Layer.isImageServerSubLayer

Description

Used to determine if the layer is an image service (ArcIMS, MOIMS, or ArcView IMS) sublayer.

Syntax

```
boolean isImageServerSubLayer()
```

Arguments

None

Returned Value

boolean True if the layer is an IMS image service (ArcIMS, MOIMS, or ArcView IMS) sublayer, false otherwise.

See Also

[Layer.isImageServerLayer](#)

Example

```
var layer = imsMap.getCurrentLayer();
if(layer.isAVIMSSubLayer() || layer.isMOIMSSubLayer() || layer.isImageServerSubLayer()){
    alert("Current layer is a sublayer.");
}
```

Layer.isMOIMSGroupLayer

Description

Used to determine if the layer is a MapObjects IMS group layer.

Syntax

boolean isMOIMSGroupLayer()

Arguments

None

Returned Value

boolean True if the layer is a MapObjects IMS group layer, false otherwise.

See Also

[Layer.isMOIMSSubLayer](#)

Example

```
var layer = imsMap.getCurrentLayer();
if(layer.isAVIMSGroupLayer() || layer.isMOIMSGroupLayer() ||
layer.isImageServerLayer()){
    alert("Current layer is a group layer.");
}
```

Layer.isMOIMSSubLayer

Description

Used to determine if the layer is a MapObjects IMS sublayer.

Syntax

boolean isMOIMSSubLayer()

Arguments

None

Returned Value

boolean True if the layer is a MapObjects IMS sublayer, false otherwise.

See Also

[Layer.isMOIMSGroupLayer](#)

Example

```
var layer = imsMap.getCurrentLayer();
if(layer.isAVIMSSubLayer() || layer.isMOIMSSubLayer() || layer.isImageServerSubLayer()){
    alert("Current layer is a sublayer.");
}
```

Layer.isSelectable

Description

Used to determine if the features on this layer can be selected.

Syntax

boolean isSelectable()

Arguments

None

Returned Value

boolean True if layer features are selectable, false otherwise.

See Also

[Layer.clearSelection](#)
[Layer.setSelectable](#)
[Layer.setSelectableByInt](#)
[Layer.setSelectionSymbol](#)

Example

```
var layer = imsMap.getSelectedLayer();
if(layer.isSelectable()){
    alert("The current layer is selectable.");
}
```

Layer.isVisible

Description

Used to determine if the layer is currently visible on the map or if it is hidden. This method will only return true if the layer is contained in the IMSToc and is currently checked as visible.

Syntax

boolean isVisible()

Arguments

None

Returned Value

boolean True if the layer is visible on the map, false otherwise.

See Also

[Layer.setVisible](#)

Example

```
function toggleVisibility{
    var layer = imsMap.getSelectedLayer();
    if(layer.isVisible()){
        layer.setVisible("false");
    }else{
        layer.setVisible("true");
    }
}
```

Layer.select

Description

Performs an attribute query and returns the search result as a String. The query string is an SQL where-clause expression. The return value is a String consisting of the results and has the fields delimited by a double colon and the rows by a pipe. For instance, a result table of:

91101 Altadena CA

92373 Redlands CA

would return the following String: 91101::Altadena::CA|92373::Redlands::CA

Syntax

String select(String query)

Arguments

query The search query to be performed on this layer.

Returned Value

String The method returns the results of the search as a String consisting of rows separated by pipes (|). Within each row, one or more field values are delimited by a double colon(::). If nothing is found then an empty String (“”) is returned.

See Also

[Layer.clearSelection](#)

[Layer.selectByCircle](#)

Example

```
var results = layer.select("POPULATION<3000000");
var resultArray = results.split("|");
```

Layer.selectByCircle

Description

Selects all the features within the circle with a given center and radius. The return value is a String consisting of the results and has the fields delimited by a double colon and the rows by a pipe.

For instance, a result table of:

91101 Altadena CA

92373 Redlands CA

would return the following String: 91101::Altadena::CA|92373::Redlands::CA

Syntax

String selectByCircle(double x, double y, double radius)

Arguments

x,y The x and y coordinates of the center of the circle.

radius The radius in map units of the circle to be selected.

Returned Value

String The method returns the results of the search as a String consisting of rows separated by pipes (|). Within each row, one or more filed values are delimited by a double colon(::). If nothing is found then an empty String ("") is returned.

See Also

[Layer.clearSelection](#)

[Layer.select](#)

Example

```
var results = layer.select(10, 10, 1.5);
var resultArray = results.split("|");
```

Layer.setFeaturesSelectable

Description

Allows or prohibits selection of features on a layer. The layer must be a feature layer.

Syntax

```
boolean setFeaturesSelectable(int selectable)
```

Arguments

selectable 1 allows feature selection on a layer, 0 prohibits feature selection on a layer.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.isSelectable](#) [Layer.setSelectable](#)
[Layer.setSelectableByInt](#)

Example

```
Function toggleSelectability(){
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    if(layer.isSelectable()){
        layer.setFeaturesSelectable(0);
    }else{
        layer.setFeaturesSelectable(1);
    }
}
```

Layer.setIdentifiable

Description

Enable/disable the identifiable property for this layer. If set to true then features can be identified (through use of the default identify tool for example).

Syntax

```
boolean setIdentifiable(boolean allowIdentify)
```

Arguments

allowIdentify Set to true to allow for the identification of features, false otherwise.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.isIdentifiable](#)
[Layer.setIdentifiableByInt](#)

Example

```
Function toggleIdentifiability(){
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    layer.setIdentifiable(!layer.isIdentifiable());
}
```

Layer.setIdentifiableByInt

Description

Enable/disable the identifiable property for this layer. If set to 1 then features can be identified (through use of the default identify tool for example), 0 disables feature identification. This method is offered as a work around alternative to Layer.setIdentifiable() for issues that may arise in some browsers passing boolean values in JavaScript.

Syntax

boolean setIdentifiable(int allowIdentify)

Arguments

allowIdentify Set to 1 to allow for the identification of features, 0 otherwise.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.isIdentifiable](#)

[Layer.setIdentifiable](#)

Example

```
Function toggleIdentifiability(){
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    if(layer.isIdentifiable()){
        layer.setIdentifiableByInt(0);
    }else{
        layer.setIdentifiableByInt(1);
    }
}
```

Layer.setMapTipField

Description

Set the map tip field of a feature layer. When the mouse cursor points to a feature then the value in this field will be shown on the map.

Syntax

```
boolean setMapTipField(String fieldName)
```

Arguments

fieldName The name of the layer field to be used for map tips. If fieldName is not valid for this layer, this method has no effect.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.getMapTipField](#)

Example

```
var names = layer.getFieldNames().split("|");
for(int i = 0; i < names.length; i++){
    if(names[i].toUpperCase().indexOf("NAME") != -1){
        layer.setMapTipField(names[i]);
        break;
    }
}
```

Layer.setMaxScale

Description

Sets the maximum display scale for this layer. If the map scale is greater than this scale then the layer will not be drawn.

Syntax

```
boolean setMaxScale(double scale)
```

Arguments

scale Maximum scale for the layer to be drawn.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.clearScales](#)
[Layer.setMinScale](#)

Example

```
var currentScale = ImsMap.getScale();
layer.setMaxScale(currentScale);
```

Layer.setMinScale

Description

Sets the minimum display scale for this layer. If the map scale is less than this scale then the layer will not be drawn.

Syntax

```
boolean setMinScale(double scale)
```

Arguments

scale Minimum scale for the layer to be drawn.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.clearScales](#)
[Layer.setMaxScale](#)

Example

```
var currentScale = ImsMap.getScale();
layer.setMinScale(currentScale);
```

Layer.setName

Description

Sets the name of the layer.

Syntax

```
boolean setName(String layerName)
```

Arguments

layerName The new name for this layer.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.getName](#)

Example

```
layer.setName("Infected_sites");
```

Layer.setQueryResultFields

Description

Uses a Collection to set the list of all the fields which are displayed when a identify is performed. Default is that all fields will be identifiable. The Collection must either have String objects or NameValuePair objects (where the name is the name of the field and the value is its alias). If a NameValuePair element is found in the Collection then the value will be used as an *alias* for that field.

Syntax

```
boolean setQueryResultsFields(Collection fields)
```

Arguments

fields This is the Collection of Strings or of NameValuePairs of for the fields, that are to be displayed during an identify. If a NameValuePair element is found in the Collection then the value will be used as an *alias* for that field.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.getQueryResultsFields](#)

Example

```
var field1 = imsMap.createNameValuePair("Names", "Names of Employers");
var field2 = imsMap.createNameValuePair("Aage_Employee", "Average Age of Employees");
var collection = imsMap.createCollection();
collection.addNameValuePairElement(field1);
collection.addNameValuePairElement(field2);
setQueryResultsFields(collection);
```

Layer.setSelectable

Description

Allows or prohibits selection of features on a layer. The layer must be a feature layer.

Syntax

```
boolean setSelectable(boolean selectable)
```

Arguments

selectable True allows feature selection on a layer, false prohibits feature selection on a layer.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.isSelectable](#) [Layer.setFeaturesSelectable](#)
[Layer.setSelectableByInt](#)

Example

```
Function toggleSelectability(){
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    layer.setFeaturesSelectable(!layer.isSelectable());
}
```

Layer.setSelectableByInt

Description

Allows or prohibits selection of features on a layer. This method is offered as a work around alternative to Layer.setSelectable() for issues that may arise in some browsers passing boolean values in JavaScript.

Syntax

boolean setSelectableByInt(int selectable)

Arguments

selectable 1 allows feature selection on a layer, 0 prohibits feature selection on a layer.

Returned Value

boolean True for success, false otherwise.

See Also

[Layer.isSelectable](#)
[Layer.setFeaturesSelectable](#)
[Layer.setSelectable](#)

Example

```
Function toggleSelectability(){
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    if(layer.isSelectable()){
        layer.setSelectableByInt(0);
    }else{
        layer.setSelectableByInt(1);
    }
}
```

Layer.setSelectionSymbol

Description

Sets the Symbol which is to be used for the display of selected features on this layer.

Syntax

```
boolean setSelectionSymbol(Symbol selectionSymbol)
```

Arguments

selectionSymbol The new Symbol to be used for selected features on this layer.

Returned Value

boolean True for success, false otherwise.

Example

```
var newSymbol = imsMap.createSymbol("MARKER_SYMBOL");
newSymbol.setSize(24);
layer.setSelectionSymbol(newSymbol);
```

Layer.setVisible

Description

Used to set the layer's currently visibility on the map.

Syntax

boolean setVisible(String visibility)

Arguments

visibility “true” to make the layer visible on the map, “false” to hide the current layer.

Returned Value

boolean True if the layer is visible on the map, false otherwise.

See Also

[Layer.isVisible](#)

[Layer.setVisibleByInt](#)

Example

```
function toggleVisibility{
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    if(layer.isVisible()){
        layer.setVisible("false");
    }else{
        layer.setVisible("true");
    }
}
```

Layer.setVisibleByInt

Description

Used to set the layer's currently visibility on the map. This method is offered as a work around alternative to Layer.setVisible() for issues that may arise in some browsers passing boolean values in JavaScript.

Syntax

```
boolean setVisibleByInt(int visibility)
```

Arguments

visibility 1 to make the layer visible on the map, 0 to hide the current layer.

Returned Value

boolean True if the layer is visible on the map, false otherwise.

See Also

[Layer.isVisible](#)
[Layer.setVisible](#)

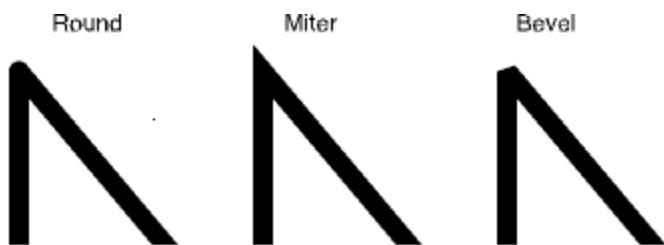
Example

```
function toggleVisibility{
    var layer = parent.mapFrame.document.IMSMap.getSelectedLayer();
    if(layer.isVisible()){
        layer.setVisibleByInt(0);
    }else{
        layer.setVisibleByInt(1);
    }
}
```

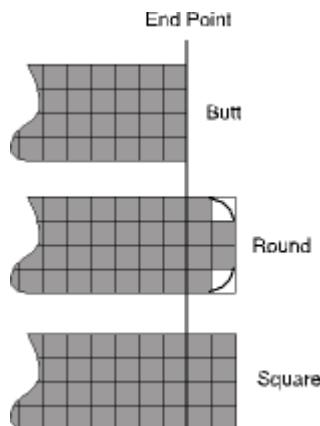
LineSymbol extends Symbol

This symbol draws line features. You can create an instance of the symbol through the `IMSMMap.createSymbol` method. The styles of the line symbol are: solid, dash, dot, dash-dot, dash-dot-dot and it provides special effects like antialiasing, transparency and overlap. The style property is handled by the `setgetStyle` methods. There are also two properties—`captype` and `jointype`—that provide different ways of line cap and line joining as shown below:

jointypes:



captypes:



See Also

[ArcXML Programmer's Reference](#)
[IMSMMap](#)
[PolygonSymbol](#)
[Symbol](#)

LineSymbol.getAntialiasing

Description

Retrieves the current antialiasing property value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a LineSymbol by default.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean	true if antialiasing is to be used, false otherwise
---------	---

See Also

[LineSymbol.setAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("default value of antialiasing property for LineSymbol is " +
symbol.getAntialiasing());
```

LineSymbol.getCapType

Description

Returns the current the cap type of a line symbol. The default cap type is round.

Syntax

```
int getCapType()
```

Arguments

None

Returned Value

int:	0 – butt
	1 - round
	2 – square

See Also

[LineSymbol.setCapType](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("default cap type for the LineSymbol is " + symbol.getCapType());
```

LineSymbol.getColor

Description

Returns the current color of the line symbol. The default color is black.

Syntax

```
String getColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[LineSymbol.setColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("the default color of the LineSymbol is " + symbol.getColor());
```

LineSymbol.getJoinType

Description

Returns the current join type of the line symbol. The default value is round.

Syntax

```
int getJoinType()
```

Arguments

None

Returned Value

int:	0 - miter
	1 - round
	2 - bevel

See Also

[LineSymbol.setJoinType](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("the default join type of the LineSymbol is " + symbol.getJoinType());
```

LineSymbol.getStyle

Description

Returns the current style of the line symbol. The default style is solid.

Syntax

```
int getStyle()
```

Arguments

None

Returned Value

int:	0 – solid line
	1 - dash line
	2 – dot line
	3 – dash-dot line
	4 – dash-dot-dot line

See Also

[LineSymbol.setStyle](#)

Example

See [LineSymbol.setJoinStyle](#)

LineSymbol.getTransparency

Description

The getTransparency method retrieves the current transparency value set on the object. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

current line transparency. It must have a value between 0 and 1.0

See Also

[LineSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("the default transparency value of the LineSymbol is " + symbol.getTransparency());
```

LineSymbol.getWidth

Description

Returns the current width of the line symbol in screen pixels. The default value is 1.

Syntax

```
int getWidth()
```

Arguments

None

Returned Value

the current width of the line symbol in screen pixels

See Also

[LineSymbol.setWidth](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
alert("the default width of the LineSymbol is " + symbol.getWidth());
```

LineSymbol.setAntialiasing

Description

Sets the current antialiasing property value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a LineSymbol by default.

Syntax

```
boolean setAntialiasing( String enabled)
```

Arguments

enabled “true” is to set antialiasing on,
“false” is to set it off.

Returned Value

true if the method was successful, false otherwise

See Also

[LineSymbol.getAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
symbol.setAntialiasing("true")
```

LineSymbol.setCapType

Description

Sets the current cap type of a line symbol. The default cap type is round.

Syntax

```
boolean setCapType( int capType)
```

Arguments

capType:
 0 – butt
 1 - round
 2 – square

Returned Value

true if the method was successful, false otherwise

See Also

[LineSymbol.getCapType](#)

Example

See the LineSymbol.setJoinCapType section and comments there

LineSymbol.setColor

Description

Sets the current color of the line symbol. The default color is black.

Syntax

```
boolean setColor( Color color)
```

Arguments

color – an instance of the `Color` class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

`LineSymbol.getColor()`
`Color`

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setColor(color1)
```

LineSymbol.setJoinType

Description

Sets the current join type of the line symbol. The default value is round.

Syntax

```
boolean setJoinType( int joinType)
```

Arguments

joinType: 0 – miter

 1 - round

 2 – bevel

Returned Value

true if the method was successful, false otherwise

See Also

[LineSymbol.getJoinType](#)

Example

see the next page

LineSymbol.setJoinType

Example

```
var layer = mapFrame.IMSMap.getLayer("Highways");
var rend = mapFrame.IMSMap.createRenderer("SIMPLE_RENDERER");
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
symbol.setJoinType(2);
symbol.setCapType(2);
symbol.setStyle(2);
symbol.setWidth(5));
alert("current style is " + symbol.getStyle());
alert(rend.setSymbol(symbol));
mapFrame.IMSMap.setLayerRenderer(layer, rend);
mapFrame.IMSMap.redraw();
setCatType(), setStyle(), setWidth() and getStyle() methods will work correctly only after
setJoinType method has been called
```

LineSymbol.setStyle

Description

Sets the current style of the line symbol. The default style is solid.

Syntax

```
boolean setStyle( int style)
```

Arguments

style: 0 – solid line
1 - dash line
2 – dot line
3 – dash-dot line
4 – dash-dot-dot line

Returned Value

true if the method was successful, false otherwise

See Also

[LineSymbol.getStyle](#)

Example

See [LineSymbol.setJoinStyle](#)

LineSymbol.setTransparency

Description

The setTransparency method sets the current transparency value set on the object. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
boolean setTransparency( double transparencyValue)
```

Arguments

transparencyValue – defines transparency to be set. It have have a value between 0 and 1.0

Returned Value

true if the method was successful, false otherwise

See Also

[LineSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("LINE_SYMBOL");
symbol.setTransparency(0.5);
```

LineSymbol.setWidth

Description

Returns the current width of the line symbol in screen pixels. The default value is 1

Syntax

```
boolean setWidth( int width)
```

Arguments

width – the width is in screen pixels

Returned Value

true if the method was successful, false otherwise

See Also

[LineSymbol.getWidth](#)

Example

See [LineSymbol.setJoinStyle](#)

MarkerSymbol extends Symbol

The MarkerSymbol is used to symbolize point features by means of one of the predefined symbols: circle, triangle, square, cross, or star. The particular symbol can be set by calling `setStyle` method. The MarkerSymbol provides special effects like antialiasing, overlap, outline and shadows. You can create an instance of the class by calling `IMSMMap.createSymbol` method.

See Also

ArcXML Programmer's Reference

IMSMMap

Symbol

MarkerSymbol.getAntialiasing

Description

Retrieves the current antialiasing value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a MarkerSymbol by default.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

true if antialiasing is to be used, false otherwise

See Also

[MarkerSymbol.setAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
alert("default value of antialiasing for a MarkerSymbol is " + symbol.getAntialiasing());
```

MarkerSymbol.getColor

Description

Returns the current color of the marker symbol. The default color value is black.

Syntax

```
String getColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[MarkerSymbol.setColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
alert("default color of a MarkerSymbol is " + symbol.getColor());
```

MarkerSymbol.getOutlineColor

Description

Returns the current outline color of the marker symbol. As the default outline color is undefined, it is needed to call setOutlineColor method first before using getOutlineColor method.

Syntax

String getOutlineColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[MarkerSymbol.setOutlineColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
var color = mapFrame.IMSMap.createColor(0,255,0);
alert("set the outline color of to green" + symbol.setGlowColor(color));
alert("the current outline color for the MarkerSymbol is " +
symbol.getOutLineColor());
```

MarkerSymbol.getShadowColor

Description

Returns the shadow color of the marker symbol. As the default shadow color is undefined, it is needed to call setShadowColor method first before using getShadowColor method.

Syntax

String getShadowColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[MarkerSymbol.setShadowColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
var color = mapFrame.IMSMap.createColor(0,255,0);
alert("set the outline color of to green" + symbol.setShadowColor(color));
alert("the current shadow color for the MarkerSymbol is " +
symbol.getOutLineColor());
```

MarkerSymbol.getSize

Description

Returns the current size of the marker symbol in screen pixels. The default size is 3.

Syntax

```
int getSize()
```

Arguments

None

Returned Value

Returns the current size value in screen pixels

See Also

[MarkerSymbol.setSize](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
alert("default size for a MarkerSymbol is " + symbol.getSize());
```

MarkerSymbol.getStyle

Description

Returns the current style of the marker symbol. The possible styles are circle, triangle, square, cross and star. The default style is circle.

Syntax

```
int getStyle()
```

Arguments

None

Returned Value

int:	0 - circle
	1 - square
	2 - triangle
	3 - cross
	4 - star

See Also

[MarkerSymbol.setStyle](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
alert("the default style for a MarkerSymbol is "+ symbol.getStyle());
```

MarkerSymbol.getTransparency

Description

The `getTransparency` method retrieves the current transparency value set on the object. The default value is 1.0. The valid value range is from 0.0 (Transparent) to 1.0 (Opaque).

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

Double value ranging from 0.0 to 1.0

See Also

[MarkerSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
alert("default value of the transparency property is " + symbol.getTransparency());
smooth or smoother. Antialiasing is switched off for a MarkerSymbol by default.
```

MarkerSymbol.setAntialiasing

Description

Sets the current antialiasing value of the symbol. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear

Syntax

```
boolean setAntialiasing( String enabled)
```

Arguments

enabled “true” is to set antialiasing on,
“false” is to set it off.

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
symbol.setAntialiasing("true")
```

MarkerSymbol.setColor

Description

Sets the current color of the marker symbol. The default color value is black.

Syntax

```
boolean setColor( Color color)
```

Arguments

color – an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setColor(color1)
instance of the Color class that defines the color to be set.
```

MarkerSymbol.setOutlineColor

Description

Sets the current outline color of the marker symbol. The default outline color is undefined.

Syntax

```
boolean setOutlineColor( Color color)
```

Arguments

color – an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getOutlineColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setOutlineColor(color1)
```

MarkerSymbol.setShadowColor

Description

Sets the shadow color of the marker symbol. The default shadow color is undefined.

Syntax

```
boolean setShadowColor( Color color)
```

Arguments

color – an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getShadowColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setShadowColor(color1)
```

MarkerSymbol.setSize

Description

Sets the current size of the marker symbol in screen pixels. The default size is 3.

Syntax

boolean setSize(int size)

Arguments

size – the size value in screen pixels

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getSize](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
symbol.setSize(10);
```

MarkerSymbol.setStyle

Description

Sets the current style of the marker symbol. The possible styles are circle, triangle, square, cross and star. The default style is circle.

Syntax

boolean `setStyle(int style)`

Arguments

style: 0 – circle
1 – square
2 - triangle
3- cross
4 - star

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getStyle](#)

Example

see the next page

MarkerSymbol.setStyle

Example

```
var layer = mapFrame.IMSMap.getLayer("Agencies");
var rend = mapFrame.IMSMap.createRenderer("SIMPLE_RENDERER");
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
symbol.setSize(10);
symbol.setStyle(4);
var color1 = mapFrame.IMSMap.createColor(0,255,0);
var color2 = mapFrame.IMSMap.createColor(0,0,255);
var color3 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setColor(color1);
symbol.setShadowColor(color2);
symbol.setOutlineColor(color3);

alert("the current size is" + symbol.getSize());
alert("the current style is " + symbol.getStyle());
alert("the current color is " + symbol.getColor());
alert("the current shadow color is " + symbol.getShadowColor());
alert(" the current outline color is " + symbol.getOutlineColor());
rend.setSymbol(symbol);
mapFrame.IMSMap.setLayerRenderer(layer, rend);
mapFrame.IMSMap.redraw();
```

MarkerSymbol.setTransparency

Description

Sets the current transparency value set on the object. The default value is 1.0. The valid value range is from 0.0 (Transparent) to 1.0 (Opaque).

Syntax

```
boolean setTransparency( double transparencyValue)
```

Arguments

transparencyValue defines transparency to be set. It must be between 0.0 and 1.0

Returned Value

true if the method was successful, false otherwise

See Also

[MarkerSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
symbol.setTransparency(0.5);
```

NameValuePair

Description

This class is used to keep a name and value pair. NameValuePairs are used to store values in a Collection (e.g. as a field name and an alias for that field). NameValuePairs are constructed using `IMSMMap.createNameValuePair()`.

See Also

[Collection](#)
[IMSMMap](#)
[Layer.setQueryResultsFields\(\)](#)

NameValuePair.getName

Description

Gets the name property of a NameValuePair as a String.

Syntax

```
String getName()
```

Arguments

None

Returned Value

String The name property of the NameValuePair.

See Also

[NameValuePair.getValue](#)
[NameValuePair.setName](#)

Example

```
var name = nameValuePair.getName();
```

NameValuePair.getValue

Description

Gets the value property of a NameValuePair as a String.

Syntax

```
String getValue()
```

Arguments

None

Returned Value

String The value property of the NameValuePair.

See Also

[NameValuePair.getName](#)
[NameValuePair.setValue](#)

Example

```
var value = nameValuePair.getValue();
```

NameValuePair.setName

Description

Sets the name property of a NameValuePair with a String.

Syntax

```
boolean setName(String name)
```

Arguments

name The name property of the NameValuePair.

Returned Value

boolean True for success, false otherwise.

See Also

[NameValuePair.getName](#)
[NameValuePair.setValue](#)

Example

```
var nameValuePair = imsMap.createNameValuePair("France", "World Champion");
...
nameValuePair.setName("Italia");
```

NameValuePair.setValue

Description:

Sets the value property of a NameValuePair with a String.

Syntax:

```
boolean setValue(String value)
```

Arguments:

value The value property of the NameValuePair.

Returned Value:

boolean True for success, false otherwise.

See Also

[NameValuePair.getValue](#)

[NameValuePair.setName](#)

Example

```
var nameValuePair = imsMap.createNameValuePair("Jack", "the boss");
...
nameValuePair.setValue("the president");
```

PolygonSymbol extends Symbol

This class contains methods to draw a polygon feature. It is supposed that a polygon feature consists of an inner area (fill area) and a boundary that looks like a line feature. Both parts of the polygon feature can be handled separately. As to concern the boundary the methods are similar as for the LineSymbol class. You can create an instance of the polygon symbol through the IMSMap.createSymbol method.

See Also

ArcXML Programmer's Reference
IMSMap
LineSymbol
Symbol

PolygonSymbol.getAntialiasing

Description

Retrieves the current antialiasing property value of the symbol boundary. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. Antialiasing is switched off for a polygon symbol by default.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean true if antialiasing is to be used, false otherwise

See Also

[PolygonSymbol.setAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("default value of antialiasing property for the polygon symbol is " +
symbol.getAntialiasing());
```

PolygonSymbol.getBoundaryColor

Description

Returns the current color of the boundary of the polygon symbol. The default value is black.

Syntax

```
String getBoundaryColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[PolygonSymbol.setBoundaryColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("default value of the boundary color for the polygon symbol is " +
symbol.getBoundaryColor());
```

PolygonSymbol.getCapStyle

Description

Returns the current cap type of the polygon symbol boundary. The default cap type is round. The cap types are defined in the same way as for the line symbol.

Syntax

```
int getCapType()
```

Arguments

None

Returned Value

int:	0 – butt
	1 - round
	2 – square

See Also

[LineSymbol.getCapType](#)
[PolygonSymbol.setCapType](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("the default boundary cap type for the polygon symbol is " + symbol.getCapType());
```

PolygonSymbol.getFillColor

Description

Returns the current fill color of the polygon symbol. This method must be used after at least one call of the setFillColor method.

Syntax

```
String getFillColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):

- R defines Red part of the RGB color value, should be between 0 and 255
- G defines Green part of the RGB color value, should be between 0 and 255
- B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[PolygonSymbol.setFillColor](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setFillColor(color1);
alert("the current fill color of the polygon symbol is "+symbol.getFillColor());
```

PolygonSymbol.setFillTransparency

Description

Returns the current fill transparency value of the polygon symbol. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getFillTransparency()
```

Arguments

None

Returned Value

Returns fill transparency. It may have a value between 0 and 1.0

See Also

[PolygonSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("the default fill transparency value for the polygon symbol is " +
symbol.getTransparency());
```

PolygonSymbol.getJoinStyle

Description

Returns the current boundary join type of the polygon symbol. The default value is round. The join types are defined in the same way as for the line symbol.

Syntax

```
int getJoinType()
```

Arguments

None

Returned Value

int:	0 – miter
	1 - round
	2 – bevel

See Also

[LineSymbol.getJoinType](#)
[PolygonSymbol.setJoinType](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("the default boundary join type of the PolygonSymbol is " + symbol.getJoinType());
```

PolygonSymbol.getStyle

Description

Returns the current boundary style of the polygon symbol. A polygon boundary as a line feature may have the following styles: solid line, dash line, dot line, dash-dotline, dash-dot-dot line. The default style is solid.

Syntax

```
int getStyle()
```

Arguments

None

Returned Value

int:	0 – solid line
	1 - dash line
	2 – dot line
	3 – dash-dot line
	4 – dash-dot-dot line

See Also

[PolygonSymbol.setStyle](#)
[LineSymbol.getStyle](#)

Example

See [PolygonSymbol.setJoinStyle](#)

PolygonSymbol.getTransparency

Description

Returns the boundary transparency value of the polygon symbol. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

Returns boundary transparency. It must have a value between 0 and 1.0

See Also

[PolygonSymbol.setTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("the default boundary transparency value of the PolygonSymbol is " +
symbol.getTransparency());
```

PolygonSymbol.getWidth

Description

Returns the current boundary width of the polygon symbol in screen pixels. The default value is 1.

Syntax

```
float getWidth()
```

Arguments

None

Returned Value

the current boundary width of the polygon symbol in screen pixels

See Also

[PolygonSymbol.setWidth](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
alert("the default boundary width value of the PolygonSymbol is "+ symbol.getWidth());
```

PolygonSymbol.setAntialiasing

Description

Sets the current antialiasing property value of the polygon symbol boundary. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother. By default, antialiasing is switched off for a polygon symbol.

Syntax

```
boolean setAntialiasing( String enabled)
```

Arguments

enabled “true” is to set antialiasing on,
“false” is to set it off.

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getAntialiasing](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
symbol.setAntialiasing("true")
```

PolygonSymbol.setBoundaryColor

Description

Sets the current color of the boundary of the polygon symbol. The default value is black

Syntax

```
boolean setBoundaryColor( Color color)
```

Arguments

color – an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getBoundaryColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setBoundaryColor(color1)
```

PolygonSymbol.setCapStyle

Description

Sets the current cap type of the polygon symbol boundary. The default cap type is round. The cap types are defined in the same way as for the line symbol.

Syntax

```
boolean setCapStyle( int capStyle)
```

Arguments

capStyle: 0 – butt
1 - round
2 – square

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getCapStyle](#)

Example

See [PolygonSymbol.setJoinCapStyle](#)

PolygonSymbol.setFillColor

Description

Sets the current fill color of the polygon symbol. This method must be used after at least one call of the setFillColor method.

Syntax

```
boolean setFillColor(Color color)
```

Arguments

color—an instance of the Color class that defines the color to be set.

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getFillColor](#)
[Color](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
var color1 = mapFrame.IMSMap.createColor(255,0,0);
symbol.setFillColor(color1)
```

PolygonSymbol.setFillTransparency

Description

Sets the current fill transparency value of the polygon symbol. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
boolean setFillTransparency( double transparencyValue)
```

Arguments

transparencyValue – defines transparency value to be set. It may have a value between 0 and 1.0.

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
symbol.setFillTransparency(0.5);
```

PolygonSymbol.setJoinStyle

Description

Sets the current boundary join type of the polygon symbol. The default value is round. The join types are defined in the same way as for the line symbol.

Syntax

```
boolean setJoinStyle( int joinStyle)
```

Arguments

joinStyle: 0 – miter
 1 - round
 2 – bevel

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getJoinStyle](#)
[LineSymbol.setJoinType](#)

Example

see the next page

PolygonSymbol.setJoinStyle

Example

```
var layer = mapFrame.IMSMap.getLayer("County");
var rend = mapFrame.IMSMap.createRenderer("SIMPLE_RENDERER");
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
symbol.setJoinStyle(2);
symbol.setCapStyle(2);
symbol.setStyle(2));
symbol.setWidth(5.0E0));
alert("current style is " + symbol.getStyle());
alert(rend.setSymbol(symbol));
mapFrame.IMSMap.setLayerRenderer(layer, rend);
mapFrame.IMSMap.redraw();
```

setCatStyle(), setStyle(), setWidth() and getStyle() methods will work correctly only after setJoinStyle method has been called

PolygonSymbol.setStyle

Description

Sets the current boundary style of the polygon symbol. A polygon boundary as a line feature may have the following styles: solid line, dash line, dot line, dash-dot line, dash-dot-dot line. The default style is solid.

Syntax

```
boolean setStyle( int style)
```

Arguments

style: 0 – solid line
1 - dash line
2 – dot line
3 – dash-dot line
4 – dash-dot-dot line

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getStyle](#)
[LineSymbol.setStyle](#)

Example

See [PolygonSymbol.setJoinStyle](#)

PolygonSymbol.setTransparency

Description

Sets the boundary transparency value of the polygon symbol. The default value is 1.0. The valid range is from 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
boolean setTransparency( double transparencyValue)
```

Arguments

transparencyValue – defines transparency value to be set. It must be between 0 and 1.0.

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getTransparency](#)

Example

```
var symbol = mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
symbol.setTransparency(0.5);
```

PolygonSymbol.setWidth

Description

Sets the current boundary width of the polygon symbol in screen pixels. The default value is 1.

Syntax

```
boolean setWidth( float width)
```

Arguments

width – the width is in screen pixels

Returned Value

true if the method was successful, false otherwise

See Also

[PolygonSymbol.getWidth](#)
[LineSymbol.setWidth](#)

Example

See [PolygonSymbol.setJoinStyle](#)

RasterFillSymbol

Description

Symbol used to fill the interior of a polygon using a raster image. The class includes methods for setting and getting of display properties for this symbol. The method `IMSMMap.createSymbol()` is used to create an object of this type.

See Also

[IMSMMap](#)
[RasterMarkerSymbol](#)
[RasterShieldSymbol](#)
[ShieldSymbol](#)

RasterFillSymbol.getAntialiasing

Description

Returns true if antialiasing is used to produce this Symbol, false if it is not.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean	True if antialiasing is used on this Symbol, false otherwise.
---------	---

See Also

[RasterFillSymbol.setAntialiasing](#)

Example

```
var isAA = rasterFillSymbol.getAntialiasing();
```

RasterFillSymbol.getImagePathString

Description

Returns the path to the image being used for this symbol. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”) and thus can only be viewed on the local machine. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “http://theworldsbestmapsite/images/symbol.jpg”).

Syntax

```
String getImagePathString()
```

Arguments

None

Returned Value

String The path of the image being used for this symbol.

See Also

[RasterFillSymbol.setImagePathString](#)

Example

```
var imagePath = rasterFillSymbol.getImagePathString();
```

RasterFillSymbol.getTransparency

Description:

Returns the current transparency of the Symbol. This value is a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

double The current transparency value of the Symbol. This value must be between 0.0 and 1.0, otherwise it is ignored.

See Also

[RasterFillSymbol.setTransparency](#)

Example

```
var transparency = rasterFillSymbol.getTransparency();
```

RasterFillSymbol.getURLString

Description

Returns the URL for the image as a String. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “`http://theworldsbestmapsite/images/symbol.jpg`”) and, when served, can be viewed by any client site. The image path is used to define images used for local projects only (e.g. “`C:\images\symbol.jpg`”).

Syntax

```
String getURLString()
```

Arguments

None

Returned Value

String The URL for the image.

See Also

[RasterFillSymbol.setURLString](#)

Example

```
var url = rasterFillSymbol.getURLString();
```

RasterFillSymbol.setAntialiasing

Description

Set to define whether or not antialiasing should be used to produce this Symbol.

Syntax

```
boolean setAntialiasing(String value)
```

Arguments

value Use “true” if antialiasing should be used, “false” otherwise.

Returned Value

boolean True for success, false otherwise

See Also

[RasterFillSymbol.getAntialiasing](#)

Example

```
rasterFillSymbol.setAntialiasing("true");
```

RasterFillSymbol.setImagePathString

Description

Sets the path to the image being used for this Symbol. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”) and thus can only be viewed on the local machine. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “http://theworldsbestmapsite/images/ symbol.jpg”).

Syntax

```
boolean setImagePathString(String path)
```

Arguments

path The absolute path to the image used for this symbol.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterFillSymbol.getImagePathString](#)

Example

```
rasterFillSymbol.setImagePathString("C:\ArcIMS\images\fillImages\sites.jpg");
```

RasterFillSymbol.setTransparency

Description

Sets the current transparency of the Symbol. This gives a mechanism to allow features behind a symbol be seen. The transparency is defined as a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax

```
boolean setTransparency(double transparency)
```

Arguments

transparency Value between 0.0 and 1.0 to set the transparency property to. Default is 1.0.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterFillSymbol.getTransparency](#)

Example

```
rasterFillSymbol.setTransparency(.5);
```

RasterFillSymbol.setURLString

Description

Sets the URL for the symbol image as a String. The URLpath is used by the spatial server to define the location of an image for a project being served (e.g. “<http://theworldsbestmapsite/images/symbol.jpg>”) and, when served, can be viewed by any client site. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”).

Syntax

```
boolean setURLString(String url)
```

Arguments

url The URL of the image.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterFillSymbol.getURLString](#)

Example

```
rasterFillSymbol.setURLString("http://www.esri.com/software/arcims/graphics/arcims.gif");
```

RasterMarkerSymbol

Description

The RasterMarkerSymbol is used as a point marker symbol using a raster image. The class includes methods for setting and getting of display properties for this symbol. The method `IMSMMap.createSymbol()` is used to create an object of this type.

See Also

[IMSMMap](#)
[RasterFillSymbol](#)
[RasterShieldSymbol](#)
[ShieldSymbol](#)

RasterMarkerSymbol.getAntialiasing

Description

Returns true if antialiasing is used to produce this Symbol, false if it is not.

Syntax

```
boolean getAntialiasing()
```

Arguments

None.

Returned Value

boolean	True if antialiasing is used on this Symbol, false otherwise.
---------	---

See Also

[RasterMarkerSymbol.setAntialiasing](#)

Example

```
var isAA = rasterMarkerSymbol.getAntialiasing();
```

RasterMarkerSymbol.getHotSpotX

Description

Returns the x-shift of the symbol from the feature it represents.

Syntax

```
int getHotSpotX()
```

Arguments

None

Returned Value

int The x-shift of the symbol

See Also

[RasterMarkerSymbol.getHotSpotY](#)
[RasterMarkerSymbol.setHotSpotX](#)
[RasterMarkerSymbol.setHotSpotY](#)

Example

```
var hotSpotX = rasterMarkerSymbol.getHotSpotX();
```

RasterMarkerSymbol.getHotSpotY

Description

Returns the y-shift of the symbol from the feature it represents.

Syntax

```
int getHotSpotY()
```

Arguments

None

Returned Value

int The y-shift of the symbol

See Also

[RasterMarkerSymbol.getHotSpotX](#)
[RasterMarkerSymbol.setHotSpotX](#)
[RasterMarkerSymbol.setHotSpotY](#)

Example

```
var hotSpotY = rasterMarkerSymbol.getHotSpotY();
```

RasterMarkerSymbol.getImagePathString

Description

Returns the path to the image being used for this symbol. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”) and thus can only be viewed on the local machine. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “http://theworldsbestmapsite/images/symbol.jpg”).

Syntax

String getImagePathString()

Arguments

None

Returned Value

String The path of the image being used for this symbol.

See Also

[RasterMarkerSymbol.setImagePathString](#)

[RasterMarkerSymbol.getURLString](#)

Example

```
var imagePath = rasterMarkerSymbol.getImagePathString();
```

RasterMarkerSymbol.getShadowColor

Description

Returns the current color of the symbol shadow as a String consisting of RGB values delimited by commas.

Syntax

```
String getShadowColor()
```

Arguments

None

Returned Value

String The RGB values of the shadow color delimited by commas (e.g. “85,170,255”).

See Also

[RasterMarkerSymbol.setShadowColor](#)

Example

```
var shadowColor = rasterMarkerSymbol.getShadowColor();
```

RasterMarkerSymbol.getSizeX

Description

Returns the width of the image in pixels.

Syntax

```
int getSizeX()
```

Arguments

None.

Returned Value

int The width of the image in pixels.

See Also

[RasterMarkerSymbol.setSizeX](#)

Example

```
var width = rasterMarkerSymbol.getSizeX();
```

RasterMarkerSymbol.getSizeY

Description

Returns the height of the image in pixels.

Syntax

```
int getSizeY()
```

Arguments

None

Returned Value

int The height of the image in pixels.

See Also

[RasterMarkerSymbol.setSizeY](#)

Example

```
var height = rasterMarkerSymbol.getSizeY();
```

RasterMarkerSymbol.getTransparency

Description:

Returns the current transparency of the Symbol. This value is a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax:

```
double getTransparency()
```

Arguments:

None

Returned Value:

double Value that transparency is set at. This value must be between 0.0 and 1.0, otherwise it is ignored.

See Also

[RasterMarkerSymbol.setTransparency](#)

Example

```
var transparency = rasterMarkerSymbol.getTransparency();
```

RasterMarkerSymbol.getURLString

Description

Returns the URL for the image as a String. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “`http://theworldsbestmapsite/images/symbol.jpg`”) and, when served, can be viewed by any client site. The image path is used to define images used for local projects only (e.g. “`C:\images\symbol.jpg`”).

Syntax

```
String getURLString()
```

Arguments

None.

Returned Value

String The URL for the image.

See Also

[RasterMarkerSymbol.setURLString](#)

Example

```
var url = rasterMarkerSymbol.getURLString();
```

RasterMarkerSymbol.setAntialiasing

Description

Set to define whether or not antialiasing should be used to produce this Symbol.

Syntax

```
boolean setAntialiasing(String value)
```

Arguments

value Use “true” if antialiasing should be used, “false”

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getAntialiasing](#)

Example

```
rasterMarkerSymbol.setAntialiasing("true");
```

RasterMarkerSymbol.setHotSpotX

Description

Sets the x-shift of the symbol from the feature it represents.

Syntax

```
boolean setHotSpotX(int xShift)
```

Arguments

xShift The x-shift of the symbol.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getHotSpotX](#)
[RasterMarkerSymbol.getHotSpotY](#)
[RasterMarkerSymbol.setHotSpotY](#)

Example

```
rasterMarkerSymbol.setHotSpotX(-2);
```

RasterMarkerSymbol.setHotSpotY

Description

Sets the y-shift of the symbol from the feature it represents.

Syntax

```
boolean setHotSpotY(int yShift)
```

Arguments

yShift The y-shift of the symbol.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getHotSpotX](#)
[RasterMarkerSymbol.getHotSpotY](#)
[RasterMarkerSymbol.setHotSpotX](#)

Example

```
rasterMarkerSymbol.setHotSpotY(5);
```

RasterMarkerSymbol.setImagePathString

Description

Sets the path to the image being used for this Symbol. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”) and thus can only be viewed on the local machine. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “http://theworldsbestmapsite/images/symbol.jpg”).

Syntax

```
boolean setImagePathString(String path)
```

Arguments

path The absolute path to the image used for this symbol.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getImagePathString](#)

Example

```
rasterMarkerSymbol.setImagePathString("C:\ArcIMS\images\fillImages\sites.jpg");
```

RasterMarkerSymbol.setShadowColor

Description

Sets the current color of the symbol shadow.

Syntax

```
boolean setShadowColor(Color newColor)
```

Arguments

newColor The new Color to be used for the shadow color.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getShadowColor](#)

Example

```
var shadowColor = imsMap.createColor(170,11,251);
rasterMarkerSymbol.setShadowColor(shadowColor);
```

RasterMarkerSymbol.setSizeX

Description

Sets the width of the image in pixels.

Syntax

```
boolean setSizeX(int width)
```

Arguments

width The width of the symbol image. Values less than 0 are ignored.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getSizeX](#)

Example

```
rasterMarkerSymbol.setSizeX(10);
```

RasterMarkerSymbol.setSizeY

Description

Sets the height of the image in pixels.

Syntax

```
boolean setSizeY(int height)
```

Arguments

height The height of the symbol image. Values less than 0 are ignored.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getSizeY](#)

Example

```
rasterMarkerSymbol.setSizeY(10);
```

RasterMarkerSymbol.setTransparency

Description

Sets the current transparency of the Symbol. This gives a mechanism to allow features behind a symbol be seen. The transparency is defined as a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax

```
boolean setTransparency(double transparency)
```

Arguments

transparency Value between 0.0 and 1.0 to set the transparency property to. Default is 1.0.

Returned Value

double True for success, false otherwise.

See Also

[RasterMarkerSymbol.getTransparency](#)

Example

```
rasterMarkerSymbol.setTransparency(.5);
```

RasterMarkerSymbol.setURLString

Description

Sets the URL for the symbol image as a String. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “<http://theworldsbestmapsite/images/symbol.jpg>”) and, when served, can be viewed by any client site. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”).

Syntax

```
boolean setURLString(String url)
```

Arguments

url The URL of the image.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterMarkerSymbol.getURLString](#)

Example

```
rasterMarkerSymbol.setURLString("http://www.esri.com/software/arcims/graphics/sudameri.gif");
```

RasterShieldSymbol

Description

The RasterShieldSymbol is used in adding roadway shield symbols using a raster image. The class includes methods for setting and getting of display properties for this symbol. The method `IMSMMap.createSymbol()` is used to create an object of this type.

See Also

[IMSMMap](#)
[RasterFillSymbol](#)
[RasterMarkerSymbol](#)
[RasterShieldSymbol](#)
[ShieldSymbol](#)

RasterShieldSymbol.getAntialiasing

Description

Returns true if antialiasing is used to produce this Symbol, false if it is not.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean	True if antialiasing is used on this Symbol, false otherwise.
---------	---

See Also

[RasterShieldSymbol.setAntialiasing](#)

Example

```
var isAA = rasterShieldSymbol.getAntialiasing();
```

RasterShieldSymbol.getBoundary

Description:

Returns true if this Symbol is drawn with a boundary, false if it is not. The boundary appears as a black border around the shield.

Syntax:

```
boolean getBoundary()
```

Arguments:

None.

Returned Value:

boolean	True if a boundary is drawn on this symbol, false otherwise.
---------	--

See Also

[RasterShieldSymbol.setBoundary](#)

Example

```
var isBoundaryDrawn = rasterShieldSymbol.getBoundary();
```

RasterShieldSymbol.getFontColor

Description

Returns the current font color as a String consisting of RGB values delimited by commas.

Syntax

```
String getFontColor()
```

Arguments

None.

Returned Value

String The RGB values of the font color delimited by commas (e.g. “12,210,130”).

See Also

[RasterShieldSymbol.setFontColor](#)

Example

```
var fontColor = rasterShieldSymbol.getFontColor();
```

RasterShieldSymbol.getFontName

Description

Returns the name of the current font as a String.

Syntax

`String getFontName()`

Arguments

None.

Returned Value

`String` The name of the font currently being used by this symbol

See Also

[RasterShieldSymbol.setFont](#)

Example

```
var fontName = rasterShieldSymbol.getFontName();
```

RasterShieldSymbol.getFontSize

Description

Returns the size of the current font as an integer.

Syntax

```
int getFontSize()
```

Arguments

None.

Returned Value

int	The point size of font currently being used.
-----	--

See Also

[RasterShieldSymbol.setFont](#)

Example

```
var fontSize = rasterShieldSymbol.getFontSize();
```

RasterShieldSymbol.getImagePathString

Description

Returns the path to the image being used for this symbol. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”) and thus can only be viewed on the local machine. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “http://theworldsbestmapsite/images/symbol.jpg”).

Syntax

String getImagePathString()

Arguments

None.

Returned Value

String The path of the image being used for this symbol.

See Also

[RasterShieldSymbol.setImagePathString](#)

Example

```
var imagePath = rasterShieldSymbol.getPathString();
```

RasterShieldSymbol.getLabelMode

Description

Returns the current value of the label mode property.

Syntax

```
int getLabelMode()
```

Arguments

None.

Returned Value

int The value associated with the current mode (See RasterShieldSymbol.setLabelMode for a list of known values).

See Also

[RasterShieldSymbol.setLabelMode](#)

Example

```
var labelMode = rasterShieldSymbol.getLabelMode();
```

RasterShieldSymbol.getPrintMode

Description

Returns the current value of the print mode property.

Syntax

```
int getPrintMode()
```

Arguments

None

Returned Value

int The value associated with the current mode (See RasterShieldSymbol.setPrintMode for a list of known values).

See Also

[RasterShieldSymbol.setPrintMode](#)

Example

```
var printMode = rasterShieldSymbol.getPrintMode();
```

RasterShieldSymbol.getShadowColor

Description

Returns the current color of the symbol shadow as a String consisting of RGB values delimited by commas.

Syntax

```
String getShadowColor()
```

Arguments

None

Returned Value

String The RGB values of the shadow color delimited by commas (e.g. “85,170,255”).

See Also

[RasterShieldSymbol.setShadowColor](#)

Example

```
var shadowColor = rasterShieldSymbol.getShadowColor();
```

RasterShieldSymbol.getTextPositionX

Description

Returns the x coordinate of the position of the Symbol text

Syntax

```
int getTextPositionX()
```

Arguments

None

Returned Value

int The x coordinate of the text.

See Also

[RasterShieldSymbol.setTextPositionX](#)

Example

```
var textXPos = rasterShieldSymbol.getTextPositionX();
```

RasterShieldSymbol.getTextPositionY

Description

Returns the y coordinate of the position of the Symbol text

Syntax

```
int getTextPositionY()
```

Arguments

None

Returned Value

int The y coordinate of the text.

See Also

[RasterShieldSymbol.setTextPositionY](#)

Example

```
var textYPos = rasterShieldSymbol.getTextPositionY();
```

RasterShieldSymbol.getTransparency

Description

Returns the current transparency of the Symbol. This value is a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

double	The current transparency value of the Symbol. This value must be between 0.0 and 1.0, otherwise it is ignored.
--------	--

See Also

[RasterShieldSymbol.setTransparency](#)

Example

```
var transparency = rasterShieldSymbol.getTransparency();
```

RasterShieldSymbol.getURLString

Description

Returns the URL for the image as a String. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “`http://theworldsbestmapsite/images/symbol.jpg`”) and, when served, can be viewed by any client site. The image path is used to define images used for local projects only (e.g. “`C:\images\symbol.jpg`”).

Syntax

```
String getURLString()
```

Arguments

None

Returned Value

String The URL for the image.

See Also

[RasterShieldSymbol.setURLString](#)

Example

```
var url = rasterShieldSymbol.getURLString();
```

RasterShieldSymbol.setAntialiasing

Description

Set to define whether or not antialiasing should be used to produce this Symbol.

Syntax

```
boolean setAntialiasing(String value)
```

Arguments

value Use “true” if antialiasing should be used, “false” otherwise.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getAntialiasing](#)

Example

```
rasterShieldSymbol.setAntialiasing("true");
```

RasterShieldSymbol.setBoundary

Description

Sets the boundary property for the shield. If set to true, a boundary is drawn as a black border around the shield. The default value is false.

Syntax

```
boolean setBoundary(String value)
```

Arguments

value “true” if a boundary should be drawn, “false” otherwise.

Returned Value

boolean True if successful, false otherwise.

See Also

[RasterShieldSymbol.getBoundary](#)

Example

```
rasterShieldSymbol.setBoundary("true");
```

RasterShieldSymbol.setFontColor

Description

Sets the current font Color.

Syntax

```
boolean setFontColor(Color color)
```

Arguments

color The new Color of the current font.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getFontColor](#)
[Color](#)

Example

```
var color = imsMap.createColor(24,32,68);  
rasterShieldSymbol.setFontColor(color);
```

RasterShieldSymbol.setFont

Description

Sets the current font style by its name and its size. (See documentation for `java.awt.Font` for a list of all the acceptable font constants.)

Syntax

`Boolean setFont(String fontName, int size)`

Arguments

`fontName` The name of the font to be used.
`size` The point size of the font to be used.

Returned Value

`boolean` True for success, false for failure.

See Also

`RasterShieldSymbol.getFontName`
`RasterShieldSymbol.getFontSize`

Example

```
rasterShieldSymbol.setFont(java.awt.Font.TRUETYPE_FONT, 10);
```

RasterShieldSymbol.setFontColor

Description

Sets the current font color using a Color object.

Syntax

```
boolean setFontColor(Color color)
```

Arguments

color The new Color for the Symbol font.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getFontColor](#)

Example

```
var fontColor = imsMap.createColor(24, 32, 109);
rasterShieldSymbol.setFontColor(fontColor);
```

RasterShieldSymbol.setImagePathString

Description

Sets the path to the image being used for this Symbol. The image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”) and thus can only be viewed on the local machine. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “http://theworldsbestmapsite/images/symbol.jpg”).

Syntax

```
boolean setImagePathString(String path)
```

Arguments

path The absolute path to the image used for this symbol.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getImagePathString](#)

Example

```
rasterShieldSymbol.setImagePathString("C:\ArcIMS\images\fillImages\sitesz.jpg");
```

RasterShieldSymbol.setLabelMode

Description:

Label mode affects the content of values displayed. You can choose to show the full value as it is found in the feature (full name) or you can choose to have just the number portion taken from the value (numeric only). The default is for full field values to be displayed.

Syntax

```
boolean setLabelMode(int mode)
```

Arguments

mode	The mode which the labels are to be printed. Unrecognized values have no effect. The acceptable values are:
0	full label
1	numeric only (e.g. if the value used is "I15", only "15" is displayed)

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getLabelMode](#)

Example

```
rasterShieldSymbol.setLabelMode(1);
```

RasterShieldSymbol.setPrintMode

Description

Print mode affects the display of labels. You can choose to display the labels as default, all upper case, all lower case, and pretty (which displays the first letter of each word in upper case and the rest in lower case).

Syntax

```
boolean setPrintMode(int mode)
```

Arguments

mode	The mode which the labels are to be printed. Unrecognized values have no effect. The acceptable values are:
0	default
1	pretty mode (first letter of each word is upper case, the rest are lower case)
2	all upper case
3	all lower case

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getPrintMode](#)

Example

```
rasterShieldSymbol.setPrintMode(3);
```

RasterShieldSymbol.setShadowColor

Description

Sets the current color of the symbol shadow.

Syntax

```
boolean setShadowColor(Color newColor)
```

Arguments

newColor The new Color to be used for the shadow color.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getShadowColor](#)

Example

```
var shadowColor = imsMap.createColor(170,11,251);
rasterShieldSymbol.setShadowColor(shadowColor);
```

RasterShieldSymbol.setTextPositionX

Description

Sets the x coordinate of the text position for the Symbol.

Syntax

```
boolean setTextPositionX(int x)
```

Arguments

x The x coordinate of the text position.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getTextPositionX](#)

Example

```
rasterShieldSymbol.setTextPositionX(-1);
```

RasterShieldSymbol.setTextPositionY

Description

Sets the y coordinate of the text position for the Symbol.

Syntax

```
boolean setTextPositionY(int y)
```

Arguments

y The y coordinate of the text position.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getTextPositionY](#)

Example

```
rasterShieldSymbol.setTextPositionY(3);
```

RasterShieldSymbol.setTransparency

Description

Sets the current transparency of the Symbol. This gives a mechanism to allow features behind a symbol be seen. The transparency is defined as a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax

```
boolean setTransparency(double transparency)
```

Arguments

transparency Value between 0.0 and 1.0 to set the transparency property to. Default is 1.0.

Returned Value

double True for success, false otherwise.

See Also

[RasterShieldSymbol.getTransparency](#)

Example

```
rasterShieldSymbol.setTransparency(.5);
```

RasterShieldSymbol.setURLString

Description

Sets the URL for the symbol image as a String. The URL path is used by the spatial server to define the location of an image for a project being served (e.g. “<http://theworldsbestmapsite/images/symbol.jpg>”) and, when served, can be viewed by any client site. While the image path is used to define images used for local projects only (e.g. “C:\images\symbol.jpg”).

Syntax

```
boolean setURLString(String url)
```

Arguments

url The URL of the image.

Returned Value

boolean True for success, false otherwise.

See Also

[RasterShieldSymbol.getURLString](#)

Example

```
rasterShieldSymbol.setURLString("http://www.esri.com/software/arcims/graphics/sudameri.gif");
```

Renderer

The Renderer class is used within various IMSMap methods which can be appropriately applied to all types of renderer. For example IMSMap.createRenderer returns a Renderer for the specified string type. Also, IMSMap.setLayerRenderer assigns to the Layer any kind of Renderer that you specify.

The renderer object is an abstract class used by all renderers in the Java Viewer Object Model. An abstract class is a Java class that can only be sub classed using the Java Language — it cannot be instantiated, therefore it is not a creatable object for use in a web browser via a scripting language. For a description of abstract classes in Java see the Java 2 Documentation at <http://java.sun.com/docs>.

See Also

[ValueMapRenderer](#)
[GroupRenderer](#)
[ValueMapLabelRenderer](#)
[LabelRenderer](#)
[ScaleDependentRenderer](#)
[SimpleRenderer](#)

ScaleDependentRenderer extends Renderer

A ScaleDependentRenderer represents a way to display a Renderer within a certain scale range. A ScaleDependentRenderer is a Renderer that contains (wraps) another Renderer. The rendering of the features or text is delegated to the wrapped Renderer, only when the current Display scale is within a certain scale range. If you want to have multiple scale ranges, then you can group a number of ScaleDependentRenderers using the GroupRenderer.

ScaleDependentRenderer.getMaximumScale

Description

Returns the maximum scale at which this renderer is displayed. If the current scale value is greater than the maximum scale, the renderer will not be shown.

Syntax

```
long getMaximumScale()
```

Arguments

None

Returned Value

long any valid number to be used in determining if it is within the scale range specified

See Also

ArcXML Programmer's Reference

Example

```
var srend = imsmap.getLayerRenderer(myLayer);
var maxscale = srend.getMaximumScale();
```

ScaleDependentRenderer.getMinimumScale

Description

Returns the minimum scale at which this renderer is displayed. If the current scale value is less than the minimum scale, the renderer will not be shown.

Syntax

```
long getMinimumScale()
```

Arguments

None

Returned Value

long any valid number to be used in determining if it is within the scale range specified

See Also

ArcXML Programmer's Reference

Example

```
var srend = imsmap.getLayerRenderer(myLayer);
var minscale = srend.getMinimumScale();
```

ScaleDependentRenderer.inRange

Description

The inRange method is used to determine if the specified scale is between the minimum and maximum value. The comparison of the value is made based on the Java Interface java.lang.Comparable. Refer to the Java Developer kit documentation for the description

Syntax

boolean inRange(Long value)

Arguments

Long any valid number to be used in determining if it is within the scale range specified

Returned Value

boolean returns true if successful, false otherwise

See Also

ArcXML Programmer's Reference

Example

```
var srend = imsmap.getLayerRenderer(myLayer);
if( srend.inRange("1,500,000") ) {
    alert("Is in range");
}
```

ScaleDependentRenderer.setMaximumScale

Description

Set the maximum scale range to display the features using a Renderer.

Syntax

```
boolean setMaximumScale(long value)
```

Arguments

Long value – a valid number representing the Maximum Scale range to use for rendering the features of the layer using the Renderer specified.

Returned Value

boolean returns true if successful, false otherwise

See Also

ArcXML Programmer's Reference

Example

```
var srend = imsmap.createRenderer("SCALE_DEPENDENT_RENDERER");
var simprend = imsmap.createRenderer("SIMPLE_RENDERER");
srend.setRenderer(simprend);
srend.setMaximumScale("2,000,000");
```

ScaleDependentRenderer.setMinimumScale

Description

Set the minimum scale range to display the features using a Renderer.

Syntax

boolean setMinimumScale(long value)

Arguments

Long value – a valid number representing the Minimum Scale range to use for rendering the features of the layer using the Renderer specified.

Returned Value

boolean returns true if successful, false otherwise

See Also

ArcXML Programmer's Reference

Example

```
var rend = imsmap.createRenderer("SCALE_DEPENDENT_RENDERER");
var simprend = imsmap.createRenderer("SIMPLE_RENDERER");
rend.setRenderer(simprend);
rend.setMinimumScale("1,000,000");
```

ScaleDependentRenderer.setRenderer

Description

The setRenderer method provides a way to provide a way to wrap a Renderer into this ScaleDependentRenderer.

Syntax

```
boolean setRenderer(Renderer rend)
```

Arguments

Renderer	rend	any valid object that represents a Renderer (i.e. SimpleRenderer, ValueMapRenderer, LabelRenderer)
----------	------	---

Returned Value

boolean	returns true if the method was successful, false otherwise
---------	--

See Also

[IMSSMap](#)

Example

```
var srend = imsmap.createRenderer("SCALE_DEPENDENT_RENDERER");
var simprend = imsmap.createRenderer("SIMPLE_RENDERER");
srend.setRenderer(simprend);
```

ShieldSymbol

Description

The ShieldSymbol is used in adding simple roadway shield symbols. The class includes methods for setting and getting of display properties for this symbol. The ShieldSymbol.setType() method lets a developer choose from five shield options—Interstate, USRoads, Rectangle, Oval, Mexican—see ShieldSymbol.setType(). The method IMSMap.createSymbol() is used to create an object of this type.

See Also

[IMSMap](#)
[ShieldSymbol](#)

ShieldSymbol.getAntialiasing

Description

Returns true if antialiasing is used to produce this Symbol, false if it is not.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean	True if antialiasing is used on this Symbol, false otherwise.
---------	---

See Also

[ShieldSymbol.setAntialiasing](#)

Example

```
var isAA = shieldSymbol.getAntialiasing();
```

ShieldSymbol.getFontColor

Description

Returns the current name of the font color as a String consisting of RGB values delimited by commas.

Syntax

`String getFontColor()`

Arguments

None

Returned Value

`String` The RGB values of the font color delimited by commas (e.g. “12,210,130”).

See Also

`ShieldSymbol.setFontColor`

Example

```
var fontColor = shieldSymbol.getFontColor();
```

ShieldSymbol.getFontName

Description

Returns the name of the current font as a String.

Syntax

```
String getFontName()
```

Arguments

None

Returned Value

String The name of the font currently being used by this symbol

See Also

[ShieldSymbol.setFont](#)

Example

```
var fontName = shieldSymbol.getFontName();
```

ShieldSymbol.getFontSize

Description

Returns the size of the current font as an integer.

Syntax

`String getFontSize()`

Arguments

None

Returned Value

`int` The point size of font currently being used.

See Also

`ShieldSymbol.setFont`

Example

```
var fontSize = shieldSymbol.getFontSize();
```

ShieldSymbol.getLabelMode

Description

Returns the current value of the label mode property.

Syntax

```
int getLabelMode()
```

Arguments

None

Returned Value

int The value associated with the current mode (See ShieldSymbol.setLabelMode for a list of known values).

See Also

[ShieldSymbol.setLabelMode](#)

Example

```
var labelMode = shieldSymbol.getLabelMode();
```

ShieldSymbol.getMinSize

Description

Returns the current minimum size for this ShieldSymbol. Default value is 0.

Syntax

`int getMinSze()`

Note: The current method syntax has an incorrect spelling of the name. This will be corrected in a future version of ArcIMS to use “Size”.

Arguments

None

Returned Value

`int` The current minimum size of the Symbol.

See Also

[ShieldSymbol.setMinSize](#)

Example

```
var minShieldSize = shieldSymbol.getMinSize();
```

ShieldSymbol.getShadowColor

Description

Returns the current color of the symbol shadow as a String consisting of RGB values delimited by commas.

Syntax

```
String getShadowColor()
```

Arguments

None

Returned Value

String The RGB values of the shadow color delimited by commas (e.g. “85,170,255”).

See Also

[ShieldSymbol.setShadowColor](#)

Example

```
var shadowColor = shieldSymbol.getShadowColor();
```

ShieldSymbol.getTransparency

Description

Returns the current transparency of the Symbol. This value is a double between 0.0 (totally transparent) and 1.0 (totally opaque). The default transparency is set to 1.0.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

double The current transparency value of the Symbol. This value must be between 0.0 and 1.0, otherwise it is ignored.

See Also

[ShieldSymbol.setTransparency](#)

Example

```
var transparency = shieldSymbol.getTransparency();
```

ShieldSymbol.getType

Description

Returns an integer corresponding to the type of shield being used to in this Symbol.

Syntax

```
int getType()
```

Arguments

None

Returned Value

int Integer corresponding to a shield type. (See ShieldSymbol.setType() for a list of shields and their corresponding values.)

See Also

[ShieldSymbol.setType](#)

Example

```
var shieldType = shieldSymbol.getType();
```

ShieldSymbol.setAntialiasing

Description

Set to define whether or not antialiasing should be used to produce this Symbol.

Syntax

```
boolean setAntialiasing(String value)
```

Arguments

value Use “true” if antialiasing should be used, “false” otherwise.

Returned Value

boolean True for success, false otherwise.

See Also

[ShieldSymbol.getAntialiasing](#)

Example

```
shieldSymbol.setAntialiasing("true");
fontName, int size)
```

ShieldSymbol.setFont

Description

Sets the current font style by its name and its size. (See documentation for `java.awt.Font` for a list of all the acceptable font constants.)

Syntax

`Boolean setFont(String`

Arguments

`fontName` The name of the font to be used.
`size` The point size of the font to be used.

Returned Value

`boolean` True for success, false for failure.

See Also

`ShieldSymbol.getFontName`
`ShieldSymbol.getFontSize`

Example

```
shieldSymbol.setFont(java.awt.Font.TRUETYPE_FONT, 10);
```

ShieldSymbol.setFontColor

Description

Sets the current font Color.

Syntax

```
boolean setFontColor(Color color)
```

Arguments

color The new Color of the current font.

Returned Value

boolean True for success, false otherwise.

See Also

[ShieldSymbol.getFontColor](#)
[Color](#)

Example

```
var color = imsMap.createColor(24,32,68);
shieldSymbol.setFontColor(color);
```

ShieldSymbol.setLabelMode

Description

Label mode affects the content of values displayed. You can choose to show the full value as it is found in the feature (full name) or you can choose to have just the number portion taken from the value (numeric only). The default is for full field values to be displayed.

Syntax

```
boolean setLabelMode(int mode)
```

Arguments

mode	The mode which the labels are to be printed. Unrecognized values have no effect. The acceptable values are:
0	full label
1	numeric only (e.g. if the value used is "I15", only "15" is displayed)

Returned Value

boolean	True for success, false otherwise.
---------	------------------------------------

See Also

[ShieldSymbol.getLabelMode](#)

Example

```
shieldSymbol.setLabelMode(1);
```

ShieldSymbol.setMinSize

Description

Sets the minimum size for this ShieldSymbol. Default value is 0.

Syntax

boolean setMinSize(int newMin)

Arguments

newMin The new minimum size for the Symbol.

Returned Value

boolean True for success, false otherwise.

See Also

[ShieldSymbol.getMinSize](#)

Example

```
shieldSymbol.setMinSize(5);
```

ShieldSymbol.setShadowColor

Description

Sets the current color of the symbol shadow.

Syntax

```
boolean setShadowColor(Color newColor)
```

Arguments

newColor The new Color to be used for the shadow color.

Returned Value:

boolean True for success, false otherwise.

See Also

[ShieldSymbol.getShadowColor](#)

Example

```
var shadowColor = imsMap.createColor(170,11,251);
shieldSymbol.setShadowColor(shadowColor);
```

ShieldSymbol.setTransparency

Description

Sets the current transparency of the Symbol. This gives a mechanism to allow features behind a symbol be seen.

Syntax

```
boolean setTransparency(double transparency)
```

Arguments

transparency Value between 0.0 and 1.0 to set the transparency property to. Default is 1.0.

Returned Value

boolean True for success, false otherwise.

See Also

[ShieldSymbol.getTransparency](#)

Example

```
shieldSymbol.setTransparency(.5);
```

ShieldSymbol.setType

Description

Sets the type of shield to be used to display the Symbol.

Syntax

```
boolean setType(int type)
```

Arguments

Type The type of shield to be used. Unknown values will have no effect. The known values are the following:

- | | |
|---|-----------------|
| 0 | Interstate |
| 1 | US Road |
| 2 | Rectangle |
| 3 | Oval |
| 4 | Mexican highway |

Returned Value

boolean True for success, false otherwise.

See Also

[ShieldSymbol.getType](#)

Example

```
shieldSymbol.setType(4);
```

SimpleRenderer extends Renderer

A Simple Renderer is used to draw one type of Feature (Point, Line or Polygon) using a Symbol. Using multiple SimpleRenderers and the GroupRenderer, common cartographic tasks such as cased lines become very simple.

SimpleRenderer.setSymbol

Description

A Symbol object consists of attributes that control how a feature is displayed. Use the setSymbol method to set the symbol that you want to use to display the features of the Layer.

Syntax

```
boolean setSymbol(Symbol symObject)
```

Arguments

Symbol symObject – The symbol to use in rendering a feature

Returned Value

boolean returns true if the method was successful, false otherwise

See Also

Renderer

Symbol

IMSMMap

Example

The example on the next page demonstrates creating cased lines using a group renderer and SimpleRenderer on a Line Layer.

SimpleRenderer.setSymbol

Example

```
var sym1, sym2, simple1, simple2, group ;
group = imsmap.createRenderer("GROUP_RENDERER");
sym1 = imsmap.createSymbol("LINE_SYMBOL");
sym2 = imsmap.createSymbol("LINE_SYMBOL");
simple1 = imsmap.createRenderer("SIMPLE_RENDERER");
simple2 = imsmap.createRenderer("SIMPLE_RENDERER");
sym1.setAntialiasing("true");
sym1.setWidth(3);
sym1.setColor(imsmap.createColor(255,0,0));
simple1.setSymbol(sym1);
sym2.setAntialiasing("true");
sym2.setWidth(1);
sym2.setColor(imsmap.createColor(255,255,0));
simple2.setSymbol(sym2);
group.addRenderer(simple1);
group.addRenderer(simple2);
imsmap.setLayerValueMapRenderer (myLayer, group );
```

Symbol

Description

The Symbol class is used within various IMSMap methods which can appropriately be applied to all types of renderer. For example IMSMap.createSymbol returns a Symbol for the specified string type. Also, IMSMap.getRendererSymbol() returns the Symbol for a given Renderer and a value (when applicable).

The Symbol class is the abstract super-class for all other symbols in the Java Viewer Object Model. Symbol is used only to generalize and defines no functionality or fields of its own. An abstract class is a Java class that can only be sub classed using the Java Language — it cannot be instantiated, therefore it is not a creatable object for use in a web browser via a scripting language. For a description of abstract classes in Java see the Java 2 Documentation at <http://java.sun.com/docs>.

See Also

CalloutMarkerSymbol
FillSymbol
GradientFillSymbol
HashLineSymbol
IMSMap
LineSymbol
MarkerSymbol
PolygonSymbol
RasterFillSymbol
RasterMarkerSymbol
RasterShieldSymbol
ShieldSymbol
TextSymbol
TrueTypeMarkerSymbol

TextSymbol extends Symbol

A TextSymbol object consists of attributes that control how text is rendered. You can set the font associated with the TextSymbol using the setFont method and its color with the setFontColor method. A TextSymbol object also allows for special effects such as Glowing, Shadows, Anti-Aliasing, and Blockout. The TextSymbol object is used with the LabelRenderers.

TextSymbol.getAntialiasing

Description

Retrieves the current antialiasing value of the text symbol object. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean	true if antialiasing is to be used, false otherwise
---------	---

See Also

[Symbol](#)

Example

```
private TextSymbol sym = (TextSymbol)renderer.getSymbol();
boolean val = sym.getAntialiasing();
```

TextSymbol.getBlockoutColor

Description

Retrieves the current color value used by the Blockout effect of the text symbol object

Syntax

String getBlockoutColor()

Arguments

None

Returned Value

- A comma delimited string value (R, G, B):
- R defines R-part of the RGB color value, should be between 0 and 255
- G defines G-part of the RGB color value, should be between 0 and 255
- B defines B-part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym = imsmap.createColor("TEXT_SYMBOL");
var rgb = sym.getBlockoutColor();
```

TextSymbol.getColor

Description

Retrieves the current shadow color as a comma delimited string containing the red, green, and blue color values.

Syntax

```
String getColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines the Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym = imsmmap.createSymbol("TEXT_SYMBOL");
sym.setColor(imsmmap.createColor(255,0,0));
var rgb = sym.getColor();
// rgb = 255,0,0
```

TextSymbol.getFontColor

Description

The getFontColor returns the current color used to draw the font of the label.

Syntax

String getFontColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines R-part of the RGB color value, should be between 0 and 255
G defines G-part of the RGB color value, should be between 0 and 255
B defines B-part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var rgb = sym.getFontColor();
```

TextSymbol.getFontName

Description

Retrieves the current name of the font used by the text symbol object

Syntax

`String getFontName()`

Arguments

None

Returned Value

`String` the name of the font used

See Also

[IMSMMap](#)
[Symbol](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var fontName = sym.getFontName();
```

TextSymbol.getFontSize

Description

Retrieves the current size of the font use by the text symbol object

Syntax

int getFontSize()

Arguments

None

Returned Value

int the size of the font specified

See Also

Symbol

Example

```
private TextSymbol sym = (TextSymbol)renderer.getSymbol();
int fontSize = sym.getFontSize();
```

TextSymbol.getGlowColor

Description

Gets the current color value used for the glowing effect of the object

Syntax

String getGlowColor()

Arguments

None

Returned Value

- A comma delimited string value (R, G, B):
- R defines R-part of the RGB color value, should be between 0 and 255
- G defines G-part of the RGB color value, should be between 0 and 255
- B defines B-part of the RGB color value, should be between 0 and 255

See Also

[IMSSMap](#)
[Symbol](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var rgb = sym.getGlowColor();
```

TextSymbol.getInterval

Description

Sets the interval of the TextSymbol.

Syntax

```
double getInterval()
```

Arguments

None

Returned Value

double a valid double value

See Also

[IMSMMap](#)
[Symbol](#)

Example

```
var sym = imsmap.createSymbol(":TEXT_SYMBOL");
var interval = sym.getInterval();
```

TextSymbol.getOutlineColor

Description

Gets the current color for the Outline effect of the symbol.

Syntax

```
String getOutlineColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var rgb = sym.getOutlineColor();
```

TextSymbol.getPrintMode

Description

Retrieves the PrintMode of the Symbol object as it was set in the setPrintMode. The default value for PrintMode is 0, none.

Syntax

```
int getPrintMode()
```

Arguments

None

Returned Value

int	0 - None: No change is made to the label
	1 - Proper Case: The first letter of each word in a label is upper case and everything else is lower case
	2 - Upper Case: All letters are upper case
	3 - Lower Case: All letters are lower case

See Also

[Symbol](#)

[ArcXML Programmer's Reference](#)

Example

This sample demonstrates setting the printmode to always use UpperCase

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var ret = sym.getPrintMode(2);
```

TextSymbol.getTransparency

Description

The getTransparency method retrieves the current transparency value set on the TextSymbol object. The default value is 1.0 with a valid range of 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

double a double primitive type with values ranging from 0.0 (Transparent) to 1.0 (Opaque)

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var transparency = sym.getTransparency();
if( var == 0 ){
    alert("Transparent Symbol");
} else {
    alert("Some level of opaque");
}
```

TextSymbol.setAntialiasing

Description

Sets the current antialiasing value used by the Text symbol object. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother.

Syntax

boolean setAntialiasing(String)

Arguments

String use “true” to enable this option, “false” otherwise

Returned Value

boolean returns true if the method was successful, false otherwise

See Also

Symbol

Example

```
private TextSymbol sym = (TextSymbol)renderer.getSymbol();
boolean ret = sym.getAntialiasing();
```

TextSymbol.setBlockoutColor

Description

Sets the current color value used by the Blockout effect of the text symbol object

Syntax

```
boolean setBlockoutColor(java.awt.Color)
```

Arguments

Color any valid color object created using the imsmap helper method, createColor(r,g,b)

Returned Value

none

See Also

[Symbol](#)
[Color](#)
[IMSMMap](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var ret = sym.setBlockoutColor(imsmap.createColor(255,255,255));
```

TextSymbol.setFont

Description

Sets the font object used by the Text symbol object to draw the text for the labels.

Syntax

```
boolean setFont(String fontname, int fontsize)
```

Arguments

String fontname - the name of the font to use
int fontsize - the size of the font to use

Returned Value

boolean true if the method was successful in setting the font, false otherwise

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var rend = imsmap.getLayerRenderer(myLayer);
var sym = rend.getSymbol();
if( !sym.setFont("times", 12) ) {
    //report the failure - do something...
```

TextSymbol.setFontColor

Description

The setFontColor method is used to set the Color of the font drawn for the label of the feature.

Syntax

```
boolean setFontColor(Color)
```

Arguments

Color any valid Color object created with the imsmap createColor helper method

Returned Value

boolean returns true if the method was set successfully, false otherwise.

See Also

Symbol
Color
IMSMMap

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var ret = sym.setFontColor(imsmap.createColor(255,0,0));
```

TextSymbol.setGlowColor

Description

Sets the color used for the Glowing effect.

Syntax

boolean setGlowColor(Color color)

Arguments

Color a valid Color object obtained from the IMSMap.createColor(r,g,b) helper method

Returned Value

boolean returns true if the method was successful, false otherwise.

See Also

Symbol

Color

IMSMMap

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var ret = sym.setGlowColor(imsmap.createColor(255,0,0));
```

TextSymbol.setInterval

Description

Sets the interval value of the object used by the label engine.

Syntax

```
boolean setInterval(double n)
```

Arguments

double a valid double value

Returned Value

boolean True if the object was successful, false otherwise.

See Also

[Symbol](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var ret = sym.setInterval(1.0);
```

TextSymbol.setOutlineColor

Description

Sets the outline color of this object.

Syntax

```
boolean setOutlineColor(java.awt.Color colorValue)
```

Arguments

Color a valid java color object

Returned Value

True if the object succeeded, false otherwise

See Also

Symbol

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
var ret = sym.setOutlineColor(imsmap.createColor(255,0,0));
```

TextSymbol.setPrintMode

Description

The PrintMode of a Symbol controls how the text of a label is displayed.

Syntax

```
boolean setPrintMode(int value)
```

Arguments

int	0 - None: No change is made to the label 1 – Title Caps: Also known as Proper case where the first letter of each word in a label is upper case and everything else is lower case 2 - UpperCase: All letters are upper case 3 - LowerCase: All letters are lower case
-----	--

Returned Value

boolean returns true if the method succeeded, false otherwise

See Also

[Symbol](#)
[ArcXML Programmer's Reference](#)
[IMSMMap](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
if( sym.setPrintMode(1) ) alert("Success");
```

TextSymbol.setShadow

Description

The shadow effect methods provide for a mirrored image or reflection to appear behind and below the text as it is labeled. Use this method to set the color value for the shadow effect on the TextSymbol. The shadow will be drawn using a 0.5 transparency.

Syntax

```
boolean setShadow(Color colorValue)
```

Arguments

Color	colorValue - a valid color value retrieved from the IMSMap.createColor() helper method
-------	--

Returned Value

boolean	returns true if the method was successful in setting the color value, false otherwise.
---------	--

See Also

Symbol
Color
IMSMMap

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
sym.setShadow(imsmap.createColor(255,0,0));
```

TextSymbol.setTransparency

Description

The setTransparency method allows you to set the level of transparency on the text. Valid values are 0.0 (Transparent) to 1.0 (Opaque).

Syntax

```
void setTransparency(double n)
```

Arguments

double	n - A valid number in the range of 0.0 to 1.0. The value 1.0 will produce completely opaque text, 0.0 will produce completely transparent text.
--------	---

Returned Value

None

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym = imsmap.createSymbol("TEXT_SYMBOL");
sym.setTransparency(1.0);
```

TrueTypeMarkerSymbol extends Symbol

A TrueTypeMarkerSymbol defines the characteristics of labels associated with a Map Layer using a TrueType font. Using a TrueType font allows you to use scaleable vector fonts which can be scaled to any size and otherwise transformed more easily than a bitmap font, and with more attractive results, though this requires a lot of numerical processing. The result of transforming a character in a Vector font in a particular way is often saved as a bitmap in a font cache to avoid repeating the calculations if that character is to be drawn again. As with the TextSymbol, A TrueTypeMarkerSymbol also allows for special effects such as Glowing, Shadows, Anti-Aliasing, and Blockout

TrueTypeMarkerSymbol.getAngle

Description

Retrieves the angle of rotation indicated by the setAngle method. The angle of rotation is measured in degrees where 0 represents the top and 180 is the bottom working counterclockwise.

Syntax

double getAngle()

Arguments

None

Returned Value

double valid values will be in the range 0.0 – 360.0, the default value is 0.

See Also

[Symbol](#)
[ArcXML Programmer's Reference](#)
[IMSSMap](#)

Example

```
var sym imsmap.createSymbol("TRUEETYPE_MARKER_SYMBOL");
var angle = sym.getAngle();
```

TrueTypeMarkerSymbol.getAntialiasing

Description

Retrieves the current antialiasing value of the text symbol object. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother.

Syntax

```
boolean getAntialiasing()
```

Arguments

None

Returned Value

boolean true if antialiasing is to be used, false otherwise

See Also

Symbol

Example

```
var sym = imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
boolean val = sym.getAntialiasing();
```

TrueTypeMarkerSymbol.getBlockoutColor

Description

Retrieves the current color value used by the Blockout effect of the text symbol object

Syntax

String getBlockoutColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym imsmmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var rgb = sym.getBlockoutColor();
```

TrueTypeMarkerSymbol.getCharacter

Description

Returns the character code in the font associated with the TrueTypeMarkerSymbol.

Syntax

String getCharachter()

Note: The current method syntax has an incorrect spelling of the name. This will be corrected in a future version of ArcIMS to use “Character”.

Arguments

None

Returned Value

String the string representation of the character

See Also

Symbol
IMSSMap

Example

```
var sym = imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var char = sym.getCharacter();
alert("The character represented in the TrueTypemarker is: " + char);
```

TrueTypeMarkerSymbol.getFontColor

Description

Sets the color used to draw the font.

Syntax

```
String getFontColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var rgb = sym.getFontColor();
```

TrueTypeMarkerSymbol.getFontName

Description

Retrieves the current name of the font used by the text symbol object

Syntax

String getFontName()

Arguments

None

Returned Value

String the name of the font used

See Also

Symbol
IMSMMap

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var fontName = sym.getFontName();
alert("The font name is: " + fontName);
```

TrueTypeMarkerSymbol.getFontSize

Description

Retrieves the current size of the font use by the text symbol object

Syntax

```
int getFontSize()
```

Arguments

None

Returned Value

int the size of the font specified

See Also

Symbol

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
int fontSize = sym.getFontSize();
```

TrueTypeMarkerSymbol.getGlowColor

Description

Retrieves the current Color value used for the glowing effect of the object

Syntax

String getGlowColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSMMap](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var rgb = sym.getGlowColor();
```

TrueTypeMarkerSymbol.getOutlineColor

Description

Gets the current color for the Outline effect of the symbol.

Syntax

```
String getOutlineColor()
```

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

[Symbol](#)
[IMSSMap](#)
[Color](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var rgb = sym.getOutlineColor();
```

TrueTypeMarkerSymbol.getColor

Description

Retrieves the current shadow color as a comma delimited string containing the red green and blue color values. Please note that this method is incorrectly named and will be corrected in a future version of ArcIMS.

Syntax

String getColor()

Arguments

None

Returned Value

A comma delimited string value (R, G, B):
R defines Red part of the RGB color value, should be between 0 and 255
G defines Green part of the RGB color value, should be between 0 and 255
B defines Blue part of the RGB color value, should be between 0 and 255

See Also

Symbol IMSMap

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
String rgb = sym.getColor();
String r = "", g = "", b = "";
java.util.StringTokenizer tok = new java.util.StringTokenizer("1,2,3", ",");
if ( tok.hasMoreTokens() ) {
    r = tok.nextToken();
    g = tok.nextToken();
    b = tok.nextToken();
}
System.out.println(r + ":" + g + ":" + b );
```

TrueTypeMarkerSymbol.getTransparency

Description

The getTransparency method retrieves the current transparency value set on the TextSymbol object. The default value is 1.0 with a valid range of 0.0 (Transparent) to 1.0 (Opaque) and any valid double type number in-between.

Syntax

```
double getTransparency()
```

Arguments

None

Returned Value

Double value ranging from 0.0 to 1.0

See Also

[Symbol](#)

Example

```
var sym = imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var transparency = sym.getTransparency();
```

TrueTypeMarkerSymbol.setAngle

Description

Sets the angle of rotation applied to the text. The angle of rotation is measured in degrees where 0 represents the top and 180 the bottom working counterclockwise.

Syntax

boolean setAngle(double)

Arguments

double a valid double number in the range of 0.0 – 360.0, the default value is 0.

Returned Value

boolean returns true if the object was successful, false otherwise.

See Also

Symbol

ArcXML Programmer's Reference

IMSSMap

Example

The following example demonstrates setting the rotation angle at 240 degrees.

```
var sym = imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret = sym.setAngle(240);
```

TrueTypeMarkerSymbol.setCharacter

Description

Sets the character code in the font associated with the TrueTypeMarkerSymbol.

Syntax

boolean setChrachter(String)

Note: The current method syntax has an incorrect spelling of the name. This will be corrected in a future version of ArcIMS to use “Character”.

Arguments

String a valid string/index value to represent the character

Returned Value

boolean returns true if the object was successful, false otherwise.

See Also

[Symbol](#)

[IMSMMap](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
boolean ret = sym.setCharacter(240);
```

TrueTypeMarkerSymbol.setAntialiasing

Description

Sets the current antialiasing value used by the Text symbol object. Antialiasing is the process of removing or reducing the jagged distortions in curves and diagonal lines so that the lines appear smooth or smoother.

Syntax

```
boolean setAntialiasing(String)
```

Arguments

String “true” for turning this option on, “false” to turn it off

Returned Value

boolean returns true if the method was successful, false otherwise

See Also

Symbol

Example

```
var sym = imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
boolean ret = sym.getAntialiasing();
```

TrueTypeMarkerSymbol.setBlockoutColor

Description

Sets the current color value used by the Blockout effect of the text symbol object

Syntax

boolean setBlockoutColor(java.awt.Color)

Arguments

Color any valid color object created using the imsmap helper method, createColor(r,g,b)

Returned Value

boolean returns true if the method was successful, false otherwise

See Also

Symbol
Color
IMSMMap

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret = sym.setBlockoutColor(imsmap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setFont

Description

Sets the font object used by the Text symbol object to draw the text for the labels.

Syntax

```
boolean setFont(String fontname, int fontsize)
```

Arguments

String	fontname - the name of the font to use
int	fontsize - the size of the font to use

Returned Value

boolean	returns true if the method was successful in setting the font, false otherwise
---------	--

See Also

[Symbol](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
if( !sym.setFont("times", 12)) {
    report the failure - do something...
}
```

TrueTypeMarkerSymbol.setFontColor

Description

Sets the color used to draw the text.

Syntax

```
boolean setFontColor(Color col)
```

Arguments

Color	color - any valid color object created using the imsmap helper method, createColor(r,g,b)
-------	--

Returned Value

boolean	returns true if the color was set successfully, false otherwise.
---------	--

See Also

[Symbol](#)
[Color](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret = sym.setFontColor(imsmap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setGlowColor

Description

Sets the color used for the Glowing effect.

Syntax

```
boolean setGlowColor(Color color)
```

Arguments

Color	any valid color object created using the imsmap helper method, createColor(r,g,b)
-------	---

Returned Value:

boolean	returns true if the method was successful, false otherwise.
---------	---

See Also

[Symbol](#)
[Color](#)
[IMSMMap](#)

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret = sym.setGlowColor(imsmap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setOutlineColor

Description

Sets the outline color of this object.

Syntax

```
boolean setOutlineColor(java.awt.Color colorValue)
```

Arguments

Color	any valid color object created using the imssmap helper method, createColor(r,g,b)
-------	--

Returned Value

boolean	returns true if the method succeeded, false otherwise
---------	---

See Also

Symbol
Color
IMSSMap

Example

```
var sym imssmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret=sym..setOutlineColor(imssmap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setShadowColor

Description

The shadow effect methods provide for a mirrored image or reflection to appear behind and below the text as it is labeled. Use this method to set the color value for the shadow effect on the TextSymbol. The shadow will be drawn using a 0.5 transparency.

Syntax

```
boolean setShadowColor(Color colorValue )
```

Arguments

Color a valid Color value using the IMSMap helper method createColor(r,g,b).

Returned Value

boolean true if the method was successful, false otherwise.

See Also

Symbol

Color

IMSMMap

Example

```
var sym imsmap.createSymbol("TRUETYPE_MARKER_SYMBOL");
if(sym.setShadowColor(imsmap.createColor(255,0,0))) {
    do something...
}
```

TrueTypeMarkerSymbol.setTransparency

Description

The setTransparency method allows you to set the level of transparency on the text. Valid values are 0.0 (Transparent) to 1.0 (Opaque).

Syntax

```
boolean setTransparency(double n)
```

Arguments

double	n - A valid number in the range of 0.0 to 1.0. The 1.0 value is a number that is guaranteed to be completely opaque. 0.0 will produce a transparent image.
--------	--

Returned Value

None

See Also

[Symbol](#)

Example

```
var sym = imsmap.createSymbol("TRUEETYPE_MARKER_SYMBOL");
var ret = sym.setTransparency(1.0);
```

ValueMapLabelRenderer extends Renderer

A ValueMapLabelRenderer is an object that represents a way of labeling features of a map layer by drawing a Symbol for each unique data value or range of data values specified by the setField or setFields property values. Field property is the name of the field in the layer that stores the text values to use as labels. The Symbol property defines how the text is drawn on the feature. The ValueMapLabelRenderer supports both unique values and ranges. Using the setFields and setSeparator methods, you can set one or more Fields that will be used to provide text for labeling.

ValueMapLabelRenderer.getSeparator

Description

The getSeparator method retrieves the current value specified to separate the display of multiple field values used to display labels on a feature.

Syntax

`String getSeparator()`

Arguments

None

Returned Value

`String` returns the string character specified in the setSeparator method.

See Also

[Renderer](#)

[ValueMapLabelRenderer.setSeparator](#)

[IMSSMap](#)

Example

```
var ValueMapLabelRenderer;  
ValueMapLabelRenderer = imsmap.getLayerLabelRenderer (myLayer);  
var separatorChar = ValueMapLabelRenderer.getSeparator();  
alert( "The separator used is: " + separatorChar );
```

ValueMapLabelRenderer.setDefaultSymbol

Description

Sets the default Symbol to be used by the ValueMapLabelRenderer to draw the text when either a ValueRange or Unique Values are specified that do not meet the criteria defined.

Syntax

boolean setSymbol(Symbol)

Arguments

Symbol A text symbol object that controls how text is rendered.

Returned Value

boolean returns true if the method was successful in setting the symbol specified,
 false otherwise

See Also

Renderer

Symbol

ArcXML Programmer's Reference

Example

See the next page

ValueMapLabelRenderer.setDefaultSymbol

Example

```
var sym, valuemapLabelRenderer;  
var range;  
sym = imsmap.createSymbol("TEXT_SYMBOL");  
valuemapLabelRenderer=imsmap.createRenderer("VALUEMAP_LABEL_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
sym.setFontColor(imsmap.createColor(255,255,255));  
valuemapLabelRenderer.setDefaultSymbol(sym);  
range = imsmap.createValueRange(myLayer, fieldName, "0", "200");  
sym.setGlowColor(imsmap.createColor(178,176,0));  
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);  
range = imsmap.createValueRange(myLayer, fieldName, "201", "400");  
sym.setGlowColor(imsmap.createColor(255,0,0));  
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);  
range = imsmap.createValueRange(myLayer, fieldName, "401", "600");  
sym.setGlowColor(imsmap.createColor(0,0,245));  
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);  
imsmap.setLayerLabelRenderer(myLayer, valuemapLabelRenderer );
```

ValueMapLabelRenderer.setField

Description

The setField method allows you to use a single field to draw text on a feature. Any value set in the setSeparator method will be ignored.

Syntax

```
boolean setField(Layer, String)
```

Arguments

Layer	the layer object to apply the field to
String	The Field name used to store values used to draw text on a feature

Returned Value

boolean	returns true if the method is successful, false otherwise
---------	---

See Also

Renderer
Layer
IMSMMap

Example

```
var sym, ValueMapLabelRenderer;  
sym = imsmmap.createSymbol("TEXT_SYMBOL");  
ValueMapLabelRenderer=imsmmap.createRenderer("VALUemap_LABEL_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
ValueMapLabelRenderer.setSymbol(sym);  
ValueMapLabelRenderer.setField(myLayer, "field1");  
imsmmap.setLayerLabelRenderer(myLayer, ValueMapLabelRenderer );
```

ValueMapLabelRenderer.setFields

Description

The setFields method allows you to use multiple fields to draw text on a feature. Use the setSeparator method when using the setFields method to identify the string used in the display between fields.

Syntax

```
boolean setFields(Layer layerObject, Collection col)
```

Arguments

Layer	layerObject – the layer object to apply the fields on
Collection	col – The collection object that contains the field names to use

Returned Value

boolean returns true if the method was successful setting the separator value specified, false otherwise

See Also

Renderer	Collection
Layer	ArcXML Programmer's Reference

Example

```
var col  
var sym, ValueMapLabelRenderer;  
sym = imsmap.createSymbol("TEXT_SYMBOL");  
ValueMapLabelRenderer = imsmap.createRenderer("VALUemap_LABEL_RENDERER");  
col = imsmap.createCollection();  
col.addStringElement("field1");  
col.addStringElement("field2");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
ValueMapLabelRenderer.setSymbol(sym);  
ValueMapLabelRenderer.setSeparator(":");  
ValueMapLabelRenderer.setFields(myLayer, col);  
imsmap.setLayerLabelRenderer(myLayer, ValueMapLabelRenderer );
```

ValueMapLabelRenderer.setSeparator

Description

The setSeparator method allows you to set a separator character when multiple fields are used to display labels on a feature.

Syntax

```
boolean setSeparator(String)
```

Arguments

String any valid string character that can be used to separate multiple field value displays when the text is drawn on a feature. The default value is a space character ““.

Returned Value

boolean returns true if the method was successful setting the separator value specified, false otherwise.

See Also

Renderer IMSMap

Example

```
var col  
var sym, ValueMapLabelRenderer;  
sym = imsmap.createSymbol("TEXT_SYMBOL");  
ValueMapLabelRenderer = imsmap.createRenderer("VALUemap_LABEL_RENDERER");  
col = imsmap.createCollection();  
col.addStringElement("field1");  
col.addStringElement("field2");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
ValueMapLabelRenderer.setSymbol(sym);  
ValueMapLabelRenderer.setSeparator(":");  
ValueMapLabelRenderer.setFields(myLayer, col);  
imsmap.setLayerLabelRenderer(myLayer, ValueMapLabelRenderer );
```

ValueMapLabelRenderer.setSymbolForRangeValue

Description

A RangeValue is a way of representing a way of classifying features into categories or classes, by drawing different symbols for text specified by the range. Using this method, you can specify the Range value and a symbol to be used for that range. When the LabelEngine draws the text on the feature, it will use the correct symbol for the value of the field.

Syntax

```
boolean setSymbolForRangeValue(Symbol symObject, ValueRange range)
```

Arguments

Symbol	A symbol object that is used to control how text is rendered
ValueRange	range - A Range defines a pair of Comparable values

Returned Value

boolean	returns true if the method was successful, false otherwise
---------	--

See Also

Renderer	Symbol
ValueRange	IMSSMap

Example

See the next page

ValueMapLabelRenderer.setSymbolForRangeValue

Example

```
var sym, valuemapLabelRenderer;  
var range;  
sym = imsmap.createSymbol("TEXT_SYMBOL");  
valuemapLabelRenderer = imsmap.createRenderer("VALUEMAP_LABEL_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
sym.setFontColor(imsmap.createColor(255,255,255));  
valuemapLabelRenderer.setDefaultSymbol(sym);  
range = imsmap.createValueRange(myLayer, fieldName, "0", "200");  
sym.setGlowColor(imsmap.createColor(178,176,0));  
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);  
range = imsmap.createValueRange(myLayer, fieldName, "201", "400");  
sym.setGlowColor(imsmap.createColor(255,0,0));  
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);  
range = imsmap.createValueRange(myLayer, fieldName, "401", "600");  
sym.setGlowColor(imsmap.createColor(0,0,245));  
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);  
imsmap.setLayerLabelRenderer(myLayer, valuemapLabelRenderer);
```

ValueMapLabelRenderer.setSymbolForUniqueValue

Description

This method is a way of symbolizing text to be drawn on features of a Layer by drawing a Symbol for each unique value. Use this method to add unique values to the LabelRenderer using a Field obtained from the Layer and the value to assign the symbol.

Syntax

```
boolean setSymbolForUniqueValue(Symbol symObject, Layer layerObject, String fieldname,  
String value)
```

Arguments

Symbol	symObject – A symbol object that is used to control how text is rendered.
Layer	layerObject - the layer object to apply unique value
String	fieldname – The name of the field obtained from the layer.
String	value – unique value of the field to apply this symbol

Returned Value

boolean	returns true if the method was successful setting the unique value, false otherwise
---------	---

See Also

Renderer	Layer
Symbol	IMSSMap

Example

See the next page

ValueMapLabelRenderer.setSymbolForUniqueValue

Example

```
var col  
var sym, valuemapLabelRenderer;  
sym = imsmap.createSymbol("TEXT_SYMBOL");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
valuemapLabelRenderer = imsmap.createRenderer("VALUemap_LABEL_RENDERER");  
valuemapLabelRenderer.setSymbolForUniqueValues(sym, myLayer, "field1", "200");  
valuemapLabelRenderer.setSymbolForUniqueValues(sym, myLayer, "field1", "300");  
valuemapLabelRenderer.setSymbolForUniqueValues(sym, myLayer, "field1", "400");  
valuemapLabelRenderer.setSymbol(sym);  
valuemapLabelRenderer.setField(myLayer, "field1");  
imsmap.setLayerLabelRenderer(myLayer, valuemapLabelRendered);
```

ValueMapRenderer extends Renderer

A ValueMapRenderer is an object that represents a way of symbolizing features of a map layer by drawing a Symbol for each unique data value or range of data values. The Field property is the name of the field in the layer that stores the text values to use. Depending on the type of feature, the Symbol property defines how the feature is drawn on the map. For those values not listed explicitly, use the DefaultSymbol method to identify a Symbol to render the features.

ValueMapRenderer.getField

Description

The getField method retrieves the value set using the setField method.

Syntax

```
String getField()
```

Arguments

None

Returned Value

String	The Field name that stores values to draw a feature
--------	---

See Also

Renderer
Layer
IMSMMap

Example

```
var valRenderer, field;  
valRenderer = imsmap.getLayerRenderer(myLayer);  
field = valRenderer.getField();
```

ValueMapRenderer.setDefaultSymbol

Description

Sets the default Symbol to be used by the ValueMapRenderer to draw the feature when either a ValueRange or Unique Values are specified that do not meet the criteria defined.

Syntax

```
boolean setDefaultSymbol(Symbol)
```

Arguments

Symbol A symbol object that controls how features are rendered.

Returned Value

boolean	returns true if the method was successful in setting the symbol specified, false otherwise
---------	---

See Also

Renderer
Symbol
ArcXML Programmer's Reference

Example

See the next page

ValueMapRenderer.setDefaultSymbol

Example

```
var sym, valRenderer, range;
sym = imsmap.createSymbol();
valRenderer = imsmap.createRenderer("VALUemap_RENDERER");
sym.setFont("times", 12);
sym.setAntialiasing("true");
sym.setFontColor(imsmap.createColor(255,255,255));
valRenderer.setDefaultSymbol(sym);
range = imsmap.createValueRange(myLayer, fieldName, "0", "200");
sym.setGlowColor(imsmap.createColor(0,0,34));
valRenderer.setSymbolForRangeValue(sym, range);
range = imsmap.createValueRange(myLayer, fieldName, "201", "400");
sym.setGlowColor(imsmap.createColor(255,0,0));
valRenderer.setSymbolForRangeValue(sym, range);
range = imsmap.createValueRange(myLayer, fieldName, "401", "600");
sym.setGlowColor(imsmap.createColor(100,200,0));
valRenderer.setSymbolForRangeValue(sym, range);
imsmap.setLayerValueMapRenderer(myLayer, valRenderer);
```

ValueMapRenderer.setField

Description

The setField method allows you to use a single field to classify the features.

Syntax

boolean setField(Layer, String)

Arguments

Layer	the layer object to apply the field to
String	The Field name used to store values used to draw text on a feature

Returned Value

boolean returns true if the method was successful, false otherwise

See Also

Renderer
Layer
IMSMMap

Example

```
var sym, valRenderer ;  
sym = imsmmap.createSymbol();  
valRenderer = imsmmap.createRenderer("VALUemap_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
valRenderer.setSymbol(sym);  
valRenderer.setField(myLayer, "field1");  
imsmmap.setLayerValueMapRenderer (myLayer, valRenderer );
```

ValueMapRenderer.setSymbolForRangeValue

Description

A Range Value is used to represent a way of classifying features into categories or classes, by drawing different symbols for features specified by the range. Using this method, you can specify the Range value and a symbol to be used for that range. When the Display Engine draws the feature, it will use the correct symbol for the value of the field.

Syntax

```
boolean setSymbolForRangeValue(Symbol symObject, ValueRange range)
```

Arguments

Symbol	A symbol object that is used to control how text is rendered
ValueRange	range - A Range defines a pair of Comparable values

Returned Value

boolean	returns true if the method was successful, false otherwise
---------	--

See Also

Renderer	Symbol
ValueRange	IMSMMap

Example

See the next page

ValueMapRenderer.setSymbolForRangeValue

Example

```
var sym, valRenderer, range;  
sym = imsmap.createSymbol();  
valRenderer = imsmap.createRenderer("VALUemap_RENDERER");  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
sym.setFontColor(imsmap.createColor(255,255,255));  
valRenderer.setDefaultSymbol(sym);  
range = imsmap.createValueRange(myLayer, fieldName, "0", "200");  
sym.setGlowColor(imsmap.createColor(0,0,34));  
valRenderer.setSymbolForRangeValue(sym, range);  
range = imsmap.createValueRange(myLayer, fieldName, "201", "400");  
sym.setGlowColor(imsmap.createColor(255,0,0));  
valRenderer.setSymbolForRangeValue(sym, range);  
range = imsmap.createValueRange(myLayer, fieldName, "401", "600");  
sym.setGlowColor(imsmap.createColor(100,200,0));  
valRenderer.setSymbolForRangeValue(sym, range);  
imsmap.setLayerValueMapRenderer(myLayer, valRenderer);
```

ValueMapRenderer.setSymbolForUniqueValue

Description

This method represents a way of symbolizing features of a Layer by drawing a Symbol for each unique data value. Use this method to add unique values to the Renderer using a Field obtained from the Layer and the value to assign the symbol. When the Display Engine draws the feature, it will use the correct symbol for the value of the field.

Syntax

```
boolean setSymbolForUniqueValue(Symbol symObject, Layer layerObject, String fieldname,  
String value)
```

Arguments

Symbol symObject – A symbol object that is used to control how text is rendered.

Layer layerObject - the layer object to apply unique value

String fieldname – The name of the field obtained from the layer.

String value – unique value of the field to apply this symbol

Returned Value

boolean	returns true if the method was successful setting the unique value, false otherwise
---------	---

See Also

Renderer	Layer
Symbol	IMSMMap

Example

See the next page

ValueMapRenderer.setSymbolForUniqueValue

Example

```
var col;  
var sym, ValueMapRenderer ;  
sym = imsmap.createSymbol();  
sym.setFont("times", 12);  
sym.setAntialiasing("true");  
ValueMapRenderer = imsmap.createRenderer("VALUE_LABEL_RENDERER");  
ValueMapRenderer.setSymbolForUniqueValues(sym, myLayer, "field1", "200");  
ValueMapRenderer.setSymbolForUniqueValues(sym, myLayer, "field1", "300");  
ValueMapRenderer.setSymbolForUniqueValues(sym, myLayer, "field1", "400");  
ValueMapRenderer.setSymbol(sym);  
ValueMapRenderer.setField(myLayer, "field1");  
imsmap.setLayerValueMapRenderer(myLayer, ValueMapRenderer );
```

ValueRange

A Range defines a pair of Comparable values that can be used with a ValueMapRenderer.

ValueRange.getLower

Description

Retrieves the lower range value as defined by the setLower method.

Syntax

String get lower()

Arguments

None

Returned Value

String the lower value, the literal “None” is returned if the no field is specified.

See Also

Symbol
ArcXML Programmer's Reference

Example

```
var val=range.getLower();
```

ValueRange.getUpper

Description

Retrieves the upper range value as defined by the setUpper method.

Syntax

`String getUpper()`

Arguments

None

Returned Value

`String` – the upper value, “None” if not set

See Also

[Symbol](#)

[ArcXML Programmer's Reference](#)

Example

```
var val = range.getUpper();
```

ValueRange.inRange

Description

The `inRange` method is used to determine if the specified value is between the upper and lower value. The comparison of the value is made based on the Java Interface `java.lang.Comparable`. Refer to the Java Developer kit documentation for the description.

Syntax

```
boolean inRange( Layer layerObject, String fieldName, String value)
```

Arguments

`Layer` a valid layer object from the `IMSMMap`

`String` Field Name to use within the Layer

`String` the upper value to use in this range

Returned Value

`boolean` true if the value specified by the third argument is within the lower and upper range values, false otherwise.

See Also

`Symbol`

`IMSMMap`

Example

```
if( range.inRange(layerObject, "Name", 50) ) {  
    do something...  
}
```

ValueRange.setLower

Description

Sets the lower value in the range. A value can be defined as one of the following data types as defined by the java.lang.Comparable Interface: Character, Long, Short, String, Float, Integer, Byte, Double, BigInteger, BigDecimal, Date.

Syntax

boolean setLower(String)

Arguments

String the lower value to use in this range, the literal “None” is returned.

Returned Value:

boolean true if method set the upper range successfully, false otherwise

See Also

Symbol

IMSMMap

Example

```
boolean ret = range.setLower( 1 );
```

ValueRange.setUpper

Description

Sets the upper value in the range. A value can be defined as one of the following data types as defined by the java.lang.Comparable Interface: Character, Long, Short, String, Float, Integer, Byte, Double, BigInteger, BigDecimal, Date.

Syntax

```
boolean setUpper(Layer, String, String)
```

Arguments

Layer a valid layer object from the IMSMap

String Field Name to use within the Layer

String the upper value to use in this range

Returned Value

boolean returns true if the method was successful, false otherwise

See Also

Symbol

IMSMap

Example

```
If( range.setUpper(lyr, "name", 100) ) {
    alert("Upper range was set successfully");
} else {
    alert("Setting the upper range failed");
}
```