ArcIMS® 9

Customizing the Java™ Viewer

ArcGIS™ ESRI

GIS by ESRI™

Contents

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® software provides a suite of tools allowing you to create very effective Web sites for your mapping and geographic information system (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®, ColdFusion®, and Java 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 Service) or streamed features (Feature Service). The Java Viewer can display one or more Image or Feature Services in any combination. In addition, data from local sources (shapefiles, ArcSDE® layers, or images) 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 scalebar 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. The Java applets are only sent to the client's machine once, after the first download. They then reside on the client's machine. 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 Service, 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 one-time downloads, but each is more than 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, Microsoft® 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 tool 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.

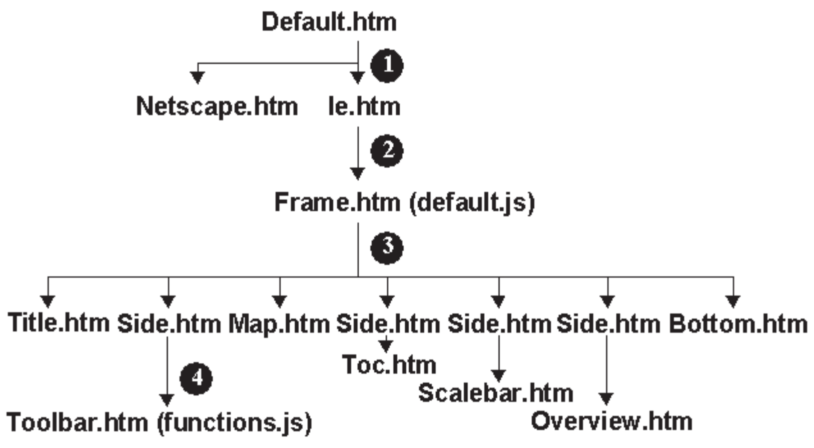
The default.js and functions.js parameter files

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, scalebar 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 file 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 calls default.js, which contains JavaScript functions and variables used later for configuring the viewer.

In step 3, once the frames are defined, the HTML files for each added frame will be executed. Initially, only the map applet is loaded. 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 applet name, Java version, and the configuration (*.axl) file to load into the the IMSApplet. 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 one-time download of the components. Blank pages are loaded for the legend, scalebar, and overview map—these applets are loaded in sequence.

When the map applet is finished loading, a JavaScript function is called to load the next applet, the legend. This action is repeated for the loading of each applet page. After all the applet pages have been loaded, the function init(), found in default.js, is called to finalize the startup sequence. The init() function calls a number of functions to load the toolbar, load the viewer configuration file (default.axl), and 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 defines the display of the buttons and calls functions.js, which contains JavaScript functions used to control the applets.

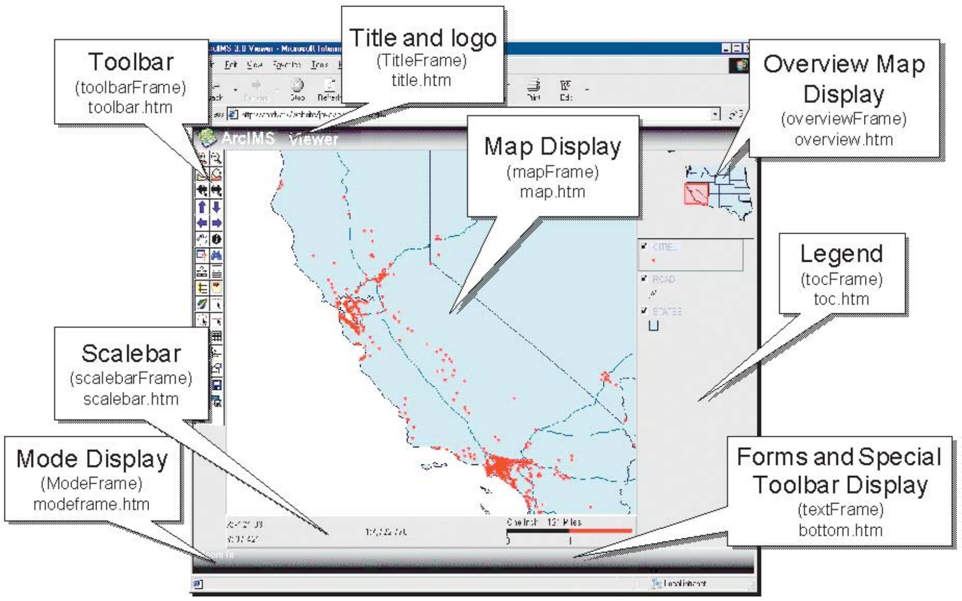
The toolbar.htm file has a table to define the number, order, and position of tools on the toolbar. Each tool has the location of the image, the name, and instructions for the onmousedown() function. Toolbar.htm references the functions.js file, which provides the JavaScript functions for supporting 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 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 scalebar applet to define the content of the ScalebarFrame. The ScalebarFrame displays the scalebar 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 service**
- **Opening files: changing the order and removing files**
- **Changing title, logo, and background**
- **Loading the service, overview map, and toolbar**
- **Setting visible layers**
- **Setting the right-click menu**
- **Setting MapTips**
- **Setting visibility and alias for results**
- **Setting the scalebar 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 service

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

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 services.

By default, the information from each service 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 service.

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

Refreshing an Image Service

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

Refreshing a Feature Service

When a Feature Service 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 service include using the Viewer Object Model to manipulate the map. This chapter contains some examples of 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 installation and open frame.htm.

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

```
<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 the parent.modeFrame.document in default.js and functions.js.
- If you remove textFrame, you will need to remove references to the 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 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 scalebar with the setAppletColor() function in default.js. See the "Setting colors for the applets" section later in this chapter for details.

Loading the service, overview map, and toolbar

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

Loading the service

The service is added to the Java Viewer when the IMSMap applet is loaded in the map.htm file. The “AXL” parameter is read by IMSMap upon initialization and, by default, loads in the service defined in the default.axl file created by Designer.

```
document.writeln(‘<object classid=“clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
width=100% height=  100% codebase=“../install/install.htm” ID=“IMSMap”>’);
document.writeln(‘<param NAME=“code” VALUE=“com/esri/aims/viewer/IMSMap”>’);
document.writeln(‘<param NAME=“name” VALUE=“IMSMap”>’);
document.writeln(‘<param NAME=“AXL” VALUE=”’ + axl + ‘“>’);
document.writeln(‘<param NAME=“SCRIPTABLE” VALUE=“TRUE”>’);
document.writeln(‘<param NAME=“type” VALUE=”application/x-java-
applet;version=1.3”>’);
document.writeln(‘<comment>’);
document.writeln(‘<embed type=”application/x-java-applet;version=1.3” width=“600”
height=“480”  code=“com/esri/ae/applet/IMSMap” name=“IMSMap” pluginspage=“../
install/install.htm”>’);
document.writeln(‘<noembed>’);
document.writeln(‘</COMMENT>’);
document.writeln(‘      No JDK 1.3 support for APPLET!!’);
document.writeln(‘  </noembed>’);
document.writeln(‘</object>’);
```

Alternatively, the Viewer API supports loading the service directly through the IMSMap applet using loadMapOnlyProject. This is currently not the default loading method for the Java Viewer.

Loading the overview map

Layers are added to the overview map with the setOverviewLayers() function in the default.js file. Only layers that are in IMSMap can be in the overview map. For Image Services, the entire service must appear in the overview map. For Feature Services, you can choose which layers to add by specifying the layer name using GetLayer() and adding the layer using addMapLayer(). If two services 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.IMSOverviewMap;
    if (overviewApplet == null)
        return;
   var mapApplet = parent.mapFrame.document.IMSMap;
    if (mapApplet == null)
        return;
   setTimeout (‘parent.overviewFrame.document.IMSOverviewMap’, 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 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, 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 the right-click menu

Right-clicking 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. By default, the setupTOCPopupMenuItems() function is commented in the default.js file. 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 Service 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 scalebar 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 scalebar 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 scalebar. Color values are based on red–green–blue values of 0–255. The fourth parameter on the setRectangleFillColor and setRectangleOutlineColor is the alpha (a) parameter used to set the opaque level for the color. Only these two methods use the alpha parameter.

```
function setAppletColor() {
     //map
     var mapApplet = parent.mapFrame.document.IMSMap;
     if (mapApplet != null)
        mapApplet.setBGColor (255,255,255);
     //overview
     var overviewApplet = parent.overviewFrame.document.IMSOverviewMap;
     if (overviewApplet != null) {
        overviewApplet.setBGColor (192,192,192);
        overviewApplet.setRectangleFillColor (255,0,0,80);
        overviewApplet.setRectangleOutlineColor (255,0,0,80);
     }
     //toc
     var tocApplet = parent.tocFrame.document.IMSToc;
     if (tocApplet != null) {
        tocApplet.setBGColor (192,192,192);
        tocApplet.setFGColor (0,0,0);
        tocApplet.refresh();
     }
     //scalebar
     var scalebarApplet = parent.scalebarFrame.document.IMSScaleBar;
     if (scalebarApplet != null) {
        scalebarApplet.setBGColor (153,255,153);
        scalebarApplet.setFGColor (255,255,0);
        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.js 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 Service, ArcView® Internet Map Server (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(‘<img src=“images/delete.gif” width=16 height=16 hspace=1
     vspace=1 border=0 alt=“Remove Current Layer”
     onmousedown=“clickFunction(\‘removelayer\’);”
     onmouseover=“window.status=\‘Remove Current Layer\’”>’);
   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 the 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 have 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 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 box: 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('<img src="images/attributes_1.gif" width=16 height=16 hspace=0
vspace=1 border=0 alt="Attributes" onmousedown="showSelect(\'\');"
onmouseover="window.status=\'Attributes\'">');
     // the next line will display attributes of selected features in Java table. .
. faster but cannot be customized
     // document.write('<img src="images/attributes_1.gif" width=16 height=16
hspace=0 vspace=1 border=0 alt="Attributes" onmousedown="displayAttributes(\'\');"
onmouseover="window.status=\'Attributes\'">');
     //
     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 service or local data to the map and service layers to the overview map. This provides a way to test the Java Viewer without creating a service 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 text links along the left side of the map.

To use this sample:

1. Create a Feature service named SanFranFeature (case-sensitive) from sf.axl.
2. Go to \javaviewer\BlackTie and open the default.axl file in a text editor. Look for the <FEATURESERVERWORKSPACE> 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\BlackTie, the frame.htm file defines each of the files that creates the entire Web page, map.htm, bottom.htm, overview.htm, etc. Look at each of these HTML files and notice that some have the background color set to black.

Java Viewer Object Model 3

IN THIS CHAPTER

- **Java Viewer Object Model classes**
- **Associated public methods**

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 the functionality and appearance of the client.

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

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

CallOutMarkerSymbol extends Symbol

A CallOutMarkerSymbol consists of a **label** and a **callout box** around the label. The methods on the following pages apply to either the label or callout box of the CallOutMarkerSymbol. This symbol works with point features only. The CallOutMarkerSymbol allows special effects such as glowing, shadows, outline, and antialiasing of the label.You can create an instance of the class by calling the IMSMap.createSymbol method.

See Also

IMSMap

Symbol

ArcXML Programmer's Reference Guide

CallOutMarkerSymbol.getAntialiasing

Description

Returns the antialiasing value of the label. Antialiasing is the process of adding pixels along lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean getAntialiasing()

Arguments

none

Returned Value

boolean

See Also

CallOutMarkerSymbol.setAntialiasing

Example

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

CallOutMarkerSymbol.getBackColor

Description
Returns the background color of the callout box. The default value is white.

Syntax
String getBackColor( )

Argument
none

Returned Value
A comma-delimited string value (RGB):
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 = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
alert(“default value of the background color for CallOutMarkerSymbol is” +
symbol.getBackColor());
```

CallOutMarkerSymbol.getBoundaryColor

Description
Returns the color of the callout box boundary. The default value is black.

Syntax
String getBoundaryColor( )

Arguments
none

Returned Value
A comma-delimited string value (RGB):
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 = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
alert(“default value of the boundary is” + symbol.getBoundaryColor());
```

CallOutMarkerSymbol.getFontColor

Description
Returns the color of the font for 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 = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
alert(“default value of  the font color for CallOutMarkerSymbol is” +
symbol.getFontColor());
```

CallOutMarkerSymbol.getFontName

Description
Returns the name of the font for the label.

Syntax
String getFontName( )

Arguments
none

Returned Value

String	The name of the font used.

See Also
CallOutMarkerSymbol.setFont

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
alert(“default font name for CallOutMarkerSymbol is” + symbol.getFontName());
```

CallOutMarkerSymbol.getFontSize

Description
Returns the size of the font for the label.

Syntax
int getFontSize( )

Arguments
none

Returned Value

int	The size (in points) of the font specified.

See Also
CallOutMarkerSymbol.setFont

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

CallOutMarkerSymbol.getGlowColor

Description
Returns the color of the glow effect around the label. The color value must first be set by the setGlowColor method.

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

CallOutMarkerSymbol.getInterval

Description
Returns the distance between the callout box and the point feature. The default value is 0.

Syntax
int getInterval( )

Arguments
none

Returned Value
int	The distance in screen pixels.

See Also
CallOutMarkerSymbol.setInterval

Example
```
var symbol = parent.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 for the label. The color value must first be set up by the setOutlineColor 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
CallOutMarkerSymbol.setOutlineColor

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

CallOutMarkerSymbol.getShadowColor

Description
Returns the color of the shadow effect for the label. The color value must first be set by the setShadowColor 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
CallOutMarkerSymbol.setShadowColor

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

CallOutMarkerSymbol.getTransparency

Description
Returns the transparency value for the symbol.

Syntax
double getTransparency( )

Arguments
none

Returned Value
double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
CallOutMarkerSymbol.setTransparency

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

CallOutMarkerSymbol.setAntialiasing

Description

Sets an antialiasing value for the label. Antialiasing is the process of adding pixels along the diagonal lines to smooth the jagged appearance.

Syntax

boolean setAntialiasing(String enabled)

Arguments

enabled	"True" to set antialiasing on. "False" to set antialiasing off. Antialiasing is off by default.

Returned Value

boolean

See Also

CallOutMarkerSymbol.getAntialiasing

Example

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

CallOutMarkerSymbol.setBackColor

Description

Sets a background color for the callout box.

Syntax

boolean setBackColor(Color color)

Arguments

color	An instance of the color class.

Returned Value

boolean

See Also

CallOutMarkerSymbol.getBackColor
Color

Example

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

CallOutMarkerSymbol.setBoundaryColor

Description
Sets a color for the boundary of the callout box.

Syntax
boolean setBoundaryColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getBoundaryColor
Color

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
var color1 = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setBoundaryColor(color);
```

CallOutMarkerSymbol.setFont

Description
Sets a font and size for the label.

Syntax
boolean setFont(String fontName, int size)

Arguments

fontName	The name of the font.
size	The size of the font in points.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getFontName
CallOutMarkerSymbol.getFontSize

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
symbol.setFont(“Arial”, 12);
```

CallOutMarkerSymbol.setFontColor

Description
Sets a font color for the label.

Syntax
boolean setFontColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getFontColor
Color

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

CallOutMarkerSymbol.setGlowColor

Description
Sets a color for the glowing effect around the label.

Syntax
boolean setGlowColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getGlowColor
Color

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

CallOutMarkerSymbol.setInterval

Description
Sets a distance between the callout box and the point feature.

Syntax
boolean setInterval (double intervalValue)

Arguments

intervalValue	Defines the distance in screen points; must be a positive number.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getInterval

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
symbol.setInterval(2);
```

CallOutMarkerSymbol.setOutlineColor

Description
Sets a color for the outline of the label.

Syntax
boolean setOutlineColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getOutlineColor
Color

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
var color = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setOutlineColor(color);
```

CallOutMarkerSymbol.setShadowColor

Description
Sets a color for the shadow effect of the label. The shadow is drawn with a 0.5 transparency.

Syntax
boolean setShadowColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
CallOutMarkerSymbol.getShadowColor
Color

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
var color = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setShadowColor(color);
```

CallOutMarkerSymbol.setTransparency

Description
Sets a level of transparency on the label.

Syntax
boolean setTransparency(double transparencyValue)

Arguments

transparencyValue	Defines the transparency value. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value
boolean

See Also
CallOutMarkerSymbol.getTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“CALLOUT_MARKER_SYMBOL”);
symbol.setTransparency(0.5);
```

Collection

Description

A collection is a 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 useful for passing data between Java and JavaScript.

See Also

IMSMap.createCollection
NameValuePair

Collection.addNameValuePairElement

Description

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

Syntax

boolean addNameValuePairElement(NameValuePair pair)

Arguments

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

Returned Value

boolean

See Also

IMSMap.getCollectionElementAt
NameValuePair

Example

```
var imsMap = parent.mapFrame.IMSMap;
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 of the collection. The index is equal to Collection.getSize().

Syntax
boolean addString(String string)

Arguments
string	The string to be added to the end of the collection.

Returned Value
boolean

See Also
IMSMap

Example
```
var imsMap = parent.mapFrame.IMSMap;
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 of 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.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 element.

Returned Value

String	The string at the given index.

See Also

Collection.addStringElement

Example

```
var imsMap= parent.mapFrame.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 such as symbols. You can create an instance of the color class by calling the IMSMap.createColor method.

See Also

IMSMap
Symbol

Example

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

Extent

The extent class is used to define a rectangle on a map. This class includes methods to get the minimum and maximum x- and y-values in map units. It is used in many methods to set the extent or the extent limit of the map. An extent can be constructed through IMSMap.createExtent().

See Also

IMSMap

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

GradientFillSymbol extends Symbol

The GradientFillSymbol consists of a set of directional lines on top of a graduated color. You can set an antialiasing effect on the directional lines. You can create an instance of the GradientFillSymbol with the IMSMap.createSymbol method.

See Also

IMSMap
Symbol
ArcXML Programmer's Reference Guide

GradientFillSymbol.getAntialiasing

Description

Returns the antialiasing value of the directional lines. Antialiasing is the process of adding pixels along the diagonal lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean getAntialiasing()

Arguments

none

Returned Value

boolean

See Also

GradientFillSymbol.setAntialiasing

Example

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

GradientFillSymbol.getEndColor

Description
Returns the end color of the graduated color. The default value 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.getStartColor
GradientFillSymbol.setEndColor

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

GradientFillSymbol.getStartColor

Description
Returns the start color of the graduated color. The default value 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.getEndColor
GradientFillSymbol.setStartColor

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

GradientFillSymbol.getStyle

Description

Returns the style of the directional lines. The default style 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 = parent.mapFrame.IMSMap.createSymbol(“GRADIENT_FILL_SYMBOL”);
alert(“the default style for the GradientFillSymbol is” + symbol.getStyle());
```

GradientFillSymbol.getTransparency

Description

Returns the transparency value of the graduated color.

Syntax

double getTransparency( )

Arguments

none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also

GradientFillSymbol.setTransparency

Example

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

GradientFillSymbol.setAntialiasing

Description

Sets the antialiasing value of the directional lines. Antialiasing is the process of adding pixels along the diagonal lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean setAntialiasing(String enabled)

Arguments

enabled	"True" sets antialiasing on. "False" sets antialiasing off.

Returned Value

boolean

See Also

GradientFillSymbol.getAntialiasing

Example

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

GradientFillSymbol.setEndColor

Description

Sets the end color of the graduated color. The default value is green.

Syntax

boolean setEndColor(Color color)

Arguments

color	An instance of the color class.

Returned Value

boolean

See Also

Color
GradientFillSymbol.getEndColor

Example

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

GradientFillSymbol.setStartColor

Description
Sets the start color of the graduated color. The default value is red.

Syntax
boolean setStartColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
Color
GradientFillSymbol.getStartColor

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

GradientFillSymbol.setStyle

Description
Sets the style of the directional lines. The default style is backward diagonal.

Syntax
boolean setStyle(int style)

Arguments

style	0—forward diagonal 1—backward diagonal 2—horizontal 3—vertical

Returned Value
boolean

See Also
GradientFillSymbol.getStyle

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

GradientFillSymbol.setTransparency

Description
Sets the transparency value of the graduated color.

Syntax
boolean setTransparency(double transparencyValue)

Arguments

transparencyValue	Transparency to be set. The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value
boolean

See Also
GradientFillSymbol.getTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“GRADIENT_FILL_SYMBOL”);
symbol.setTransparency(0.5);
```

GroupRenderer extends Renderer

A GroupRenderer is used to draw a layer using a collection of renderers. The renderers are drawn in the order in which 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 renderer)

Arguments

renderer	A valid renderer object.

Returned Value

boolean

See Also

IMSMap
Renderer

Example

```
//create a value map renderer
var layer = parent.mapFrame.IMSMap.getLayer(“Zipcodes”);
var renderer1 = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
renderer1.setField(layer, “PERSONS”);
//create a group renderer
var groupRenderer = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
//add the value map renderer to the group renderer
if(groupRenderer.addRenderer(renderer1))
   alert(“renderer was added”);
```

GroupRenderer.clearRenderers

Description
Removes all renderers from the collection.

Syntax
boolean clearRenderers()

Arguments
none

Returned Value
boolean

See Also
IMSMap
Renderer

Example
```
//create a value map renderer
var layer = parent.mapFrame.IMSMap.getLayer(“Zipcodes”);
var renderer1 = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
renderer1.setField(layer, “PERSONS”);
//create a group renderer
var groupRenderer = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
//add the value map renderer to the group renderer
if(groupRenderer.addRenderer(renderer1))
   alert(“renderer was added”);

//
if (groupRenderer.clearRenderers())
   alert(“all the renderers are deleted”);
```

GroupRenderer.getSize

Description
Returns the number of renderers in the collection.

Syntax
int getSize()

Arguments
none

Returned Value
int	The total number of renderers in the collection.

See Also
IMSMap
Renderer

Example
See the next page.

GroupRenderer.getSize

Example

```
//create a value map renderer
var layer = parent.mapFrame.IMSMap.getLayer(“Zipcodes”);
var renderer1 = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
renderer1.setField(layer, “PERSONS”);
//create a group renderer
var groupRenderer = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
//add the value map renderer to the group renderer
if(groupRenderer.addRenderer(renderer1))
   alert(“renderer was added”);

//
alert(“the total number of renderers is” + groupRenderer.getSize());
```

GroupRenderer.indexOf

Description

Returns the index of a specified renderer within the collection.

Syntax

boolean indexOf(Renderer renderer)

Arguments

renderer	A valid renderer object.

Returned Value

boolean

See Also

IMSMap
Renderer

Example

```
//create a value map renderer
var layer = parent.mapFrame.IMSMap.getLayer(“Zipcodes”);
var renderer1 = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
renderer1.setField(layer, “PERSONS”);
//create a group renderer
var groupRenderer = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
//add the value map renderer to the group renderer
if(groupRenderer.addRenderer(renderer1))
  alert(“renderer was added”);

//
alert(“the index of the renderer is” + groupRenderer.indexOf(renderer1));
```

GroupRenderer.moveRenderer

Description
Moves a renderer up or down in the draw order. The “fromPosition” index specifies the old position of the renderer, and the “toPosition” specifies the new position within the collection. The map is redrawn when this method is used.

Syntax
boolean moveRenderer(int fromPosition, int toPosition)

Arguments

fromPosition	The index of the renderer to be moved.
toPosition	The new renderer index.

Returned Value
boolean

See Also
Renderer
ArcXML Programmer's Reference Guide

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

GroupRenderer.removeRenderer

Description
Removes a renderer from the collection.

Syntax
boolean removeRenderer(Renderer renderer)

Arguments

renderer	A valid renderer object.

Returned Value
boolean

See Also
IMSMap
Renderer

Example
See the next page.

GroupRenderer.removeRenderer

Example

```
//create a value map renderer
var layer = parent.mapFrame.IMSMap.getLayer(“Zipcodes”);
var renderer1 = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
renderer1.setField(layer, “PERSONS”);
//create a group renderer
var groupRenderer = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
// add the value map renderer to the group renderer
if(groupRenderer.addRenderer(renderer1))
  alert(“renderer was added”);

//
 if (groupRenderer.removeRenderer(renderer1))
  alert(“the renderer was removed”);
```

GroupRenderer.removeRendererAt

Description

Removes the renderer at a given index from the collection.

Syntax

boolean removeRendererAt(int index)

Arguments

index	Value specifying the position of the renderer to be removed.

Returned Value

boolean

See Also

Renderer
ArcXML Programmer's Reference Guide

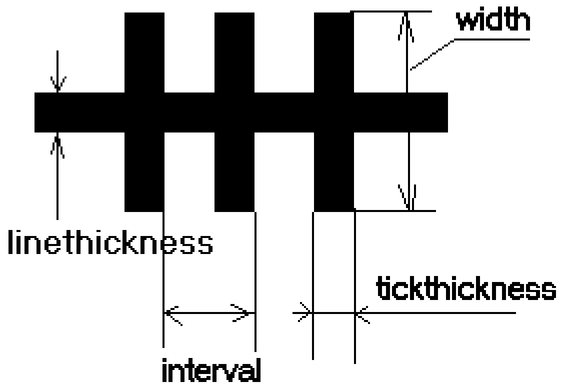
Example

```
//create a value map renderer
var layer = parent.mapFrame.IMSMap.getLayer(“Zipcodes”);
var renderer1 = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
renderer1.setField(layer, “PERSONS”);
//create a group renderer
var groupRenderer = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
// add the value map renderer to the group renderer
if(groupRenderer.addRenderer(renderer1))
  alert(“renderer was added”);

//
  if(groupRenderer.removeRendererAt(0))
  alert(“the renderer was removed”);
```

HashLineSymbol extends Symbol

The HashLineSymbol is used to draw line features. There are two styles (types) of the HashLineSymbol—background and foreground. A style is set by calling the setStyle method. If style is background, the symbol draws as a simple line without the crosshash. If the style is foreground, the symbol is drawn as a railroad. There are four properties of this symbol: width, line thickness, interval, and tick thickness; these properties define the railroad size. These properties are shown below:



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

See Also

IMSMap
Symbol
ArcXML Programmer's Reference Guide

HashLineSymbol.getAntialiasing

Description
Returns the antialiasing value of the directional lines. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax
boolean getAntialiasing()

Arguments
none

Returned Value
boolean

See Also
HashLineSymbol.setAntialiasing

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

HashLineSymbol.getColor

Description
Returns the color of the 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 = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
alert(“the default  color of the LineSymbol is” +  symbol.getColor());
```

HashLineSymbol.getInterval

Description
Returns the value of the interval property of the symbol.

Syntax
double getInterval( )

Arguments
none

Returned Value

double	The interval property in screen points. The default value is 8.

See Also
HashLineSymbol.setInterval

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

HashLineSymbol.getLineThickness

Description
Returns the value of the line thickness property of the symbol. The default value is 1.

Syntax
int getLineThickness( )

Arguments
none

Returned Value

int	The value of the line thickness property in screen pixels.

See Also
HashLineSymbol.setLineThickness

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

HashLineSymbol.getStyle

Description
Returns the style of the 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 = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
alert(“the default style of the hash line symbol is” + symbol.getStyle());
```

HashLineSymbol.getTickThickness

Description
Returns the value of the tick thickness property. The default value is 1.

Syntax
int getTickThickness( )

Arguments
none

Returned Value
int	The value of tick thickness property in screen pixels.

See Also
HashLineSymbol.setTickThickness

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
alert(“the default value of the tick thickness property is” +
symbol.getTickThickness());
```

HashLineSymbol.getTransparency

Description
Returns the transparency value of the symbol.

Syntax
double getTransparency( )

Arguments
none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
HashLineSymbol.setTransparency

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

HashLineSymbol.getWidth

Description
Returns the width of the symbol.

Syntax
int getWidth( )

Arguments
none

Returned Value

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

See Also
HashLineSymbol.setWidth

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

HashLineSymbol.setAntialiasing

Description

Sets the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance.

Syntax

boolean setAntialiasing(String enabled)

Arguments

enabled	“True” to set antialiasing on. “False” to set antialiasing off. Antialiasing is off by default.

Returned Value

boolean

See Also

HashLineSymbol.getAntialiasing

Example

```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setAntialiasing(“true”);
```

HashLineSymbol.setColor

Description

Sets the color of the symbol. The default color is black.

Syntax

boolean setColor(Color color)

Arguments

color	An instance of the color class.

Returned Value

boolean

See Also

Color
HashLineSymbol.getColor

Example

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

HashLineSymbol.setInterval

Description
Sets the interval property value of the symbol. The default value is 8.

Syntax
boolean setInterval(double intervalValue)

Arguments
intervalValue	The interval value in screen pixels.

Returned Value
boolean

See Also
HashLineSymbol.getInterval

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setInterval(5);
```

HashLineSymbol.setLineThickness

Description
Sets the value of the line thickness property. The default value is 1.

Syntax
boolean setLineThickness(int lineThickness)

Arguments
lineThickness	The line thickness value.

Returned Value
boolean

See Also
HashLineSymbol.getLineThickness

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setLineThickness(3);
```

HashLineSymbol.setStyle

Description
Sets the style of the symbol. The default style is foreground.

Syntax
boolean setStyle(int style)

Arguments

style:	0—foreground
	1—background

Returned Value
boolean

See Also
HashLineSymbol.getStyle

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setStyle(1);
```

HashLineSymbol.setTickThickness

Description
Sets the value of the tick thickness property. The default value is 1.

Syntax
boolean setTickThickness(int tickThickness)

Arguments

tickThickness	The tick thickness value in screen pixels.

Returned Value
boolean

See Also
HashLineSymbol.getTickThickness

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setTickThickness(3);
```

HashLineSymbol.setTransparency

Description
Sets the transparency value of the symbol.

Syntax
boolean setTransparency(double transparencyValue)

Arguments

transparencyValue	The transparency to be set. The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value
boolean

See Also
HashLineSymbol.getTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setTransparency(0.5);
```

HashLineSymbol.setWidth

Description
Sets the width of the hash line symbol. The default value is 6.

Syntax
boolean setWidth(double width)

Arguments

width	The width in screen pixels.

Returned Value
boolean

See Also
HashLineSymbol.getWidth

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“HASH_LINE_SYMBOL”);
symbol.setWidth(3);
```

IMSMap

Description

The applet that displays the map and functions and is a “container” for 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. 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.

See Also

IMSOverviewMap
IMSScaleBar
IMSToc
IMSToolBar

IMSMap.addAVServiceLayer

Description

Adds a layer from an ArcView IMS site.

Syntax

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

Arguments

URL	The URL to the server.
mapName	The name of the ArcView IMS map.
viewName	The name of the ArcView IMS view.
visibility	True if the layer is to be visible, false otherwise.

Returned Value

boolean

See Also

IMSMap.addServiceLayers
IMSMap.removeServiceLayers

Example

```
var imsMap= parent.mapFrame.IMSMap;
imsMap.addAVServiceLayer(“http://spacecowboy/”, “ESRICampus”, “view1”, “true”);
```

IMSMap.addLabel

Description
Adds a label 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, in map units, for placement of the label.

Returned Value
boolean

See Also
IMSMap.createAcetateLayer
IMSMap.getLabelRenderingFields

Example
```
var imsMap= parent.mapFrame.IMSMap;
var x = 711.0;
var y = 21.0;
imsMap.addPoint(x,y);
imsMap.addLabel(“The hideout”, x, y);
```

IMSMap.addLayer

Description
Adds a layer to the map.

Syntax
boolean addLayer(Layer layer)

Arguments

layer	The layer to be added.

Returned Value
boolean

Example
```
var imsMap= parent.mapFrame.IMSMap;
var newLayer = imsMap.createSDELayer(“http://entropy/”, “esri_sde”,“hsimpson”,
“doh!_nuts”, “lots”);
imsMap.addLayer(newLayer);
```

IMSMap.addMapNotesLayer

Description

Adds a new MapNotes layer. 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.

Returned Value

boolean

See Also

IMSMap.startMapNotesTool

Example

```
function addNewMapNotesLayer(layerName){
var imsMap= parent.mapFrame.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);
   }
}
```

IMSMap.addMOIMSServiceLayer

Description
Adds a layer from a MapObjects IMS site.

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

Arguments
URL	The URL to the server.
serviceName	The name of the MapObjects IMS service.
visibility	True if the layer is to be visible, false otherwise.

Returned Value
boolean

See Also
IMSMap.addServiceLayers
IMSMap.removeServiceLayers

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.addMOIMSServiceLayer("http://opu/", "bookstores", "true");
```

IMSMap.addPoint

Description
Adds a point to the acetate layer.

Syntax
boolean addPoint(double x, double y)

Arguments
x,y	The x- and y-coordinates, in map units, for placement of the point.

Returned Value
boolean

See Also
IMSMap.createAcetateLayer

Example
```
var imsMap = parent.mapFrame.IMSMap;
var x = 3.1415;
var y = 2.7182;
imsMap.addPoint(x,y);
```

IMSMap.addServiceLayers

Description
Adds all the services from an ArcIMS site.

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

Arguments

URL	The URL to the server.
name	The name of the ArcIMS site.
serviceType	0 for Feature Service, 1 for Image Service.
visibility	True if the layer is to be made visible, false otherwise.

Returned Value
boolean

See Also
IMSMap.removeServiceLayers

Example
```
var serviceType = 0;
var imsMap= parent.mapFrame.IMSMap;
imsMap.addServiceLayers(“http://twoball/”, “FrenchCompanies”, serviceType, “true”);
```

IMSMap.cancel

Description
Cancels the data retrieval for a Feature Service.

Syntax
boolean cancel()

Arguments
none

Returned Value
boolean

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

IMSMap.clearFeatureCache

Description
Clears the local cache of all Feature Service layers.

Syntax
void clearFeatureCache()

Arguments
none

Returned Value
void

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

IMSMap.clearSelections

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

Syntax
boolean clearSelections()

Arguments
none

Returned Value
boolean

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

IMSMap.closeProject

Description
Closes the project.

Syntax
boolean closeProject()

Arguments
none

Returned Value
boolean

See Also
IMSMap.loadProject

Example
```
var imsMap= parent.mapFrame.IMSMap;
imsMap.closeProject();
```

IMSMap.copyMapImageToFile

Description
Copies the map as an image file in JPEG format.

Syntax
boolean copyMapImageToFile(String filename)

Arguments
filename	Pathname of the file to be written.

Returned Value
boolean

See Also
IMSMap.copyTOCImageToFile

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

IMSMap.copyTOCImageToFile

Description
Copies the table of contents (TOC) as an image file in JPEG format.

Syntax
boolean copyTOCImageToFile(String filename)

Arguments
filename	Pathname of the file to be written.

Returned Value
boolean

See Also
IMSMap.copyMapImageToFile

Example
```
var filePath = “/spacecowboy1/pub/images/niftytoc.jpg”;
var imsMap= parent.mapFrame.IMSMap;
imsMap.copyTOCImageToFile(filePath);
```

IMSMap.createAcetateLayer

Description
Creates an acetate layer on the map. An acetate layer is for adding graphics or text on top of the map layers. If the acetate layer already exists, this method does nothing.

Syntax
boolean createAcetateLayer()

Arguments
none

Returned Value
boolean

See Also
IMSMap.removeAcetateLayer

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

IMSMap.createCollection

Description
Creates an empty collection.

Syntax
Collection createCollection()

Arguments
none

Returned Value
Collection

See Also
Collection
IMSMap.getCollectionElementAt

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

IMSMap.createColor

Description
Creates a color defined by the RGB arguments.

Syntax
Color createColor(int R, int G, int B)

Arguments
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

Returned Value
color	A new color defined by the given RGB values.

See Also
Color

Example
See the next page.

IMSMap.createColor

Example

```
var imsMap= parent.mapFrame.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();
```

IMSMap.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 the southwest corner of the desired extent.
width, height	The width and height of the desired extent.

Returned Value

Extent	Returns a new extent defined by the given values.

See Also

Extent
IMSMap.setExtent
IMSMap.setExtentLimit

Example

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

IMSMap.createFeatureServerLayer

Description
Adds a layer from a FeatureServer to the map.

Syntax
Layer createFeatureServerLayer(String url, String serviceName, String layerName)

Arguments

url	The string representation of the URL.
serviceName	The name of the service.
layerName	The name of a layer from the service.

Returned Value
Layer

See Also
IMSMap.addLayer
IMSMap.createImageServerLayer
IMSMap.redraw
Layer

Example
```
var Layer = parent.mapFrame.IMSMap.createFeatureServerLayer("http://gotti","mexico_f",
"states");
if ( Layer !=null ) {
 alert("The layer can be added to the map");
 parent.mapFrame.IMSMap.addLayer(Layer);
 parent.mapFrame.IMSMap.redraw();

}
else alert("Error : The Layer is null");
```

IMSMap.createImageFileLayer

Description
Adds a layer from an image file to the map.

Syntax
Layer createImageFileLayer(String path, String imageFile)

Arguments

path	The path to the image file.
imageFile	The name of the image file.

Returned Value
Layer

See Also
IMSMap.addLayer
IMSMap.createShapeFileLayer
IMSMap.redraw
Layer

Example
```
var layer = parent.mapFrame.IMSMap.createImageFileLayer(“c:\\data”,
“sanfran.tif”);
parent.mapFrame.IMSMap.addLayer(layer);
parent.mapFrame.IMSMap.redraw();
```

IMSMap.createImageServerLayer

Description
Adds a layer from an ImageServer to the map.

Syntax
Layer createFeatureServerLayer(String url, String serviceName)

Arguments

url	The string representation of the URL.
serviceName	The name of a service.

Returned Value
Layer

See Also
IMSMap.addLayer
IMSMap.createFeatureServerLayer
IMSMap.redraw
Layer

Example
```
var Layer = parent.mapFrame.IMSMap.createImageServerLayer(“http://
gotti”,“mexico_i”);
if ( Layer !=null ) {
 alert(“The layer can be added to the map”);
 parent.mapFrame.IMSMap.addLayer(Layer);
 parent.mapFrame.IMSMap.redraw();

}
else alert(“Error : The Layer is null”);
```

IMSMap.createNameValuePair

Description
Creates and returns a NameValuePair.

Syntax
NameValuePair createNameValuePair(String name, String value)

Arguments

name	The name of the pair.
value	The value to associate with the name.

Returned Value

NameValuePair	A NameValuePair with the given name and value.

See Also
NameValuePair

Example
```
var imsMap= parent.mapFrame.IMSMap;
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 the parameter is not recognized, then null is returned.

Syntax
Renderer createRenderer(String type)

Arguments

type	The type of renderer to be created.

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
“VALUE_LABEL_RENDERER”	ValueMapLabelRenderer

See Also
Renderer

Example
```
var imsMap= parent.mapFrame.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();
```

IMSMap.createSDELayer

Description
Creates a 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 ArcSDE server.
instance	The instance of the ArcSDE server.
userName	Username.
pswd	Password.
layerName	The name of the layer to be created.

Returned Value
Layer	The requested ArcSDE layer.

See Also
Layer

Example
```
var imsMap= parent.mapFrame.IMSMap;
 var newLayer = imsMap.createSDELayer(“http://spring/”, “port:5151”,
“hsimpson”,“doh!_nuts”, “lots”);
if(newLayer != null){
   imsMap.addLayer(newLayer);
}else{
   alert(“Layer is null.”);
}
```

IMSMap.createShapeFileLayer

Description
Adds a layer from a shapefile to the map.

Syntax
Layer createShapeFileLayer(String path, String shapeFile)

Arguments

path	The path to the shapefile.
shapeFile	The name of the shapefile.

Returned Value
Layer

See Also
IMSMap.addLayer
IMSMap.createImageFileLayer
IMSMap.redraw
Layer

Example
```
var layer = parent.mapFrame.IMSMap.createShapeFileLayer(“c:\\data”, “streets”);
parent.mapFrame.IMSMap.addLayer(layer);
parent.mapFrame.IMSMap.redraw();
```

IMSMap.createSymbol

Description
Creates a symbol object of the type defined in the parameter.

Syntax
Symbol createSymbol(String type)

Arguments

type	The type of symbol to be created.

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.

See Also
Symbol

Example
```
var imsMap= parent.mapFrame.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.
lower, upper	The lower and upper values of the range.

Returned Value

ValueRange	A ValueRange defined by the parameters.

See Also

ValueRange

Example

```
var imsMap= parent.mapFrame.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 Attributes dialog box. Once features have been selected, the Attributes dialog box displays the fields and values for those features.

Syntax
boolean displayAttributesUI()

Arguments
none

Returned Value
boolean

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

IMSMap.displayBufferUI

Description
Displays the Buffer dialog box. Once features have been selected, the Buffer dialog box is used to specify the buffer distance and buffer units to create a buffer around the selected features.

Syntax
boolean displayBufferUI()

Arguments
none

Returned Value
boolean

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

IMSMap.displayCatalogUI

Description
Displays the Catalog. The Catalog is used to add layers to a map.

Syntax
boolean displayCatalogUI()

Arguments
none

Returned Value
boolean

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayCatalogUI();
```

IMSMap.displayCatalogUI1

Description
Displays the Catalog. The Catalog is used to add layers to a map. In contrast to IMSMap.displayCatalogUI, this method allows you to control the layer visibility at the time the layer is added.

Syntax
boolean displayCatalogUI1(int LayerVisibility)

Arguments

LayerVisibility	1—The layer will be visible when added to the map. Any other value—The layer will be invisible.

Returned Value
boolean

Example
```
var imsMap = parent.mapFrame.IMSMap;
var layerVisibility = 0;
if (parent.tocFrame.IMSToc == null) layerVisibility = 1;
imsMap.displayCatalogUI1(layerVisibility);
```

IMSMap.displayFindUI

Description
Displays the Find dialog box. The Find dialog box performs a case-sensitive query on strings.

Syntax
boolean displayFindUI()

Arguments
none

Returned Value
boolean

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayFindUI();
```

IMSMap.displayGeocodeUI

Description

Displays the Locate Address dialog box. The Locate Address dialog box locates addresses on a layer given an address, street intersection, or single field input.

Syntax

boolean displayGeocodeUI()

Arguments

none

Returned Value

boolean

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayGeocodeUI();
```

IMSMap.displayGeographyNetwork

Description

Displays the Geography Network℠ interface. The Geography Network interface can be used to search and add data to the map.

Syntax

boolean displayGeographyNetwork()

Arguments

none

Returned Value

boolean

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayGeographyNetwork();
```

IMSMap.displayLayerPropertiesUI

Description
Displays the Layer Properties dialog box. The Layer Properties dialog box is used to change the properties of the selected layer.

Syntax
boolean displayLayerPropertiesUI()

Arguments
none

Returned Value
boolean

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayLayerPropertiesUI();
```

IMSMap.displayMapTipsUI

Description
Displays the MapTips dialog box. The MapTips dialog box is used to define small text popups of a field’s value.

Syntax
boolean displayMapTipsUI()

Arguments
none

Returned Value
boolean

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayMapTipsUI();
```

IMSMap.displayOpenProjectUI

Description

Displays the Open Project dialog box. The Open Project dialog box is used to open a project file.

Syntax

boolean displayOpenProjectUI()

Arguments

none

Returned Value

boolean

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayOpenProjectUI();
```

IMSMap.displayPrintUI

Description

Displays the Print dialog box. The Print dialog box is used to print the map and legend.

Syntax

boolean displayPrintUI()

Arguments

none

Returned Value

boolean

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displayPrintUI();
```

IMSMap.displayQueryBuilderUI

Description

Displays the Query Builder dialog box. The Query Builder dialog box is used to perform a query on the selected layer.

Syntax

boolean displayQueryBuilderUI()

Arguments

none

Returned Value

boolean

Example

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

IMSMap.displaySaveAsProjectUI

Description

Displays the Save dialog box. The Save dialog box is used to save the map to a project file.

Syntax

boolean displaySaveAsProjectUI()

Arguments

none

Returned Value

boolean

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.displaySaveAsProjectUI();
```

IMSMap.displayStoredQueryUI

Description

Displays the Stored Query dialog box. The Stored Query dialog box is used to save predefined queries that can be used with the Search tool.

Syntax

boolean displayStoredQueryUI()

Arguments

none

Returned Value

boolean

Example

```
if (checkSelectedLayer()) {
  parent.mapFrame.IMSMap.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)

IMSMap.enableFunction

Description

33 Select (Circle, Rectangle, Line, Polygon)
34 Find and Query Builder
36 Clear Selections
37 Stored Query

40 All Layer Actions (41, 42, 43, 44, 45, 46, 47, 48, 49)
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 MapTips
49 Geography Network

Syntax

boolean enableFunction(int functionID, boolean value)

Arguments

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

Returned Value

boolean

Example

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

IMSMap.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.IMSMap;
alert(imsMap.getAppletInfo());
```

IMSMap.getBGColor

Description
Returns the background color of the map.

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
IMSMap.setBGColor

Example
```
var imsMap = parent.mapFrame.IMSMap;
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.getCartScale

Description
Returns the map scale as an Representative Fraction (RF).

Syntax
double getCartScale()

Arguments
none

Returned Value
double	The map scale. If extent is null, 1 is returned.

See Also
IMSMap.setMapUnit

Example
```
var imsMap = parent.mapFrame.IMSMap;
var cscale =  imsMap.getCartScale();
alert(“The cartographic scale is: ” + cscale);
```

IMSMap.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
Collection
IMSMap.createCollection
NameValuePair

Example
```
var imsMap = parent.mapFrame.IMSMap;
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());
```

IMSMap.getExtent

Description
Returns the extent of the map.

Syntax
Extent getExtent()

Arguments
none

Returned Value

Extent	Returns a new extent defined by the given values.

See Also
Extent
IMSMap.getFullExtent
IMSMap.setExtent
IMSMap.setExtentLimit

Example
```
var imsMap = parent.mapFrame.IMSMap;
var currentExtent = imsMap.getExtent();
var theExtentString = “Current Extent:\n”;
theExtentString += currentExtent.getXMin() + “,” + currentExtent.getYMin() + “-”;
theExtentString += currentExtent.getXMax() + “,” + currentExtent.getYMax() + “\n”;
alert(theExtentString);
```

IMSMap.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 accessing each string using the collection’s getStringElement() method.

Syntax
Collection getFeatureLayerNames()

Arguments
none

Returned Value
Collection

See Also
Collection

Example
```
var imsMap = parent.mapFrame.IMSMap;
var layerNames = imsMap.getFeatureLayerNames();
var numLayers = layerNames.getSize();
alert(“There are” + numLayers + “Feature Layers.”);
```

IMSMap.getFullExtent

Description
Returns the full extent of the map.

Syntax
Extent getFullExtent()

Arguments
none

Returned Value

Extent	Returns a new extent defined by the given values.

See Also
Extent
IMSMap.getExtent
IMSMap.setExtent
IMSMap.setExtentLimit

Example
```
var imsMap = parent.mapFrame.IMSMap;
var fullExtent = imsMap.getFullExtent();
imsMap.setExtent(fullExtent);
```

IMSMap.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

See Also
Collection
Layer

Example
```
var imsMap = parent.mapFrame.IMSMap;
var selectedLayer = imsMap.getSelectedLayer();
var iFields = imsMap.getIdentifiableFields(selectedLayer);
var numberOfIFields = iFields.getSize();
alert(“This layer has” + numberOfIFields +  “identifiable fields.”);
```

IMSMap.getLabelRenderingFields

Description

Returns a collection consisting of all of the fields of the given layer that 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 searches for fields that are rendered by a LabelRenderer.
renderer	A valid renderer object.

Returned Value

Collection

See Also

Collection
LabelRenderer

Example

```
var imsMap = parent.mapFrame.IMSMap;
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.”);
}
```

IMSMap.getLayer

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

Syntax
Layer getLayer(String layerName)

Arguments

LayerName	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
IMSMap.getLayerCount
IMSMap.getLayerNames
IMSMap.getLayerNamesFromService

Example
```
var theNameToLookFor = “Internet_Cafes”;
var imsMap = parent.mapFrame.IMSMap;
var layer = imsMap.GetLayer(theNameToLookFor);
if(layer == null){
   alert(“Layer not found.”);
}
```

IMSMap.getLayerCount

Description
Returns the 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
IMSMap.getLayerNames
Layer

Example
```
var imsMap = parent.mapFrame.IMSMap;
var numberOfLayers = imsMap.getLayerCount();
for(int i=0; i < numberOfLayers; i++){
   imsMap.removeLayerByIndex(i);
}
```

IMSMap.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 a new extent defined by the given values.

See Also

Extent
IMSMap.getExtent
IMSMap.getFullExtent
IMSMap.setExtent
IMSMap.setExtentLimit

Example

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

IMSMap.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	The renderer of the type defined in the layer.

See Also

IMSMap.getLabelRenderingFields
IMSMap.getLayerLabelRendererType
IMSMap.setLayerLabelRenderer
LabelRenderer
ValueMapLabelRenderer

Example

```
var imsMap = parent.mapFrame.IMSMap;
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.”);
}
```

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

See Also

IMSMap.getLabelRenderingFields
IMSMap.setLayerLabelRenderer
LabelRenderer
ValueMapLabelRenderer

Example

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

IMSMap.getLayerNames

Description

Returns the names of the 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

See Also

IMSMap.getLayerCount
IMSMap.getLayerNamesFromService
Layer

Example

```
var imsMap = parent.mapFrame.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 + “.”);
```

IMSMap.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

See Also

IMSMap.getLayerCount
IMSMap.getLayerNames
Layer

Example

```
var SERVICE = “SanFranFeature”;
var URL = “http://kat/”;
var imsMap = parent.mapFrame.IMSMap;
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 + “.”);
}
```

IMSMap.getLayerRenderer

Description
Returns the renderer assigned to this layer as a renderer.

Syntax
Renderer getLayerRenderer(Layer layer)

Arguments
layer	This is the layer that is being rendered.

Returned Value
Renderer	The renderer of the type defined in the parameter. If the layer is invalid, null is returned.

See Also
IMSMap.setLayerRenderer
Renderer

Example
```
var imsMap = parent.mapFrame.IMSMap;
var layer = imsMap.getSelectedLayer();
var renderer = imsMap.getLayerRenderer(layer);
if(renderer == null){
   alert(“No renderer for layer:” + layer);
}
```

IMSMap.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	The renderer of the type defined in the parameter. If the renderer passed to this method is not a GroupRenderer or if the index is invalid, null is returned.

See Also
GroupRenderer
IMSMap.getLayerRenderer
IMSMap.getLayerRendererAtType
IMSMap.setLayerRenderer

Example
See the next page.

IMSMap.getLayerRendererAt

Example

```
var imsMap = parent.mapFrame.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 == “VALUE_LABEL_RENDERER”){
      var renderer = imsMap.getLayerRendererAt(gRenderer, INDEX);
      var symbol = imsMap.getValueMapLabelDefaultSymbol(renderer);
```

IMSMap.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_RENDERER”, “VALUEMAP_RENDERER”, “SCALE_DEPENDENT_RENDERER”, “GROUP_RENDERER”, “LABEL_RENDERER”, “VALUE_LABEL_RENDERER”.

See Also

IMSMap.getLayerRenderer
IMSMap.getLayerRendererAt
IMSMap.getLayerRendererType

Example

See the next page.

IMSMap.getLayerRendererAtType

Example

```
var imsMap = parent.mapFrame.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 == "VALUE_LABEL_RENDERER"){
      var renderer = imsMap.getLayerRendererAt(gRenderer, INDEX);
      var symbol = imsMap.getValueMapLabelDefaultSymbol(renderer);
```

IMSMap.getLayerRendererType

Description

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

Syntax

String getLayerRendererType(Layer layer)

Arguments

layer	The layer to get information from.

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

IMSMap.getLayerRenderer
IMSMap.getLayerRendererAt
IMSMap.getLayerRendererAtType

Example

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

IMSMap.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

IMSMap.addMapNotesLayer
IMSMap.selectMapNotesLayer
IMSMap.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit
IMSMap.startMapNotesTool
IMSMap.stopMapNotesTool

Example

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

IMSMap.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 a new extent defined by the given values.

See Also

Extent
IMSMap.setExtent
IMSMap.setExtentLimit

Example

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

IMSMap.getMapUnit

Description
Returns the map units of the map.

Syntax
int getMapUnit()

Arguments
none

Returned Value
int — Returns the map units as an int. The following are all the known values:

decimal_degrees	0
feet	3
meters	5

See Also
IMSMap.setMapUnit

Example
```
var imsMap = parent.mapFrame.IMSMap;
var MAP_UNIT = imsMap.getMapUnit();
```

IMSMap.getMeasureUnit

Description
Returns the measure unit.

Syntax
int getMeasureUnit()

Arguments
none

Returned Value
int — Returns the measure unit as an int. The following are all the known values:

feet	3
miles	4
meters	5
kilometers	6

See Also
IMSMap.setMeasureUnit

Example
```
var imsMap = parent.mapFrame.IMSMap;
var MEASURE_UNIT = imsMap.getMeasureUnit();
```

IMSMap.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.IMSMap;
var ovMap = imsMap.getOverviewMap();
ovMap.redraw();
```

IMSMap.getRendererSymbol

Description
Returns a new symbol for the given renderer. If that renderer is a ValueMapRenderer or a ValueMapLabelRenderer, then a nonnull 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	A valid renderer object.
value	The value used to generate the symbol. Only need be nonnull 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
IMSMap.createSymbol
IMSMap.getRendererSymbolType
Symbol

Example
See the next page.

IMSMap.getRendererSymbol

Example

```
var imsMap = parent.mapFrame.IMSMap;
var labelName = “Buried Treasure.”;
var renderer = imsMap.getLayerRenderer(imsMap.getSelectedLayer());
if(renderer != null)
   var XmarksTheSpot = imsMap.getRendererSymbol(renderer, labelName);
```

IMSMap.getRendererSymbolType

Description

Returns a new symbol for the given renderer. If that renderer is a ValueMapRenderer or a ValueMapLabelRenderer, then a nonnull 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	A valid renderer object.
value	The value used to generate the symbol object. Only need be nonnull 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 of 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

IMSMap.createSymbol
IMSMap.getRendererSymbol
Symbol

Example

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

IMSMap.getScale

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

Syntax
double getScale()

Arguments
none

Returned Value

double	The scale of the map.

See Also
IMSMap.getMapUnit
IMSMap.setMapUnit

Example
```
var imsMap = parent.mapFrame.IMSMap;
var scale = imsMap.getScale();
```

IMSMap.getScaleBar

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

Syntax
IMSScaleBar getScaleBar()

Arguments
none

Returned Value

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

See Also
IMSScaleBar

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

IMSMap.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
IMSMap.getLayer
IMSMap.setSelectedLayer
Layer

Example
```
var imsMap = parent.mapFrame.IMSMap;
var layer = imsMap.getSelectedLayer();
if(layer != null){
var type = imsMap.getLayerRendererType(layer);
alert(“The layer is rendered by a” + type);
}
```

IMSMap.getSelectedTool

Description
Returns the name of the 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
IMSMap.selectTool

Example
```
var imsMap = parent.mapFrame.IMSMap;
var selectedTool = imsMap.getSelectedTool();
if(selectedTool == “ZOOM_IN”){
   alert(“Zoom in tool is active.”);
}
```

IMSMap.getShowStackTrace

Description
Returns the 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 IMSMap as the constants: SHOW_NO_STACK_TRACES, SHOW_RUNTIME_STACK_TRACES, SHOW_ALL_STACK_TRACES.

See Also
IMSMap.setShowStackTrace

Example
```
var imsMap = parent.mapFrame.IMSMap;
var traceLevel = imsMap.getShowStackTrace();
```

IMSMap.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

See Also

Collection

Example

```
function getLayerStoredQueries() {
var imsMap = parent.mapFrame.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

```
var imsMap = parent.mapFrame.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);
      }
   }
}
```

IMSMap.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

See Also
Collection
IMSMap.getSubLayer

Example
```
var imsMap = parent.mapFrame.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);
        }
   }
}
```

IMSMap.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.IMSMap;
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 that 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.IMSMap;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerRendererType(layer);

if(rendererType == “VALUEMAP_RENDERER”){
   var renderer = imsMap.getLayerRenderer(layer);
   var symbol = imsMap.getValueMapDefaultSymbol(renderer);
}
```

IMSMap.getValueMapDefaultSymbolType

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

Syntax
String 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 of the following 13 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
IMSMap.getValueMapDefaultSymbol
IMSMap.getValueMapLabelDefaultSymbol
IMSMap.getValueMapLabelDefaultSymbolType
ValueMapRenderer

Example
```
var imsMap = parent.mapFrame.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”){
   }
}
```

IMSMap.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

IMSMap.getValueMapDefaultSymbol
IMSMap.getValueMapDefaultSymbolType
IMSMap.getValueMapLabelDefaultSymbolType
ValueMapRenderer

Example

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

if(rendererType == “VALUE_LABEL_RENDERER”){
   var renderer = imsMap.getLayerLabelRenderer(layer);
   var symbol = imsMap.getValueMapLabelDefaultSymbol(renderer);
}
```

IMSMap.getValueMapLabelDefaultSymbolType

Description

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

Syntax

String 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 of the following 13 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

IMSMap.getValueMapDefaultSymbol
IMSMap.getValueMapDefaultSymbolType
IMSMap.getValueMapLabelDefaultSymbol
ValueMapRenderer

Example

```
var imsMap = parent.mapFrame.IMSMap;
var layer = imsMap.getSelectedLayer();
var rendererType = imsMap.getLayerLabelRendererType(layer);
if(rendererType == “VALUE_LABEL_RENDERER”){
   var renderer = imsMap.getLayerLabelRenderer(layer);
   var symbolType = imsMap.getValueMapLabelDefaultSymbolType(renderer);
}
```

IMSMap.getWrappedRenderer

Description

Returns the renderer that is being wrapped by a ScaleDependentRenderer.

Syntax

Renderer getWrappedRenderer(Layer layer, ScaleDependentRenderer scaleRenderer)

Arguments

layer	The layer for which information is desired.
scaleRenderer	The ScaleDependentRenderer that is wrapping another Renderer.

Returned Value

Renderer	The renderer of the type defined in the parameter. If either argument is invalid, null is returned.

See Also

IMSMap.getWrappedRendererType

Example

```
var imsMap = parent.mapFrame.IMSMap;
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);
if(wrapRendType == “SIMPLE_RENDERER”){
}
```

IMSMap.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 that 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”, “VALUE_LABEL_RENDERER”.

See Also

IMSMap.getWrappedRenderer

Example

```
var imsMap = parent.mapFrame.IMSMap;
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”){
   alert(“Renderer is:” + wrapRendType);
}
```

IMSMap.goBackToExtent

Description
Changes the extent of the map to the previous extent if one exists.

Syntax
boolean goBackToExtent()

Arguments
none

Returned Value
boolean

See Also
IMSMap.goForwardToExtent

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.goBackToExtent();
```

IMSMap.goForwardToExtent

Description
Changes the extent of the map to the next extent if one exists.

Syntax
boolean goForwardToExtent()

Arguments
none

Returned Value
boolean

See Also
IMSMap.goBackToExtent

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.goForwardToExtent();
```

IMSMap.isEditNotesModified

Description
Checks if the EditNotes tool is open and if a change has been made since the last submit.

Syntax
boolean isEditNotesModified()

Arguments
none

Returned Value
boolean

See Also
IMSMap.selectEditNotesToolFunction
IMSMap.selectTool
IMSMap.startEditNotesTool
IMSMap.stopEditNotesTool

Example
```
var imsMap = parent.mapFrame.IMSMap;
if (imsMap.isEditNotesModified()) {
alert(“EditNote has been modified.”);
}
```

IMSMap.isMapNotesEditable

Description
Checks if the MapNotes session is editable.

Syntax
boolean isMapNotesEditable()

Arguments
none

Returned Value
boolean

See Also
IMSMap.addMapNotesLayer
IMSMap.selectMapNotesLayer
IMSMap.setMapNotesEditable
IMSMap.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit

Example
```
var imsMap = parent.mapFrame.IMSMap;
if (!imsMap.isMapNotesEditable()) {
alert(“Cannot edit the MapNotes.”);
}
```

IMSMap.isProjectModified

Description

Checks if the project has been modified since the last time it was saved.

Syntax

boolean isProjectModified()

Arguments

none

Returned Value

boolean

See Also

IMSMap.closeProject
IMSMap.loadMapOnlyProject
IMSMap.loadProject
IMSMap.saveProject

Example

```
var imsMap = parent.mapFrame.IMSMap;
if (parent.mapFrame.IMSMap.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 from a given URL. Only the map information is loaded. Information for the overview map, toolbar, TOC, and scalebar is not loaded.

Syntax
boolean loadMapOnlyProject(String url)

Arguments

url	The URL of the map-only project to load.

Returned Value
boolean

See Also

IMSMap.closeProject
IMSMap.loadProject
IMSMap.saveProject

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.loadMapOnlyProject(“http://headhunter/projects/survival.axl”);
```

IMSMap.loadProject

Description
Loads a project file from a given URL.

Syntax
boolean loadProject(String url)

Arguments

url	The URL of the project to load.

Returned Value
boolean

See Also
IMSMap.closeProject
IMSMap.loadMapOnlyProject
IMSMap.saveProject

Example
```
var imsMap = parent.mapFrame.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 moveSelectedLayer(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

See Also
IMSMap.addLayer
IMSMap.removeLayer

Example
```
var imsMap = parent.mapFrame.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

See Also
IMSMap.zoom
IMSMap.zoomIn
IMSMap.zoomOut

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.pan(“PAN_EAST”);
```

IMSMap.redraw

Description
Redraws 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

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.removeSelectedLayer();
imsMap.redraw();
```

IMSMap.removeAcetateLayer

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

Syntax
boolean removeAcetateLayer()

Arguments
none

Returned Value
boolean

See Also
IMSMap.createAcetateLayer

Example
```
var imsMap = parent.mapFrame.IMSMap;
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

See Also
IMSMap.addLayer
IMSMap.removeLayerByIndex
IMSMap.removeServiceLayers

Example
```
function removeAndTell(){
   var imsMap = parent.mapFrame.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

See Also
IMSMap.addLayer
IMSMap.removeLayer
IMSMap.removeServiceLayers

Example
```
function removeAndTell(index){
   var imsMap = parent.mapFrame.IMSMap;
   imsMap.removeLayerByIndex(index);
   imsMap.redraw();

   alert("Layer at index " + index + " removed.");
}
```

IMSMap.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

See Also
IMSMap.addLayer
IMSMap.removeLayer

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

IMSMap.saveMapOnlyProject

Description
Saves the project to a map configuration 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

See Also
IMSMap.loadMapOnlyProject
IMSMap.loadProject
IMSMap.saveProject

Example
```
var imsMap = parent.mapFrame.IMSMap;
var DEFAULT_FILE = “D:\\ArcIMS\\Projects\\MapOnly\\default.axl”;
imsMap.saveMapOnlyProject(DEFAULT_FILE);
```

IMSMap.saveProject

Description
Saves the project to a map configuration file.

Syntax
boolean saveProject(String path)

Arguments
path	The path of the file to be saved.

Returned Value
boolean

See Also
IMSMap.loadMapOnlyProject
IMSMap.loadProject
IMSMap.saveMapOnlyProject

Example
```
var imsMap = parent.mapFrame.IMSMap;
var DEFAULT_FILE = “D:\\ArcIMS\\Projects\\projects\\temp.axl”;
imsMap.saveProject(DEFAULT_FILE);
```

IMSMap.selectEditNotesToolFunction

Description

Selects the mode of the EditNotes tool.

Syntax

boolean selectEditNotesToolFunction(String mode)

Arguments

mode	The function of the EditNotes tool. 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

See Also

IMSMap.selectTool
IMSMap.startEditNotesTool
IMSMap.stopEditNotesTool

Example

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

IMSMap.selectMapNotesLayer

Description
Selects an active MapNotes layer.

Syntax
boolean selectMapNotesLayer(String name)

Arguments

name	The name of the MapNotes layer to be selected.

Returned Value
boolean

See Also
IMSMap.addMapNotesLayer
IMSMap.getMapNotesLayer
IMSMap.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit
IMSMap.startMapNotesTool
IMSMap.stopMapNotesTool

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.selectMapNotesLayer(“GarageSaleAt_380NewYork_Redlands”);
```

IMSMap.selectMapNotesToolFunction

Description

Selects the mode of the MapNotes tool.

Syntax

boolean selectMapNotesToolFunction(String mode)

Arguments

mode	The function of the MapNotes tool. 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

See Also

IMSMap.addMapNotesLayer
IMSMap.getMapNotesLayers
IMSMap.selectMapNotesLayer
IMSMap.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit
IMSMap.startMapNotesTool
IMSMap.stopMapNotesTool

Example

```
function selectMNTool(mode){
   var imsMap = parent.mapFrame.IMSMap;
if (mode==0) {
imsMap.selectMapNotesToolFunction(“MAPNOTES_TOOL_SELECT_FEATURES”);
} else if (mode==1) {
}
```

IMSMap.selectTool

Description

Selects 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

See Also

IMSMap.selectEditNotesToolFunction
IMSMap.selectMapNotesToolFunction

Example

```
function selectTool(mode){
   var imsMap = parent.mapFrame.IMSMap;
if (mode==0)
   imsMap.selectTool(“ZOOM_IN_TOOL”)
}
```

IMSMap.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

See Also
IMSMap.getBGColor

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

IMSMap.setDefaultProjectDir

Description
Sets the default directory to be used for opening and closing 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

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setDefaultProjectDir(“d://myAXLs”);
```

IMSMap.setEditNotesDescription

Description

Sets the EditNotes description and submits the EditNotes session. After submitting, the EditNotes session is still active.

Syntax

boolean setEditNotesDescription(String description)

Arguments

description	A description of the EditNotes session being submitted.

Returned Value

boolean

See Also

IMSMap.selectEditNotesToolFunction
IMSMap.selectTool
IMSMap.setEditNotesFolder
IMSMap.startEditNotesTool
IMSMap.stopEditNotesTool

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setEditNotesDescription(“Florida after 10% polar cap melt.”);
```

IMSMap.setEditNotesFolder

Description

Sets the EditNotes folder for a session. The EditNotes folder is created through ArcIMS Administrator.

Syntax

boolean setEditNotesFolder(String server, String name)

Arguments

server	The name of the server where the EditNotes folder is. The server can be specified as: “servername” (In this case, http is assumed.) “http://servername/” “https://servername/”
name	The name of the EditNotes folder.

Returned Value

boolean

See Also

IMSMap.selectEditNotesToolFunction
IMSMap.selectTool
IMSMap.setEditNotesDescription
IMSMap.startEditNotesTool
IMSMap.stopEditNotesTool

Example

```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setEditNotesFolder(“http://servername/”, “Edits”);
```

IMSMap.setExtent

Description

Sets the extent of the map.

Syntax

boolean setExtent(Extent extent)

Arguments

extent	Returns a new extent defined by the given values.

Returned Value

boolean

See Also

Extent
IMSMap.getExtent
IMSMap.getFullExtent
IMSMap.setExtentLimit

Example

```
var MINWIDTH = 10;
var MINHEIGHT = 10;
var imsMap = parent.mapFrame.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);
}
```

IMSMap.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

See Also
Extent
IMSMap.getExtent
IMSMap.getFullExtent
IMSMap.setExtent

Example
```
var imsMap = parent.mapFrame.IMSMap;
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
Sets the label renderer for a given layer.

Syntax
boolean setLayerLabelRenderer(Layer layer, Renderer renderer)

Arguments

layer	The layer to have the labels rendered on.
renderer	A valid renderer object.

Returned Value
boolean

See Also
IMSMap.createRenderer
IMSMap.setLayerRenderer
LabelRenderer
Layer
Renderer
ValueMapLabelRenderer

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

IMSMap.setLayerRenderer

Description

Sets the renderer for a given layer.

Syntax

boolean setLayerRenderer(Layer layer, Renderer renderer)

Arguments

layer	The layer to be rendered.
renderer	A valid renderer object.

Returned Value

boolean

See Also

IMSMap.setLayerLabelRenderer
Layer
Renderer

Example

```
var imsMap= parent.mapFrame.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();
```

IMSMap.setMapNotesEditable

Description

Sets the ability for the user to edit during the 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

See Also

IMSMap.isMapNotesEditable

Example

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

IMSMap.setMapNotesFolder

Description
Sets the name of the MapNotes folder.

Syntax
boolean setMapNotesFolder(String name)

Arguments

name	The name of the MapNotes folder to be used for the session.

Returned Value
boolean

See Also
IMSMap.addMapNotesLayer
IMSMap.getMapNotesLayer
IMSMap.selectMapNotesLayer
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit
IMSMap.setMapNotesToolFunction
IMSMap.startMapNotesTool
IMSMap.stopMapNotesTool

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setMapNotesFolder(“myMapNotes”);
```

IMSMap.setMapNotesFolderOnServer

Description
Sets the name and server for the MapNotes folder.

Syntax
boolean setMapNotesFolderOnServer(String server, String name)

Arguments

server	The name of the server where the EditNotes folder is. The server can be specified as: “servername” (In this case, http is assumed.) “http://servername/” “https://servername/”
name	The name of the MapNotes folder.

Returned Value
boolean

See Also
IMSMap.addMapNotesLayer
IMSMap.getMapNotesLayer
IMSMap.selectMapNotesLayer
IMSMap.setMapNotesFolder
IMSMap.setMapNotesSubmitLimit
IMSMap.setMapNotesToolFunction
IMSMap.startMapNotesTool
IMSMap.stopMapNotesTool

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setMapNotesFolderOnServer(“http://servername/”, “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 nonimage edits to a MapNote.

Syntax

void setMapNotesSubmitLimit(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.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesToolFunction
IMSMap.startMapNotesTool
IMSMap.stopMapNotesTool

Example

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

IMSMap.setMapUnit

Description
Sets the map units.

Syntax
boolean setMapUnit(String unit)

Arguments

unit	The following strings are all the known values (invalid values are ignored): “decimal_degrees” “feet” “meters”

Returned Value
boolean

See Also
IMSMap.getMapUnit

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setMapUnit(“feet”);
```

IMSMap.setMeasureUnit

Description
Sets the measure unit for the map.

Syntax
boolean setMeasureUnit(String unit)

Arguments

unit	The following strings are all the known values (invalid values are ignored): “feet” “miles” “meters” “kilometers”

Returned Value
boolean

See Also
IMSMap.getMeasureUnit

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setMeasureUnit(“meters”);
```

IMSMap.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

See Also
IMSMap.getSelectedLayer

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setSelectedLayer(“redlands_pools”);
```

IMSMap.setSelectionLineSymbol

Description
Sets the symbol for displaying selected line features.

Syntax
boolean setSelectionLineSymbol(LineSymbol selectionSymbol)

Arguments

selectionSymbol	The new symbol to be used for selected line features.

Returned Value
boolean

See Also
IMSMap.setSelectionMarkerSymbol
IMSMap.setSelectionPolygonSymbol
Layer.setSelectionSymbol

Example
```
var imsMap = parent.mapFrame.IMSMap;
var ls = imsMap.createSymbol(“LINE_SYMBOL”);
var cl2 = imsMap.createColor(0,255,0);
ls.setColor(cl2);
ls.setWidth(6);
imsMap.setSelectionLineSymbol(ls);
imsMap.displayFindUI();
```

IMSMap.setSelectionMarkerSymbol

Description
Sets the symbol for display of selected point features.

Syntax
boolean setSelectionMarkerSymbol(MarkerSymbol selectionSymbol)

Arguments

selectionSymbol	The new symbol to be used for selected point features.

Returned Value
boolean

See Also
IMSMap.setSelectionLineSymbol
IMSMap.setSelectionPolygonSymbol
Layer.setSelectionSymbol

Example
See the next page.

IMSMap.setSelectionMarkerSymbol

Example

```
var imsMap = parent.mapFrame.IMSMap;
var ms = imsMap.createSymbol(“MARKER_SYMBOL”);
var cl2 = imsMap.createColor(0,255,0);
ms.setColor(cl2);
ms.setSize(10);
imsMap.setSelectionMarkerSymbol(ms);
imsMap.displayFindUI();
```

IMSMap.setSelectionPolygonSymbol

Description

Sets the symbol for display of selected polygon features.

Syntax

boolean setSelectionPolygonSymbol(PolygonSymbol selectionSymbol)

Arguments

selectionSymbol	The new symbol to be used for selected polygon features.

Returned Value

boolean

See Also

IMSMap.setSelectionLineSymbol
IMSMap.setSelectionMarkerSymbol
Layer.setSelectionSymbol

Example

```
var imsMap = parent.mapFrame.IMSMap;
var ps = imsMap.createSymbol(“POLYGON_SYMBOL”);
var cl2 = imsMap.createColor(0,255,0);
ps.setFillColor(cl2);
imsMap.setSelectionPolygonSymbol(ps);
imsMap.displayFindUI();
```

IMSMap.setShowStackTrace

Description
Sets the 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
IMSMap.getShowStackTrace

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.setShowStackTrace(imsMap.SHOW_NO_STACK_TRACES);
```

IMSMap.startEditNotesTool

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

Syntax
boolean startEditNotesTool()

Arguments
none

Returned Value
boolean

See Also
IMSMap.selectEditNotesToolFunction
IMSMap.selectTool
IMSMap.setEditNotesDescription
IMSMap.setEditNotesFolder
IMSMap.stopEditNotesTool

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.startEditNotesTool();
```

IMSMap.startMapNotesTool

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

Syntax
boolean startMapNotesTool()

Arguments
none

Returned Value
boolean

See Also
IMSMap.addMapNotesLayer
IMSMap.getMapNotesLayers
IMSMap.selectMapNotesToolFunction
IMSMap.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit
IMSMap.stopMapNotesTool

Example
```
var imsMap = parent.mapFrame.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

See Also
IMSMap.selectEditNotesToolFunction
IMSMap.selectTool
IMSMap.setEditNotesDescription
IMSMap.setEditNotesFolder
IMSMap.startEditNotesTool

Example
```
var imsMap = parent.mapFrame.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

See Also

IMSMap.addMapNotesLayer
IMSMap.getMapNotesLayers
IMSMap.selectMapNotesToolFunction
IMSMap.setMapNotesFolder
IMSMap.setMapNotesFolderOnServer
IMSMap.setMapNotesSubmitLimit
IMSMap.startMapNotesTool

Example

```
var imsMap = parent.mapFrame.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 is equal to 1, there is no change. If zoom factor is equal to or less than 0, there is no effect.

Returned Value
boolean

See Also
IMSMap.zoomIn
IMSMap.zoomOut
IMSMap.zoomToCartScale
IMSMap.zoomToScale
IMSMap.zoomToSelectedLayer
IMSMap.zoomToSelection

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.zoom(1.6180339887);
```

IMSMap.zoomIn

Description
Zooms in by a factor of the parameter.

Syntax
boolean zoomIn(int factor)

Arguments

factor	Factor to zoom in by.

Returned Value
boolean

See Also
IMSMap.zoom
IMSMap.zoomOut
IMSMap.zoomToCartScale
IMSMap.zoomToScale
IMSMap.zoomToSelectedLayer
IMSMap.zoomToSelection

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.zoomIn(3);
```

IMSMap.zoomOut

Description
Zooms out by a factor of the parameter.

Syntax
boolean zoomOut(int factor)

Arguments

factor	Factor to zoom out by.

Returned Value
boolean

See Also
IMSMap.zoom
IMSMap.zoomIn
IMSMap.zoomToCartScale
IMSMap.zoomToScale
IMSMap.zoomToSelectedLayer
IMSMap.zoomToSelection

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.zoomOut(5);
```

IMSMap.zoomToCartScale

Description
Zoom to the given scale where the scale is a cartographic scale defined by the RF.

Syntax
boolean zoomToCartScale(int factor)

Arguments

factor	Factor to zoom by.

Returned Value
boolean

See Also
IMSMap.zoom
IMSMap.zoomIn
IMSMap.zoomOut
IMSMap.zoomToScale
IMSMap.zoomToSelectedLayer
IMSMap.zoomToSelection

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.zoomToCartScale(12000);
```

IMSMap.zoomToScale

Description
Zooms to scale where scale is map units per pixel.

Syntax
boolean zoomToScale(double scale)

Arguments

scale	Scale to zoom to.

Returned Value
boolean

See Also
IMSMap.zoom
IMSMap.zoomIn
IMSMap.zoomOut
IMSMap.zoomToCartScale
IMSMap.zoomToSelectedLayer
IMSMap.zoomToSelection

Example
```
var imsMap = parent.mapFrame.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

See Also
IMSMap.zoom
IMSMap.zoomIn
IMSMap.zoomOut
IMSMap.zoomToCartScale
IMSMap.zoomToScale
IMSMap.zoomToSelection

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.zoomToSelectedLayer();
```

IMSMap.zoomToSelection

Description
Zooms to the extent 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

See Also
IMSMap.zoom
IMSMap.zoomIn
IMSMap.zoomOut
IMSMap.zoomToCartScale
IMSMap.zoomToScale
IMSMap.zoomToSelectedLayer

Example
```
var imsMap = parent.mapFrame.IMSMap;
imsMap.zoomToSelection(“Cities”, “POPULATION > 9000000”);
```

IMSOverviewMap

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 the 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 the removeLayer method. The extent of the map is defined by a rectangular frame that by default is half transparent and red. 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

IMSOverviewMap.addLayer

Description

Adds a layer to the overview map. The layer is displayed in the overview map when the IMSOverviewMap.redraw() method has been executed.

Syntax

boolean addLayer(String layerName);

Arguments

layerName	The name of the layer to be added.

Returned Value

boolean

See Also

IMSOverviewMap.addMapLayer
IMSOverviewMap.layerExists
IMSOverviewMap.removeLayer

Example

```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
var layerNames = parent.mapFrame.IMSMap.getLayerNames();
var layerName = layerNames.get(1);
if (overviewMap.addLayer(layerName))
  alert(layerName + “layer will be added to the overview map”);
overviewMap.redraw();
```

IMSOverviewMap.addMapLayer

Description

Adds a layer to the overview map. The layer is displayed in the overview map when the IMSOverviewMap.redraw() method has been executed.

Syntax

boolean addLayer(Layer layer);

Arguments

layer	A layer being added.

Returned Value

boolean

See Also

IMSOverviewMap.addLayer
IMSOverviewMap.layerExists
IMSOverviewMap.removeLayer

Example

```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
  var layernames = parent.mapFrame.IMSMap.getLayerNames();
  var layerName = layerNames.get(1);
  var layer = parent.mapFrame.IMSMap.getLayer(layerName);
  if (overviewMap.addMapLayer(layer))
    alert(layerName + “layer will be added to the overview map”);
  overviewMap.redraw();
```

IMSOverviewMap.getBGColor

Description

Returns the background color of the overview map.

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

IMSOverviewMap.setBGColor

Example

```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
 var layerNames = parent.mapFrame.IMSMap.getLayerNames();
 var layerName = layerNames.get(1);
 if ( overviewMap.addLayer(layerName))
   alert(layerName + “layer will be added to the overview map”);
 overviewMap.redraw();

 //
 alert (“the background color of the overview map is” + overviewMap.getBGColor());
```

IMSOverviewMap.getRectangleFillColor

Description
Returns the fill color of the rectangular frame on the overview map.

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
IMSOverviewMap.setRectangleFillColor

Example
```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
 var layerNames = parent.mapFrame.IMSMap.getLayerNames();
 var layerName = layerNames.get(1);
 if (overviewMap.addLayer(layerName))
   alert(layerName + “layer will be added to the overview map”);
 overviewMap.redraw();

 //
 alert (“the fill color of the overview map rectangular frame is” +
overviewMap.getRectangleFillColor());
```

IMSOverviewMap.getRectangleOutlineColor

Description
Returns the outline color of the rectangular frame on the overview map.

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
IMSOverviewMap.setRectangleOutlineColor

Example
```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
 var layerNames = parent.mapFrame.IMSMap.getLayerNames();
 var layerName = layerNames.get(1);
 if (overviewMap.addLayer(layerName))
   alert(layerName + “layer will be added to the overviewmap”);
overviewMap.redraw();

 //
 alert (“the outline color of the overview map rectangular frame is” +
overviewMap.getRectangleOutlineColor());
```

IMSOverviewMap.layerExists

Description

Checks to see if a particular layer is in the overview map.

Syntax

boolean layerExists(Layer layer)

Arguments

layer	The layer being checked.

Returned Value

boolean

See Also

IMSOverviewMap.addLayer
IMSOverviewMap.addMapLayer
IMSOverviewMap.removeLayer

Example

```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
 var layerNames = parent.mapFrame.IMSMap.getLayerNames();
 var layerName = layerNames.get(1);
 var layer = parent.mapFrame.IMSMap.getLayer(layerName);
 if (overviewMap.addMapLayer(layer))
   alert(layerName + “layer will be added to the overview map”);
 overviewMap.redraw();

 //
 if(overviewMap.layerExists(layer))
   alert(“The layer of the  ”+ layerName +“ is in the layer set”);
```

IMSOverviewMap.redraw

Description
Redraws the overview map to reflect changes.

Syntax
boolean redraw();

Arguments
none

Returned Value
boolean

See Also
IMSOverviewMap

Example
```
alert(overviewFrame.IMSOverviewMap.redraw());
```

IMSOverviewMap.removeLayer

Description

Removes a layer from the overview map. The layer is removed when the IMSOverviewMap.redraw() method has been executed.

Syntax

boolean removeLayer(Layer layer);

Arguments

layer	A layer being removed.

Returned Value

boolean

See Also

IMSOverviewMap.addLayer
IMSOverviewMap.addMapLayer
IMSOverviewMap.layerExists

Example

```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
 var layerNames = parent.mapFrame.IMSMap.getLayerNames();
 var layerName = layerNames.get(1);
 var layer = parent.mapFrame.IMSMap.getLayer(layerName);
 if (overviewMap.addMapLayer(layer))
   alert(layerName + “layer will be added to the overview map”);
 overviewMap.redraw();

 //
 if(overviewMap.layerExists(layer)) (
   alert(“The layer of the  ”+ layerName +“ is in the layer set”);
    //remove the layer
   overviewMap.removeLayer(layer);
}
if(overviewMap.layerExists(layer))
   alert(“The layer of the  ”+ layerName +“ is in the layer set”);
else
  alert(“The layer of the  ”+ layerName +“ is not in the layer set”);
overviewMap.redraw();
```

IMSOverviewMap.setBGColor

Description
Sets the background color of the overview map. The color changes when the IMSOverviewMap.redraw() method has been executed.

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
boolean

See Also
IMSOverviewMap.getBGColor
IMSOverviewMap.redraw

Example
```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
 var layerNames = parent.mapFrame.IMSMap.getLayerNames();
 var layerName = layerNames.get(1);
 if (overviewMap.addLayer(layerName))
   alert(layerName + “layer will be added to the overview map”);

 //
 if(overviewMap.setBGColor(0,255,0))
   alert(“the background color will be set to green”);
 overviewMap.redraw();
```

IMSOverviewMap.setMap

Description
Sets an IMSMap to be used with the overview map 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
boolean

Example
```
alert(“reset the same IMSMap object that is already used”);
alert(overviewFrame.IMSOverviewMap.setMap(mapFrame.IMSMap));
overviewFrame.IMSOverviewMap.redraw();
```

IMSOverviewMap.setRectangleFillColor

Description

Sets a fill color of the rectangular frame on the overview map. The color is changed when the overview map redraw() method has been executed.

Syntax

boolean setRectangleFillColor(int R, int G, int B, int A)

Arguments

R	defines the red part of the RGBA color value, should be between 0 and 255
G	defines the green part of the RGBA color value, should be between 0 and 255
B	defines the blue part of the RGBA color value, should be between 0 and 255
A	defines the transparency of the RGBA color value, should be between 0 and 255

Returned Value

boolean

See Also

IMSOverviewMap.getRectangleFillColor

Example

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

   //
  alert("setting map rectangular frame fill color");
  if( overviewMap.setRectangleFillColor(0,0,255,255) )
   alert(" the fill color of the overview map rectangular frame will be set to  " +
overviewMap.getRectangleFillColor());
overviewMap.redraw();
```

IMSOverviewMap.setRectangleOutlineColor

Description

Sets an outline color of the rectangular frame on the overview map. The color is changed when the overview map redraw() method had been executed.

Syntax

boolean setRectangleOutlineColor(int R, int G, int B, int A)

Arguments

R	defines the red part of the RGBA color value, should be between 0 and 255
G	defines the green part of the RGBA color value, should be between 0 and 255
B	defines the blue part of the RGBA color value, should be between 0 and 255
A	defines the transparency of the RGBA color value, should be between 0 and 255

Returned Value

boolean

See Also

IMSOverviewMap.getRectangleOutlineColor

Example

```
var overviewMap = parent.overviewFrame.IMSOverviewMap;
  var layerNames = parent.mapFrame.IMSMap.getLayerNames();
  var layerName = layerNames.get(1);
  if ( overviewMap.addLayer(layerName))
       overviewMap.redraw();
   //
  alert(“setting map rectangular frame outline color”);
  if( overviewMap.setRectangleOutlineColor(0,0,255,255) )
   alert(“ the outline color of the overview map rectangular frame will be set to  ” +
overviewMap.getRectangleOutlineColor());
overviewMap.redraw();
```

IMSScaleBar

The IMSScaleBar provides methods to customize a scalebar panel. It extends javax.swing.JApplet. The scalebar panel is used to display information about the scale used by the IMSMap applet. It is placed below the IMSMap window on the ArcIMS Java Custom Applet page. The environment creates an 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 background color of the scalebar.

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 background color of the IMSScaleBar panel is" +
parent.scalebarFrame.IMSScaleBar.getBGColor());
```

IMSScaleBar.getFGColor

Description
Returns the text color of the scalebar.

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 foreground color of the IMSScaleBar panel is" +
parent.scalebarFrame.IMSScaleBar.getFGColor());
```

IMSScaleBar.getScaleUnit

Description
Returns the scale units for the scalebar.

Syntax
int getScaleUnit()

Arguments
none

Returned Value
3—Feet
4—Miles
5—Meters
6—Kilometers

See Also
IMSScaleBar.getScaleUnit

Example
```
alert("the scale unit of the IMSScaleBar panel is" +
parent.scalebarFrame.IMSScaleBar.getScaleUnit());
```

IMSScaleBar.getScreenUnit

Description
Returns the screen units for the scalebar.

Syntax
int getScreenUnit()

Arguments
none

Returned Value
1—centimeters
2—inches

See Also
IMSScaleBar.setScreenUnit

Example
```
alert(“the screen unit of the IMSScaleBar panel is” +
parent.scalebarFrame.IMSScaleBar.getScreenUnit());
```

IMSScaleBar.refresh

Description
Refreshes the scalebar.

Syntax
boolean refresh();

Arguments
none

Returned Value
boolean

See Also
IMSScaleBar

Example
```
alert(parent.scalebarFrame.IMSScaleBar.refresh());
```

IMSScaleBar.setBGColor

Description
Sets the background color of the scalebar.

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
boolean

See Also
IMSScaleBar.getBGColor

Example
```
alert("set the background color to white");
parent.scalebarFrame.IMSScaleBar.setBGColor(255,255,255));
alert("the background color is" + parent.scalebarFrame.IMSScaleBar.getBGColor());
```

IMSScaleBar.setFGColor

Description
Sets the text color for the scalebar.

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
boolean

See Also
IMSScaleBar.getFGColor

Example
```
alert("set the foreground color to white");
parent.scalebarFrame.IMSScaleBar.setFGColor(255,255,255));
alert("the foreground color is " + parent.scalebarFrame.IMSScaleBar.getFGColor());
```

IMSScaleBar.setScaleUnit

Description
Sets the scale units for the scalebar.

Syntax
boolean setScaleUnit(String scaleUnit)

Arguments
scaleUnit “FEET”, “METERS”, “KILOMETERS”, “MILES”

Returned Value
boolean

See Also
IMSScaleBar.getScaleUnit

Example
```
alert(parent.scalebarFrame.IMSScaleBar.setScaleUnit(“METERS”));
```

IMSScaleBar.setScreenUnit

Description
Sets the screen units for the scalebar.

Syntax
boolean setScreenUnit(String screenUnit)

Arguments
screenUnit “INCHES” , “CENTIMETERS”

Returned Value
boolean

See Also
IMSScaleBar.setScreenUnit

Example
```
alert(parent.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 table of contents. Legends can be "selected" by clicking on them. A layer becomes the active layer if its legend is selected. You can customize a table of contents by setting such properties as its background color.

The IMSToc applet will register itself with the IMSMap applet.

The instance of the IMSToc can be accessed through the IMSMap.getToc() method.

See Also
IMSMap
Layer

IMSToc.getBGColor

Description
Returns the background color of the table of contents.

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 background color of the ToC panel is" +
parent.tocFrame.IMSToc.getBGColor());
```

IMSToc.getFGColor

Description

Returns the text color for the table of contents.

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

IMSToc.setFGColor

Example

```
alert(“set the red color of a layer legend text on the TOC”);
 if (parent.tocFrame.IMSToc.setFGColor(255,0,0))
   alert(“the current color of a layer legend text is” +
parent.tocFrame.IMSToc.getFGColor());
 else alert(“Error occurred while setting color”);
```

IMSToc.getSelectedLayer

Description

Returns the layer selected in the table of contents.

Syntax

Layer getSelectedLayer()

Arguments

none

Returned Value

Layer	The selected layer, null if no layer is selected.

Example

```
var layer = parent.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 table of contents.

Syntax
boolean isDrawBorderEnabled()

Arguments
none

Returned Value
boolean

See Also
IMSToc.setDrawBorderEnabled

Example
```
var IMSToc = parent.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	The layer being checked.

Returned Value
boolean

See Also
IMSToc.setLayerExpanded

Example
```
var IMSToc = parent.mapFrame.IMSMap.getToc();
var layer = parent.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 table of contents.

Syntax
boolean refresh();

Arguments
none

Returned Value
boolean

Example
```
alert(parent.tocFrame.IMSToc.refresh());
```

IMSToc.setBGColor

Description
Sets the background color of the table of contents.

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
boolean

See Also
IMSToc.getBGColor

Example
```
alert(“set the background color of the IMSToc panel to white”);
alert(parent.tocFrame.IMSToc.setBGColor(255,255,255));
```

IMSToc.setDrawBorderEnabled

Description
Sets whether the table of contents' 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
boolean

See Also
IMSToc.isDrawBorderEnabled

Example
```
alert("force the TOC border to be drawn");
var IMSToc = parent.mapFrame.IMSMap.getToc();
alert(IMSToc.setDrawBorderEnabled("true"));
```

IMSToc.setFGColor

Description

Sets a text color for the table of contents.

Syntax
boolean setFGColor(int R, int G, int B)

Arguments

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

Returned Value
boolean

See Also
IMSToc.getFGColor

Example
```
alert("set the red color of a layer legend text on the TOC");
 if (parent.tocFrame.IMSToc.setFGColor(255,0,0))
   alert("the current color of a layer legend text is" +
parent.tocFrame.IMSToc.getFGColor());
 else alert("Error occurred while setting color");
```

IMSToc.setLayerExpanded

Description
Sets whether a layer classification legend is displayed. It is displayed by default.

Syntax
boolean setLayerExpanded(Layer layer, String expanded)

Arguments
layer	The name of the specified layer.
expanded	"True" if the layer classification legend is displayed, "false" if not.

Returned Value
boolean

Example
```
alert("force that the classification legend not to be displayed for the county
layer");
var IMSToc = parent.mapFrame.IMSMap.getToc();
var layer = parent.mapFrame.IMSMap.getLayer("County");
alert(IMSToc.setLayerExpanded(layer, "false"));
```

IMSToc.setMap

Description
Associates the table of contents with a specified IMSMap.

Syntax
boolean setMap(IMSMap imsMap);

Arguments
imsMap	An IMSMap applet.

Returned Value
boolean

See Also
IMSMap

Example
```
alert(parent.tocFrame.IMSToc.setMap(parent.mapFrame.IMSMap));
```

IMSToc.setSelectedLayer

Description
Sets the selected (active) layer for the table of contents.

Syntax
boolean setSelectedLayer(Layer layer)

Arguments

layer	Name of the layer to be selected.

Returned Value
boolean

See Also
IMSToc.getSelectedLayer

Example
```
if (parent.tocFrame.IMSToc.setSelectedLayer(“County”))
   alert(“the County has been set up as the selected layer on the TOC panel”);
else (“Error occurred while setting up selected layer”);
```

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 table of contents, for instance if you manually set the Visible property of the layer and want the table of contents 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
boolean

See Also
Layer

Example
```
var layer = parent.mapFrame.IMSMap.getLayer(“County”);
alert( layer.setVisible(“false”));
alert(parent.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

Returns the value of the separator used to display multiple fields in a label. The separator can be any string character.

Syntax

String getSeparator()

Arguments

none

Returned Value

String	Returns the string character used as the separator.

See Also

IMSMap
LabelRenderer.setSeparator
Renderer

Example

```
var labelrenderer;
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); //Layer with LabelRenderer
labelrenderer = parent.mapFrame.IMSMap.getLayerLabelRenderer (myLayer);
var separatorChar = labelrenderer.getSeparator();
alert(“The separator used is:” + separatorChar);
```

LabelRenderer.setField

Description
Sets a field to be used for the label.

Syntax
boolean setField(Layer layer, String field)

Arguments

layer	The layer object to apply the field to.
field	The field name used to store values used to draw text on a feature.

Returned Value
boolean

See Also
IMSMap
Layer
Renderer

Example
```
var sym, labelrenderer;
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); // Layer with LabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
labelrenderer = parent.mapFrame.IMSMap.createRenderer(“LABEL_RENDERER”);
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
labelrenderer.setSymbol(sym);
labelrenderer.setField(myLayer, “field1”);
labelrenderer = parent.mapFrame.IMSMap.getLayerLabelRenderer(myLayer);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer,labelrenderer);
```

LabelRenderer.setFields

Description
Sets multiple fields to be used for the label. Uses the setSeparator method to identify the string used to separate each field.

Syntax
boolean setFields(Layer layer, Collection col)

Arguments

layer	The layer object to apply the fields on.
col	The collection object that contains the field names to use.

Returned Value
boolean

See Also
Collection
Layer
Renderer
ArcXML Programmer's Reference Guide

Example
```
var col;
var sym, labelrenderer;
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); // Layer with LabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
labelrenderer = parent.mapFrame.IMSMap.createRenderer(“LABEL_RENDERER”);
col = parent.mapFrame.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 = parent.mapFrame.IMSMap.getLayerLabelRenderer(myLayer);
```

LabelRenderer.setSeparator

Description

Sets 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 separator)

Arguments

separator	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

See Also

IMSMap
LabelRenderer.getSeparator
Renderer

Example

```
var col;
var sym, labelrenderer;
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); // Layer with
LabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
labelrenderer = parent.mapFrame.IMSMap.createRenderer(“LABEL_RENDERER”);
col = parent.mapFrame.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 = parent.mapFrame.IMSMap.getLayerLabelRenderer(myLayer);
```

LabelRenderer.setSymbol

Description
Sets the text symbol for the label.

Syntax
boolean setSymbol(Symbol sym)

Arguments

sym	A text symbol object that controls how text is rendered.

Returned Value
boolean

See Also
Renderer
Symbol
ArcXML Programmer's Reference Guide

Example
```
var sym, labelrenderer;
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); // Layer with LabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
labelrenderer = parent.mapFrame.IMSMap.createRenderer("LABEL_RENDERER");
sym.setFont("times", 12);
sym.setAntialiasing("true");
labelrenderer.setSymbol(sym);
labelrenderer.setField("myfieldname");
labelrenderer = parent.mapFrame.IMSMap.getLayerLabelRenderer(myLayer);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, labelrenderer );
```

Layer

Description
A layer is a visual representation of data obtained from a spatial dataset. It contains methods that allow for getting and setting display properties as well as methods that help in distinguishing the type of data source being used to produce this layer. Construction of layers can be done through IMSMap.createXXXLayer() methods.

See Also
IMSMap

Layer.clearScales

Description
Clears the scale factors set on the layer so the layer will draw at any scale.

Syntax
boolean clearScales()

Arguments
none

Returned Value
boolean

See Also
Layer.getMaxScale
Layer.getMinScale
Layer.setMaxScale
Layer.setMinScale

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
layer.clearScales();
```

Layer.clearSelection

Description
Clears all features that are currently selected.

Syntax
boolean clearSelection()

Arguments
none

Returned Value
boolean

See Also
Layer.setSelectable
Layer.setSelectionSymbol

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
layer.clearSelection();
```

Layer.getFieldNames

Description
Returns all the field names for the layer as a pipe (|) delimited string.

Syntax
String getFieldNames()

Arguments
none

Returned Value
String	Field names delimited by a pipe (|). If the layer is not a feature layer, then an empty string (“”) is returned.

See Also
Layer.getMapTipField
Layer.getQueryResultFields
Layer.setMapTipField
Layer.setQueryResultFields

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var names = layer.getFieldNames();
var nameArray = names.split(“|”);
```

Layer.getMapTipField

Description
Returns the name of field that is used for MapTips. MapTips are small popups that display data for a particular field.

Syntax
boolean getMapTipField()

Arguments
none

Returned Value
String	The name of the field being used for MapTips. If no field has been selected, then null is returned.

See Also
Layer.setMapTipField

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var mtField = layer.getMapTipField();
```

Layer.getMaxScale

Description
Returns the maximum scale at which the layer is displayed. If the 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
Layer.setMinScale

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var max = layer.getMaxScale();
```

Layer.getMinScale

Description
Returns the minimum scale factor at which the layer is displayed. If the 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.setMaxScale
Layer.setMinScale

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var min = layer.getMinScale();
```

Layer.getName

Description
Returns the name of the layer.

Syntax
String getName()

Arguments
none

Returned Value

String	The name of the layer. If the layer does not have a name or if one cannot be found, an empty String ("") is returned.

See Also
Layer.setName

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var name = layer.getName();
```

Layer.getQueryResultFields

Description
Returns a collection containing the names of all the fields that are displayed when an identify is performed. All fields are identifiable by default.

Syntax
Collection getQueryResultFields()

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.setQueryResultFields

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var qrFields = layer.getQueryResultFields();
```

Layer.getSelectionSet

Description
Returns the selection set of the 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 layer = parent.mapFrame.IMSMap.getSelectedLayer();
var selectionSet = layer.getSelectionSet();
```

Layer.hasGeocodeExtension

Description
Determines if the layer can be geocoded.

Syntax
boolean hasGeocodeExtension()

Arguments
none

Returned Value
boolean

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var isLayerGeocodable = layer.hasGeocodeExtension();
```

Layer.hasSelectionSet

Description
Checks whether given layer has a selection set or not.

Syntax
boolean hasSelectionSet()

Arguments
none

Returned Value
boolean

See Also
Layer.clearSelection

Layer.getSelectionSet

Layer.setSelectable

Layer.setSelectionSymbol

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if (layer.hasSelectionSet())
   alert(“the layer has a selection set”);
else alert(“the layer doesn’t have a selection set”);
```

Layer.isAVIMSGroupLayer

Description
Checks if the layer is an ArcView IMS group layer.

Syntax
boolean isAVIMSGroupLayer()

Arguments
none

Returned Value
boolean

See Also
Layer.isAVIMSSubLayer

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isAVIMSGroupLayer() || layer.isMOIMSGroupLayer() ||
layer.isImageServerLayer()){
  alert(“Current layer is a group layer.”);
}
```

Layer.isAVIMSSubLayer

Description
Checks if the layer is an ArcView IMS sublayer.

Syntax
boolean isAVIMSSubLayer()

Arguments
none

Returned Value
boolean

See Also
Layer.isAVIMSGroupLayer

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isAVIMSSubLayer() || layer.isMOIMSSubLayer() ||
layer.isImageServerSubLayer()){
  alert(“Current layer is a sublayer.”);
}
```

Layer.isCapable

Description

Checks if a layer or sublayer has a given capability.

Syntax

boolean isCapable(String capability)

Arguments

capability	ATTRIBUTE_QUERY is for Identify, Find, QueryBuilder, MapTips, Attributes. GEOCODE is for Geocoding. STORED_QUERY is for Stored Query. SPATIAL_QUERY is for Select Features, Buffer.

Returned Value

boolean

Example

```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isCapable(“GEOCODE”){
   alert(“The current layer is capable of geocoding.”);
}
```

Layer.isFeatureLayer

Description

Checks if the layer is a feature layer. A feature layer is a layer from an ArcIMS Feature Service, a sublayer from an Image Service (ArcIMS, MapObjects IMS, and ArcView IMS) that has vectors, or a shapefile or ArcSDE layer from a local data source.

Syntax

boolean isFeatureLayer()

Arguments

none

Returned Value

boolean

Example

```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isFeatureLayer()){
   alert(“The layer is a feature layer.”);
}
```

Layer.isIdentifiable

Description
Checks if features on this layer can be identified.

Syntax
boolean isIdentifiable()

Arguments
none

Returned Value
boolean

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isIdentifiable()){
  alert(“The layer is identifiable.”);
}
```

Layer.isImageLayer

Description
Checks if the layer is an image layer.

Syntax
boolean isImageLayer()

Arguments
none

Returned Value
boolean

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isImageLayer){
  alert(“The layer is an image layer.”);
}
```

Layer.isImageServerLayer

Description
Checks if the layer is an Image Service (ArcIMS, MapObjects IMS, or ArcView IMS).

Syntax
boolean isImageServerLayer()

Arguments
none

Returned Value
boolean

See Also
Layer.isImageServerSubLayer

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isAVIMSGroupLayer() || layer.isMOIMSGroupLayer() ||
layer.isImageServerLayer()){
   alert(“The layer is a group layer.”);
}
```

Layer.isImageServerSubLayer

Description
Checks if the layer is an Image Service sublayer (ArcIMS, MapObjects IMS, or ArcView IMS).

Syntax
boolean isImageServerSubLayer()

Arguments
none

Returned Value
boolean

See Also
Layer.isImageServerLayer

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isAVIMSSubLayer() || layer.isMOIMSSubLayer() ||
layer.isImageServerSubLayer()){
   alert(“The layer is a sublayer.”);
}
```

Layer.isMOIMSGroupLayer

Description
Checks if the layer is a MapObjects IMS group layer.

Syntax
boolean isMOIMSGroupLayer()

Arguments
none

Returned Value
boolean

See Also
Layer.isMOIMSSubLayer

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isAVIMSGroupLayer() || layer.isMOIMSGroupLayer() ||
layer.isImageServerLayer()){
  alert(“The layer is a group layer.”);
}
```

Layer.isMOIMSSubLayer

Description
Checks if the layer is a MapObjects IMS sublayer.

Syntax
boolean isMOIMSSubLayer()

Arguments
none

Returned Value
boolean

See Also
Layer.isMOIMSGroupLayer

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isAVIMSSubLayer() || layer.isMOIMSSubLayer() ||
layer.isImageServerSubLayer()){
  alert(“The layer is a sublayer.”);
}
```

Layer.isSelectable

Description

Checks if the features on this layer can be selected.

Syntax

boolean isSelectable()

Arguments

none

Returned Value

boolean

See Also

Layer.clearSelection
Layer.setSelectable
Layer.setSelectableByInt
Layer.setSelectionSymbol

Example

```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
if(layer.isSelectable()){
   alert("The layer is selectable.");
}
```

Layer.isVisible

Description

Checks if the layer is visible. This method will only return true if the layer is contained in the table of contents and is currently checked as visible.

Syntax

boolean isVisible()

Arguments

none

Returned Value

boolean

See Also

Layer.setVisible

Example

```
function toggleVisibility{
var layer = parent.mapFrame.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 a 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 layer = parent.mapFrame.IMSMap.getSelectedLayer();
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 field 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 layer = parent.mapFrame.IMSMap.getSelectedLayer();
var results = layer.selectByCircle(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

See Also
Layer.isSelectable
Layer.setSelectable
Layer.setSelectableByInt

Example
```
Function toggleSelectability(){
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   if(layer.isSelectable()){
      layer.setFeaturesSelectable(0);
    }else{
      layer.setFeaturesSelectable(1);
    }
}
```

Layer.setIdentifiable

Description
Enables/Disables the identifiable property for this layer. If set to true, then features can be identified.

Syntax
boolean setIdentifiable(boolean allowIdentify)

Arguments

allowIdentify	Set to true to allow for the identification of features, false otherwise.

Returned Value
boolean

See Also
Layer.isIdentifiable
Layer.setIdentifiableByInt

Example
```
Function toggleIdentifiability(){
   var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   layer.setIdentifiable(!(layer.isIdentifiable()));
}
```

Layer.setIdentifiableByInt

Description

Enables/Disables 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

See Also

Layer.isIdentifiable
Layer.setIdentifiable

Example

```
Function toggleIdentifiability(){
   var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   if(layer.isIdentifiable()){
      layer.setIdentifiableByInt(0);
   }else{
      layer.setIdentifiableByInt(1);
   }
}
```

Layer.setMapTipField

Description
Sets 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 MapTips. If fieldName is not valid for this layer, this method has no effect.

Returned Value
boolean

See Also
Layer.getMapTipField

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
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

See Also
Layer.clearScales
Layer.setMinScale

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var currentScale = parent.mapFrame.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

See Also
Layer.clearScales
Layer.setMaxScale

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var currentScale = parent.mapFrame.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

See Also
Layer.getName

Example
```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
layer.setName(“Infected_sites”);
```

Layer.setQueryResultFields

Description

Sets the list of all the fields that are displayed when an identify is performed. All fields will be identifiable by default. The collection must either have String objects or NameValuePair objects (where the name is the field name 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 setQueryResultFields(Collection fields)

Arguments

fields	This is the collection of Strings or of NameValuePairs 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

See Also

Layer.getQueryResultFields

Example

```
var field1 = parent.mapFrame.IMSMap.createNameValuePair(“Names”, “Names of
Employers”);
var field2 = parent.mapFrame.IMSMap.createNameValuePair(“Age_Employee”, “Average
Age of Employees”);
var collection = parent.mapFrame.IMSMap.createCollection();
collection.addNameValuePairElement(field1);
collection.addNameValuePairElement(field2);
setQueryResultFields(collection);
```

Layer.setSelectable

Description

Allows or prohibits the 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

See Also

Layer.isSelectable
Layer.setFeaturesSelectable
Layer.setSelectableByInt

Example

```
Function toggleSelectability(){
   var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   layer.setSelectable(!(layer.isSelectable()));
}
```

Layer.setSelectableByInt

Description

Allows or prohibits the 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

See Also

Layer.isSelectable
Layer.setFeaturesSelectable
Layer.setSelectable

Example

```
Function toggleSelectability(){
   var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   if(layer.isSelectable()){
      layer.setSelectableByInt(0);
   }else{
      layer.setSelectableByInt(1);
   }
}
```

Layer.setSelectionSymbol

Description

Sets the symbol that 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

Example

```
var layer = parent.mapFrame.IMSMap.getSelectedLayer();
var newSymbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
newSymbol.setSize(24);
layer.setSelectionSymbol(newSymbol);
```

Layer.setVisible

Description
Sets the layer’s current visibility on the map.

Syntax
boolean setVisible(String visibility)

Arguments

visibility	“True” to make the layer visible on the map; “false” to hide the layer.

Returned Value
boolean

See Also
Layer.isVisible
Layer.setVisibleByInt

Example
```
function toggleVisibility{
   var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   if(layer.isVisible()){
      layer.setVisible(“false”);
   }else{
      layer.setVisible(“true”);
   }
}
```

Layer.setVisibleByInt

Description

Sets the layer's current 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 layer.

Returned Value

boolean

See Also

Layer.isVisible
Layer.setVisible

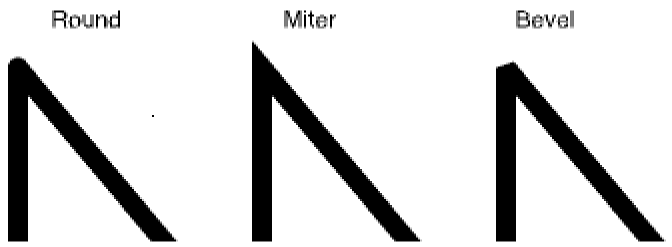
Example

```
function toggleVisibility{
   var layer = parent.mapFrame.IMSMap.getSelectedLayer();
   if(layer.isVisible()){
      layer.setVisibleByInt(0);
   }else{
      layer.setVisibleByInt(1);
}
}
```

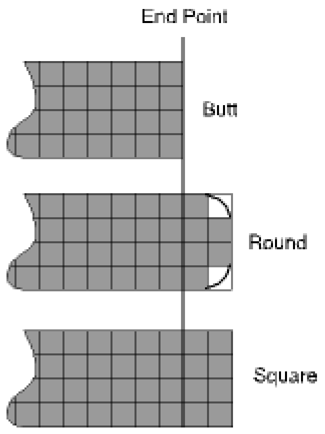
LineSymbol extends Symbol

This symbol draws line features. You can create an instance of the symbol through the IMSMap.createSymbol method. The styles of the line symbol are solid, dash, dot, dash–dot, and dash–dot–dot, and it provides special effects such as antialiasing, transparency, and overlap. The style property is handled by the set/getStyle methods. There are also two properties—captype and jointype—that provide different ways of line capping and line joining as shown below:

jointypes:



captypes:



See Also

IMSMap
PolygonSymbol
Symbol
ArcXML Programmer's Reference Guide

LineSymbol.getAntialiasing

Description
Returns the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax
boolean getAntialiasing()

Arguments
none

Returned Value
boolean

See Also
LineSymbol.setAntialiasing

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
alert(“default value of antialiasing property for LineSymbol is” +
symbol.getAntialiasing());
```

LineSymbol.getCapType

Description
Returns 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 = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
alert(“default cap type for the LineSymbol is” +  symbol.getCapType());
```

LineSymbol.getColor

Description
Returns the 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 = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
alert(“the default color of the LineSymbol is” +  symbol.getColor());
```

LineSymbol.getJoinType

Description
Returns the 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 = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
alert(“the default join type of the LineSymbol is” + symbol.getJoinType());
```

LineSymbol.getStyle

Description
Returns the 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
```
var symbol = parent.mapFrame.IMSMap.createSymbol (“LINE_SYMBOL”);
alert (“the default style of the LineSymbol is” + symbol.getStyle());
```

LineSymbol.getTransparency

Description
Returns the transparency value set on the object.

Syntax
double getTransparency( )

Arguments
none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
LineSymbol.setTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
alert(“the default transparency value of the LineSymbol is” +
symbol.getTransparency());
```

LineSymbol.getWidth

Description
Returns the width of the line symbol. The default value is 1.

Syntax
int getWidth( )

Arguments
none

Returned Value

int	The width of the line symbol in screen pixels.

See Also
LineSymbol.setWidth

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
alert(“the default  width of the LineSymbol is” +  symbol.getWidth());
```

LineSymbol.setAntialiasing

Description
Sets the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax
boolean setAntialiasing(String enabled)

Arguments

enabled	“True” to set antialiasing on. “False” to set antialiasing off.

Returned Value
boolean

See Also
LineSymbol.getAntialiasing

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
symbol.setAntialiasing(“true”)
```

LineSymbol.setCapType

Description
Sets the 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
boolean

See Also
LineSymbol.getCapType

Example
See the LineSymbol.setJoinType section and comments there.

LineSymbol.setColor

Description
Sets the 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
boolean

See Also
Color
LineSymbol.getColor

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
var color1 = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setColor(color1)
```

LineSymbol.setJoinType

Description
Sets the 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
boolean

See Also
LineSymbol.getJoinType
Renderer
Symbol

Example
```
var layer = parent.mapFrame.IMSMap.getLayer(“Highways”);
var rend = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
var symbol = parentmapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
symbol.setJoinType(2));
symbol.setCapType(2));
symbol.setStyle(2));
symbol.setWidth(5));
alert(“style is” + symbol.getStyle());
alert(rend.setSymbol(symbol));
parent.mapFrame.IMSMap.setLayerRenderer(layer, rend);
parent.mapFrame.IMSMap.redraw();

Note: setCapType(), setStyle(), setWidth() and getStyle() methods will work
correctly only after setJoinType method has been called
```

LineSymbol.setStyle

Description
Sets the 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
boolean

See Also
LineSymbol.getStyle

Example
See LineSymbol.setJoinType.

LineSymbol.setTransparency

Description
Sets the transparency value of the symbol.

Syntax
boolean setTransparency(double transparencyValue)

Arguments

transparencyValue	Defines the transparency to be set. The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value
boolean

See Also
LineSymbol.getTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
symbol.setTransparency(0.5);
```

LineSymbol.setWidth

Description
Sets the width of the line symbol.

Syntax
boolean setWidth(int width)

Arguments

width	The width in screen pixels. The default value is 1.

Returned Value
boolean

See Also
LineSymbol.getWidth

Example
See LineSymbol.setJoinType.

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 the setStyle method. The MarkerSymbol provides special effects such as antialiasing, overlap, outline, and shadows. You can create an instance of the class by calling IMSMap.createSymbol method.

See Also

IMSMap
Symbol
ArcXML Programmer's Reference Guide

MarkerSymbol.getAntialiasing

Description

Returns the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean getAntialiasing()

Arguments

none

Returned Value

boolean

See Also

MarkerSymbol.setAntialiasing

Example

```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
alert(“default value of antialiasing for a MarkerSymbol is” +
symbol.getAntialiasing());
```

MarkerSymbol.getColor

Description
Returns the color of the marker 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
MarkerSymbol.setColor

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
alert(“default color of a MarkerSymbol is” + symbol.getColor());
```

MarkerSymbol.getOutlineColor

Description
Returns the outline color of the marker symbol. The default outline color is undefined. A call to setOutlineColor must be made before using the 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 = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
var color = parent.mapFrame.IMSMap.createColor(0,255,0);
alert(“set the outline color to green” + symbol.setOutlineColor(color));
alert(“the outline color for the MarkerSymbol is” +
symbol.getOutLineColor());
```

MarkerSymbol.getShadowColor

Description
Returns the shadow color of the marker symbol. The default shadow color is undefined. A call to setShadowColor must be made before using the 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 = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
var color = parent.mapFrame.IMSMap.createColor(0,255,0);
alert(“set the outline color to green” + symbol.setShadowColor(color));
alert(“the shadow color for the MarkerSymbol is” +
symbol.getShadowColor());
```

MarkerSymbol.getSize

Description
Returns the size of the marker symbol. The default size is 3.

Syntax
int getSize( )

Arguments
none

Returned Value
int	Returns the size value in screen pixels.

See Also
MarkerSymbol.setSize

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
alert(“default size for a MarkerSymbol is” + symbol.getSize());
```

MarkerSymbol.getStyle

Description
Returns the style of the marker symbol. 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 = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
alert(“the default style for a MarkerSymbol is” +  symbol.getStyle());
```

MarkerSymbol.getTransparency

Description
Returns the transparency value of the symbol.

Syntax
double getTransparency( )

Arguments
none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
MarkerSymbol.setTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
alert(“default value of the transparency property is” + symbol.getTransparency());
```

MarkerSymbol.setAntialiasing

Description
Sets the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax
boolean setAntialiasing(String enabled)

Arguments

enabled	"True" to set antialiasing on. "False" to set antialiasing off.

Returned Value
boolean

See Also
MarkerSymbol.getAntialiasing

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol("MARKER_SYMBOL");
symbol.setAntialiasing("true");
```

MarkerSymbol.setColor

Description
Sets the color of the marker symbol. The default color is black.

Syntax
boolean setColor(Color color)

Arguments

color	An instance of the color class.

Returned Value
boolean

See Also
Color
MarkerSymbol.getColor

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

MarkerSymbol.setOutlineColor

Description
Sets the 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.

Returned Value
boolean

See Also
Color
MarkerSymbol.getOutlineColor

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
var color1 = parent.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.

Returned Value
boolean

See Also
Color
MarkerSymbol.getShadowColor

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
var color1 = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setShadowColor(color1);
```

MarkerSymbol.setSize

Description
Sets the size of the marker symbol. The default size is 3.

Syntax
boolean setSize(int size)

Arguments

size	The size value in screen pixels.

Returned Value
boolean

See Also
MarkerSymbol.getSize

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
symbol.setSize(10);
```

MarkerSymbol.setStyle

Description
Sets the style of the marker symbol. The default style is circle.

Syntax
boolean setStyle(int style)

Arguments

style:	0—circle
	1—square
	2—triangle
	3—cross
	4—star

Returned Value
boolean

See Also
MarkerSymbol.getStyle

Example
See the next page.

MarkerSymbol.setStyle

Example

```
var layer = parent.mapFrame.IMSMap.getLayer(“Agencies”);
var rend = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
var symbol = parent.mapFrame.IMSMap.createSymbol(“MARKER_SYMBOL”);
symbol.setSize(10);
symbol.setStyle(4);
var color1 = parent.mapFrame.IMSMap.createColor(0,255,0);
var color2 = parent.mapFrame.IMSMap.createColor(0,0,255);
var color3 = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setColor(color1);
symbol.setShadowColor(color2);
symbol.setOutlineColor(color3);

alert(“the size is” + symbol.getSize());
alert(“the style is” + symbol.getStyle());
alert(“the color is” + symbol.getColor());
alert(“the shadow color is” + symbol.getShadowColor());
alert(“the outline color is” + symbol.getOutlineColor());
rend.setSymbol(symbol);
parent.mapFrame.IMSMap.setLayerRenderer(layer, rend);
parent.mapFrame.IMSMap.redraw();
```

MarkerSymbol.setTransparency

Description

Sets the transparency value of the symbol.

Syntax

boolean setTransparency(double transparencyValue)

Arguments

transparencyValue	Defines transparency to be set. The default value is 1.0. The valid value range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value

boolean

See Also

MarkerSymbol.getTransparency

Example

```
var symbol = parent.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 IMSMap.createNameValuePair().

See Also

Collection
IMSMap
Layer.setQueryResultFields

NameValuePair.getName

Description

Returns 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 nameValuePair = parent.mapFrame.IMSMap.createNameValuePair (“France”, “World
Champion”);
var name = nameValuePair.getName();
```

NameValuePair.getValue

Description
Returns 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 nameValuePair = parent.mapFrame.IMSMap.createNameValuePair(“France”, “World
Champion”);
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

See Also
NameValuePair.getName
NameValuePair.setValue

Example
```
var nameValuePair = parent.mapFrame.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

See Also
NameValuePair.getValue
NameValuePair.setName

Example
```
var nameValuePair = parent.mapFrame.IMSMap.createNameValuePair(“Jack”, ”the boss”);
nameValuePair.setValue(“the president”);
```

PolygonSymbol extends Symbol

This class contains methods to draw a polygon feature. 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. You can create an instance of the polygon symbol through the IMSMap.createSymbol method.

See Also

IMSMap
LineSymbol
Symbol
ArcXML Programmer's Reference Guide

PolygonSymbol.getAntialiasing

Description

Returns the antialiasing value of the symbol boundary. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean getAntialiasing()

Arguments

none

Returned Value

boolean

See Also

PolygonSymbol.setAntialiasing

Example

```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“default value of antialiasing property for the polygon symbol is” +
symbol.getAntialiasing());
```

PolygonSymbol.getBoundaryColor

Description
Returns the 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 = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“default value of the boundary color for the polygon symbol is” +
symbol.getBoundaryColor());
```

PolygonSymbol.getCapStyle

Description
Returns the cap type of the polygon symbol boundary. The default cap type is round.

Syntax
int getCapStyle( )

Arguments
none

Returned Value

int:	0—butt
	1—round
	2—square

See Also
PolygonSymbol.setCapStyle

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“the default boundary cap style for the polygon symbol is” +
symbol.getCapStyle());
```

PolygonSymbol.getFillColor

Description

Returns the fill color of the polygon symbol. This method can only be used after an initial 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 = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
var color1 = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setFillColor(color1));
alert(“the fill color of the polygon symbol is” + symbol.getFillColor());
```

PolygonSymbol.getFillStyle

Description
Returns the fill style of the polygon symbol.

Syntax
int getFillStyle( )

Arguments
none

Returned Value

0—solid fill
1—transparent fill
2—horizontal fill
3—vertical fill
4—upward diagonal fill
5—downward diagonal fill
6—cross fill
7—diagonal cross fill
8—light gray fill
9—gray fill
10—dark gray fill

See Also

PolygonSymbol.setFillStyle
ArcXML Programmer's Reference Guide

Example

```
var layer = parent.mapFrame.IMSMap.getLayer('states');
var rend = parent.mapFrame.IMSMap.createRenderer("SIMPLE_RENDERER");
var symbol = parent.mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
var color1 = parent.mapFrame.IMSMap.createColor(0,255,255);
symbol.setFillColor(color1);
symbol.setStyle(2);
symbol.setFillStyle(8);
alert("the fill style of the polygon symbol is" + symbol.getFillStyle());
```

PolygonSymbol.getFillTransparency

Description
Returns the fill transparency value of the polygon symbol.

Syntax
double getFillTransparency( )

Arguments
none

Returned Value
double	Returns fill transparency. The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
PolygonSymbol.setTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“the default fill transparency value for the polygon symbol is” +
symbol.getFillTransparency());
```

PolygonSymbol.getJoinStyle

Description
Returns the boundary join type of the polygon symbol. The default value is round.

Syntax
int getJoinStyle( )

Arguments
none

Returned Value
int:	0—miter 1—round 2—bevel

See Also
LineSymbol.getJoinType
PolygonSymbol.setJoinStyle

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“the default boundary join type of the PolygonSymbol is” +
symbol.getJoinStyle());
```

PolygonSymbol.getStyle

Description
Returns the boundary style of the polygon 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.getStyle
PolygonSymbol.setStyle

Example
See PolygonSymbol.setJoinStyle.

PolygonSymbol.getTransparency

Description
Returns the boundary transparency value of the polygon symbol.

Syntax
double getTransparency( )

Arguments
none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
PolygonSymbol.setTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“the default boundary transparency value of the PolygonSymbol is” +
symbol.getTransparency());
```

PolygonSymbol.getWidth

Description
Returns the boundary width of the polygon symbol. The default value is 1.

Syntax
float getWidth( )

Arguments
none

Returned Value

float	The boundary width of the polygon symbol in screen pixels.

See Also
PolygonSymbol.setWidth

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
alert(“the default boundary width value of the PolygonSymbol is” +
symbol.getWidth());
```

PolygonSymbol.setAntialiasing

Description
Sets the antialiasing value of the polygon symbol boundary. Antialiasing is the process of adding pixels along the diagonal lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax
boolean setAntialiasing(String enabled)

Arguments

enabled	“True” to set antialiasing on. “False” to set antialiasing off.

Returned Value
boolean

See Also
PolygonSymbol.getAntialiasing

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
symbol.setAntialiasing(“true”)
```

PolygonSymbol.setBoundaryColor

Description
Sets the 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.

Returned Value
boolean

See Also
Color
PolygonSymbol.getBoundaryColor

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

PolygonSymbol.setCapStyle

Description
Sets the cap type of the polygon symbol boundary. The default cap type is round.

Syntax
boolean setCapStyle(int capStyle)

Arguments

capStyle:	0—butt
	1—round
	2—square

Returned Value
boolean

See Also
PolygonSymbol.getCapStyle

Example
See PolygonSymbol.setJoinStyle.

PolygonSymbol.setFillColor

Description
Sets the fill color of the polygon symbol. This method can only be used after an initial call of the setFillColor method.

Syntax
boolean setFillColor(Color color)

Arguments
color	An instance of the color class.

Returned Value
boolean

See Also
Color
PolygonSymbol.getFillColor

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
var color1 = parent.mapFrame.IMSMap.createColor(255,0,0);
symbol.setFillColor(color1);
```

PolygonSymbol.setFillStyle

Description
Sets the fill style for a polygon symbol.

Syntax
boolean setFillStyle(int fstyle)

Arguments

fstyle:	0—solid fill
	1—transparent fill
	2—horizontal fill
	3—vertical fill
	4—upward diagonal fill
	5—downward diagonal fill
	6—cross fill
	7—diagonal cross fill
	8—light gray fill
	9—gray fill
	10—dark gray fill

Returned Value
boolean

See Also
PolygonSymbol.getFillStyle
ArcXML Programmer's Reference Guide

Example
```
var layer = parent.mapFrame.IMSMap.getLayer(‘states’);
var rend = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
var color1 = parent.mapFrame.IMSMap.createColor(0,255,255);
symbol.setFillColor(color1);
symbol.setStyle(2);
symbol.setFillStyle(8);
alert(“FillStyle is” + symbol.getFillStyle());
rend.setSymbol(symbol);
parent.mapFrame.IMSMap.setLayerRenderer(layer,rend);
parent.mapFrame.IMSMap.redraw();
```

PolygonSymbol.setFillTransparency

Description
Sets the 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).

Syntax
boolean setFillTransparency(double transparencyValue)

Arguments

transparencyValue	Defines transparency value to be set.

Returned Value
boolean

See Also
PolygonSymbol.getTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
symbol.setFillTransparency(0.5);
```

PolygonSymbol.setJoinStyle

Description
Sets the boundary join type of the polygon symbol. The default value is round.

Syntax
boolean setJoinStyle(int joinStyle)

Arguments

joinStyle:	0—miter
	1—round
	2—bevel

Returned Value
boolean

See Also
LineSymbol.setJoinType
PolygonSymbol.getJoinStyle

Example
See the next page.

PolygonSymbol.setJoinStyle

```
var layer = parent.mapFrame.IMSMap.getLayer(“County”);
var rend = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
var symbol = parent.mapFrame.IMSMap.createSymbol(“POLYGON_SYMBOL”);
symbol.setJoinStyle(2);
symbol.setCapStyle(2);
symbol.setStyle(2);
symbol.setWidth(5.0);
alert(“style is” + symbol.getStyle());
alert(rend.setSymbol(symbol));
parent.mapFrame.IMSMap.setLayerRenderer(layer,rend);
parent.mapFrame.IMSMap.redraw();
```

PolygonSymbol.setStyle

Description
Sets the boundary style of the polygon 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
boolean

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).

Syntax
boolean setTransparency(double transparencyValue)

Arguments
transparencyValue	Defines transparency value to be set.

Returned Value
boolean

See Also
PolygonSymbol.getTransparency

Example
```
var symbol = parent.mapFrame.IMSMap.createSymbol("POLYGON_SYMBOL");
symbol.setTransparency(0.5);
```

PolygonSymbol.setWidth

Description
Sets the boundary width of the polygon symbol. The default value is 1.

Syntax
boolean setWidth(float width)

Arguments
width	The width is in screen pixels.

Returned Value
boolean

See Also
LineSymbol.setWidth
PolygonSymbol.getWidth

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 display properties for this symbol. The method IMSMap.createSymbol() is used to create an object of this type.

See Also

IMSMap
RasterMarkerSymbol
RasterShieldSymbol
ShieldSymbol

RasterFillSymbol.getAntialiasing

Description

Returns the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the diagonal lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean getAntialiasing()

Arguments

none

Returned Value

boolean

See Also

RasterFillSymbol.setAntialiasing

Example

```
var rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
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 rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
var imagePath = rasterFillSymbol.getImagePathString();
```

RasterFillSymbol.getTransparency

Description
Returns the transparency value of the symbol.

Syntax
double getTransparency()

Arguments
none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
RasterFillSymbol.setTransparency

Example
```
var rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
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:/site/image/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 rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
var url = rasterFillSymbol.getURLString();
```

RasterFillSymbol.setAntialiasing

Description

Sets the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean setAntialiasing(String value)

Arguments

value	Use “true” if antialiasing should be used, “false” otherwise.

Returned Value

boolean

See Also

RasterFillSymbol.getAntialiasing

Example

```
var rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
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

See Also

RasterFillSymbol.getImagePathString

Example

```
var rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
rasterFillSymbol.setImagePathString(“C:\\ArcIMS\\images\\fillImages\\sites.jpg”);
```

RasterFillSymbol.setTransparency

Description

Sets the transparency value of the symbol.

Syntax

boolean setTransparency(double transparency)

Arguments

transparency	The default value is 1.0 with a valid range of 0.0 (transparent) to 1.0 (opaque).

Returned Value

boolean

See Also

RasterFillSymbol.getTransparency

Example

```
var rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
rasterFillSymbol.setTransparency(.5);
```

RasterFillSymbol.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

See Also

RasterFillSymbol.getURLString

Example

```
var rasterFillSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_FILL_SYMBOL”);
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 display properties for this symbol. The method IMSMap.createSymbol() is used to create an object of this type.

See Also

IMSMap
RasterFillSymbol
RasterShieldSymbol
ShieldSymbol

RasterMarkerSymbol.getAntialiasing

Description

Returns the antialiasing value of the symbol. Antialiasing is the process of adding pixels along the lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean getAntialiasing()

Arguments

none

Returned Value

boolean

See Also

RasterMarkerSymbol.setAntialiasing

Example

```
var rasterMarkerSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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 rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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 rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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.getURLString
RasterMarkerSymbol.setImagePathString

Example

```
var rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
var imagePath = rasterMarkerSymbol.getImagePathString();
```

RasterMarkerSymbol.getShadowColor

Description

Returns the 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 rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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 rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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 rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
var height = rasterMarkerSymbol.getSizeY();
```

RasterMarkerSymbol.getTransparency

Description

Returns the transparency value of the symbol.

Syntax

double getTransparency()

Arguments

none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also

RasterMarkerSymbol.setTransparency

Example

```
var rasterMarkerSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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 rasterMarkerSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
var url = rasterMarkerSymbol.getURLString();
```

RasterMarkerSymbol.setAntialiasing

Description
Sets 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

See Also
RasterMarkerSymbol.getAntialiasing

Example
```
var rasterMarkerSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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

See Also
RasterMarkerSymbol.getHotSpotX
RasterMarkerSymbol.getHotSpotY
RasterMarkerSymbol.setHotSpotY

Example
```
var rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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

See Also
RasterMarkerSymbol.getHotSpotX
RasterMarkerSymbol.getHotSpotY
RasterMarkerSymbol.setHotSpotX

Example
```
var rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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

See Also
RasterMarkerSymbol.getImagePathString

Example
```
var rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
rasterMarkerSymbol.setImagePathString(“C:\\ArcIMS\\images\\fillImages\\sites.jpg”);
```

RasterMarkerSymbol.setShadowColor

Description
Sets the 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

See Also
RasterMarkerSymbol.getShadowColor

Example
```
var rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
var shadowColor = parent.mapFrame.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

See Also
RasterMarkerSymbol.getSizeX

Example
```
var rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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

See Also
RasterMarkerSymbol.getSizeY

Example
```
var rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
rasterMarkerSymbol.setSizeY(10);
```

RasterMarkerSymbol.setTransparency

Description
Sets the transparency of the symbol.

Syntax
boolean setTransparency(double transparency)

Arguments

transparency	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value
boolean

See Also
RasterMarkerSymbol.getTransparency

Example
```
var rasterMarkerSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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

See Also

RasterMarkerSymbol.getURLString

Example

```
var rasterMarkerSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_MARKER_SYMBOL”);
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 display properties for this symbol. The method IMSMap.createSymbol() is used to create an object of this type.

See Also

IMSMap
RasterFillSymbol
RasterMarkerSymbol
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

See Also

RasterShieldSymbol.setAntialiasing

Example

```
var rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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

See Also

RasterShieldSymbol.setBoundary

Example

```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var isBoundaryDrawn = rasterShieldSymbol.getBoundary();
```

RasterShieldSymbol.getFontColor

Description

Returns 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

RasterShieldSymbol.setFontColor

Example

```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var fontColor = rasterShieldSymbol.getFontColor();
```

RasterShieldSymbol.getFontName

Description
Returns the name of the 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 rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var fontName = rasterShieldSymbol.getFontName();
```

RasterShieldSymbol.getFontSize

Description
Returns the size of the 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 rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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 rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var imagePath = rasterShieldSymbol.getImagePathString();
```

RasterShieldSymbol.getLabelMode

Description

Returns the value of the label mode property.

Syntax

int getLabelMode()

Arguments

none

Returned Value

int	The value associated with the mode (see RasterShieldSymbol.setLabelMode for a list of known values).

See Also

RasterShieldSymbol.setLabelMode

Example

```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var labelMode = rasterShieldSymbol.getLabelMode();
```

RasterShieldSymbol.getPrintMode

Description
Returns the value of the print mode property.

Syntax
int getPrintMode()

Arguments
none

Returned Value

int	The value associated with the mode (see RasterShieldSymbol.setPrintMode for a list of known values).

See Also
RasterShieldSymbol.setPrintMode

Example
```
var rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var printMode = rasterShieldSymbol.getPrintMode();
```

RasterShieldSymbol.getShadowColor

Description
Returns the 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 rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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 rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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 rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var textYPos = rasterShieldSymbol.getTextPositionY();
```

RasterShieldSymbol.getTransparency

Description
Returns the transparency of the symbol.

Syntax
double getTransparency()

Arguments
none

Returned Value
double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
RasterShieldSymbol.setTransparency

Example
```
var rasterShieldSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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 rasterShieldSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var url = rasterShieldSymbol.getURLString();
```

RasterShieldSymbol.setAntialiasing

Description
Sets 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

See Also
RasterShieldSymbol.getAntialiasing

Example
```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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

See Also
RasterShieldSymbol.getBoundary

Example
```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setBoundary(“true”);
```

RasterShieldSymbol.setFont

Description

Sets the 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

See Also

RasterShieldSymbol.getFontName
RasterShieldSymbol.getFontSize

Example

```
var rasterShieldSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setFont(“Dialog”, 10);
```

RasterShieldSymbol.setFontColor

Description

Sets the font color.

Syntax

boolean setFontColor(Color color)

Arguments

color	The new color of the font.

Returned Value

boolean

See Also

Color
RasterShieldSymbol.getFontColor

Example

```
var rasterShieldSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var fontColor = parent.mapFrame.IMSMap.createColor(24, 32, 68);
rasterShieldSymbol.setFontColor(Color);
```

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”) 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

See Also
RasterShieldSymbol.getImagePathString

Example
```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setImagePathString(“C:\\ArcIMS\\images\\fillImages\\sitesz.jpg”);
```

RasterShieldSymbol.setLabelMode

Description
Sets the label mode for the symbol. Label mode affects the content of values displayed. You can choose to show the full value as it is found in the feature (full label), or you can choose to have just the number portion taken from the value (numeric only). The default is for numeric only values to be displayed.

Syntax
boolean setLabelMode(int mode)

Arguments

mode	The mode to print the labels. 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

See Also
RasterShieldSymbol.getLabelMode

Example
```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setLabelMode(1);
```

RasterShieldSymbol.setPrintMode

Description

Sets the print mode for the symbol. Print mode affects the display of labels. You can choose to display the labels as default, all uppercase, all lowercase, and pretty (which displays the first letter of each word in uppercase and the rest in lowercase).

Syntax

boolean setPrintMode(int mode)

Arguments

mode	The mode in 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 uppercase, the rest are lowercase) 2—all uppercase 3—all lowercase

Returned Value

boolean

See Also

RasterShieldSymbol.getPrintMode

Example

```
var rasterShieldSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setPrintMode(3);
```

RasterShieldSymbol.setShadowColor

Description

Sets the color of the symbol’s shadow.

Syntax

boolean setShadowColor(Color newColor)

Arguments

newColor	The new shadow color.

Returned Value

boolean

See Also

RasterShieldSymbol.getShadowColor

Example

```
var rasterShieldSymbol =
  parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
var shadowColor = parent.mapFrame.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

See Also
RasterShieldSymbol.getTextPositionX

Example
```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
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

See Also
RasterShieldSymbol.getTextPositionY

Example
```
var rasterShieldSymbol = parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setTextPositionY(3);
```

RasterShieldSymbol.setTransparency

Description
Sets the transparency of the symbol.

Syntax
boolean setTransparency(double transparency)

Arguments

transparency	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value

double	The transparency value of the symbol.

See Also
RasterShieldSymbol.getTransparency

Example
```
var rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setTransparency(.5);
```

RasterShieldSymbol.setURLString

Description
Sets the URL for the symbol’s image. The URL path is used by the ArcIMS spatial server to define the location of an image (e.g., “http://theworldsbestmapsite/images/symbol.jpg”).

Syntax
boolean setURLString(String url)

Arguments

url	The URL of the image.

Returned Value
boolean

See Also
RasterShieldSymbol.getURLString

Example
```
var rasterShieldSymbol =
   parent.mapFrame.IMSMap.createSymbol(“RASTER_SHIELD_SYMBOL”);
rasterShieldSymbol.setURLString(“http://www.esri.com/software/arcims/graphics/
sudameri.gif”);
```

Renderer

The Renderer class is used within various IMSMap methods that can be appropriately applied to all types of renderers. 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 subclassed 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

GroupRenderer
LabelRenderer
ScaleDependentRenderer
SimpleRenderer
ValueMapLabelRenderer
ValueMapRenderer

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 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 scale value is greater than the maximum scale, the renderer will not be shown.

Syntax

long getMaximumScale()

Arguments

none

Returned Value

long

See Also

ScaleDependentRenderer.setMaximumScale
ArcXML Programmer's Reference Guide

Example

```
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); //Layer with
ScaleDependentRenderer
var srend = parent.mapFrame.IMSMap.getLayerRenderer(myLayer);
var maxscale = srend.getMaximumScale();
```

ScaleDependentRenderer.getMinimumScale

Description
Returns the minimum scale at which this renderer is displayed. If the scale value is less than the minimum scale, the renderer will not be shown.

Syntax
long getMinimumScale()

Arguments
none

Returned Value
long

See Also
ScaleDependentRenderer.setMinimumScale
ArcXML Programmer's Reference Guide

Example
```
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); //Layer with
ScaleDependentRenderer
var srend = parent.mapFrame.IMSMap.getLayerRenderer(myLayer);
var minscale = srend.getMinimumScale();
```

ScaleDependentRenderer.inRange

Description
Determines 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
value	Any valid number to be used in determining if it is within the scale range specified.

Returned Value
boolean

See Also
ArcXML Programmer's Reference Guide

Example
```
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer(); //Layer with
ScaleDependentRenderer
var srend = parent.mapFrame.IMSMap.getLayerRenderer(myLayer);
if (srend.inRange(“1,500,000”) ) {
   alert(“Is in range”);
}
```

ScaleDependentRenderer.setMaximumScale

Description

Sets the maximum scale range to display the features using a Renderer.

Syntax

boolean setMaximumScale(long value)

Arguments

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

See Also

ScaleDependentRenderer.getMaximumScale
ArcXML Programmer's Reference Guide

Example

```
var srend = parent.mapFrame.IMSMap.createRenderer(“SCALE_DEPENDENT_RENDERER”);
var simprend = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
srend.setRenderer(simprend);
srend.setMaximumScale(“2,000,000”);
```

ScaleDependentRenderer.setMinimumScale

Description

Sets the minimum scale range to display the features using a Renderer.

Syntax

boolean setMinimumScale(long value)

Arguments

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

See Also

ScaleDependentRenderer.getMinimumScale
ArcXML Programmer's Reference Guide

Example

```
var srend = parent.mapFrame.IMSMap.createRenderer(“SCALE_DEPENDENT_RENDERER”);
var simprend = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
srend.setRenderer(simprend);
srend.setMinimumScale(“1,000,000”);
```

ScaleDependentRenderer.setRenderer

Description
Sets a renderer for a ScaleDependentRenderer.

Syntax
boolean setRenderer(Renderer rend)

Arguments

rend	A valid object renderer (i.e., SimpleRenderer, ValueMapRenderer, LabelRenderer).

Returned Value
boolean

See Also
IMSMap

Example
```
var srend = parent.mapFrame.IMSMap.createRenderer(“SCALE_DEPENDENT_RENDERER”);
var simprend = parent.mapFrame.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 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.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

See Also

ShieldSymbol.setAntialiasing

Example

```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var isAA = shieldSymbol.getAntialiasing();
```

ShieldSymbol.getFontColor

Description
Returns the 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 shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var fontColor = shieldSymbol.getFontColor();
```

ShieldSymbol.getFontName

Description
Returns the name of the 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 shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var fontName = shieldSymbol.getFontName();
```

ShieldSymbol.getFontSize

Description
Returns the size of the 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 shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var fontSize = shieldSymbol.getFontSize();
```

ShieldSymbol.getLabelMode

Description
Returns the value of the label mode property.

Syntax
int getLabelMode()

Arguments
none

Returned Value

int	The value associated with the mode (see ShieldSymbol.setLabelMode for a list of known values).

See Also
ShieldSymbol.setLabelMode

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var labelMode = shieldSymbol.getLabelMode();
```

ShieldSymbol.getMinSize

Description
Returns the minimum size for this symbol. Default value is 0.

Syntax
int getMinSize()

Arguments
none

Returned Value

int	The minimum size of the symbol.

See Also
ShieldSymbol.setMinSize

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var minShieldSize = shieldSymbol.getMinSize();
```

ShieldSymbol.getShadowColor

Description
Returns the 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 shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var shadowColor = shieldSymbol.getShadowColor();
```

ShieldSymbol.getTransparency

Description
Returns the transparency of the symbol.

Syntax
double getTransparency()

Arguments
none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
ShieldSymbol.setTransparency

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var transparency = shieldSymbol.getTransparency();
```

ShieldSymbol.getType

Description
Returns an integer corresponding to the type of shield being used 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 shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var shieldType = shieldSymbol.getType();
```

ShieldSymbol.setAntialiasing

Description
Sets 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

See Also
ShieldSymbol.getAntialiasing

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
shieldSymbol.setAntialiasing(“true”);
```

ShieldSymbol.setFont

Description
Sets the 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 fontsize)

Arguments
fontname	The name of the font to use.
fontsize	The size of the font to use.

Returned Value
boolean

See Also
ShieldSymbol.getFontName
ShieldSymbol.getFontSize

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
shieldSymbol.setFont(“Dialog”, 10);
```

ShieldSymbol.setFontColor

Description
Sets the font color.

Syntax
boolean setFontColor(Color color)

Arguments
color	The new color of the font.

Returned Value
boolean

See Also
ShieldSymbol.getFontColor
Color

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var color = parent.mapFrame.IMSMap.createColor(24,32,68);
shieldSymbol.setFontColor(color);
```

ShieldSymbol.setLabelMode

Description
Sets how the labels are displayed. You can choose to show the full value as it is found in the feature (full label), or you can choose to have just the number portion taken from the value (numeric only). The default is for numeric only values to be displayed.

Syntax
boolean setLabelMode(int mode)

Arguments
mode	The mode that the labels are to be printed in. 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

See Also
ShieldSymbol.getLabelMode

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
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

See Also
ShieldSymbol.getMinSize

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
shieldSymbol.setMinSize(5);
```

ShieldSymbol.setShadowColor

Description
Sets the 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

See Also
ShieldSymbol.getShadowColor

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
var shadowColor = parent.mapFrame.IMSMap.createColor(170,11,251);
shieldSymbol.setShadowColor(shadowColor);
```

ShieldSymbol.setTransparency

Description
Sets the transparency of the symbol.

Syntax
boolean setTransparency(double transparency)

Arguments

transparency	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value
boolean

See Also
ShieldSymbol.getTransparency

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
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—U.S. Road 2—Rectangle 3—Oval 4—Mexican Highway

Returned Value
boolean

See Also
ShieldSymbol.getType

Example
```
var shieldSymbol = parent.mapFrame.IMSMap.createSymbol(“SHIELD_SYMBOL”);
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 simple.

SimpleRenderer.setSymbol

Description

Sets the symbol that you want to use to display the features of the layer. A symbol object consists of attributes that control how a feature is displayed.

Syntax

boolean setSymbol(Symbol sym)

Arguments

sym	The symbol to use in rendering a feature.

Returned Value

boolean

See Also

IMSMap
Renderer
Symbol

Example

The following example demonstrates creating cased lines using a GroupRenderer and SimpleRenderer on a line layer.

```
var sym1, sym2, simple1, simple2, group;
group = parent.mapFrame.IMSMap.createRenderer(“GROUP_RENDERER”);
sym1 = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
sym2 = parent.mapFrame.IMSMap.createSymbol(“LINE_SYMBOL”);
simple1 = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
simple2 = parent.mapFrame.IMSMap.createRenderer(“SIMPLE_RENDERER”);
sym1.setAntialiasing(“true”);
sym1.setWidth(3);
sym1.setColor(parent.mapFrame.IMSMap.createColor(255,0,0));
simple1.setSymbol(sym1);
sym2.setAntialiasing(“true”);
sym2.setWidth(1);
sym2.setColor(parent.mapFrame.IMSMap.createColor(255,255,0));
simple2.setSymbol(sym2);
group.addRenderer(simple1);
group.addRenderer(simple2);
parent.mapFrame.IMSMap.setLayerRenderer (myLayer, group);
```

Symbol

Description

The Symbol class is used within various IMSMap methods that can be appropriately applied to all types of renderers. 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 superclass 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 subclassed 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
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, Antialiasing, and Blockout. The TextSymbol object is used with the LabelRenderers.

TextSymbol.getAntialiasing

Description

Returns the 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

See Also

Symbol
TextSymbol.setAntialiasing

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var isAA = sym.getAntialiasing();
```

TextSymbol.getBlockoutColor

Description
Returns the 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
IMSMap
Symbol
TextSymbol.setBlockoutColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var rgb = sym.getBlockoutColor();
```

TextSymbol.getFontColor

Description
Returns the 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 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
IMSMap
Symbol
TextSymbol.setFontColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var rgb = sym.getFontColor();
```

TextSymbol.getFontName

Description
Returns the 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
IMSMap
Symbol
TextSymbol.setFont

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
var fontName = sym.getFontName();
```

TextSymbol.getFontSize

Description
Returns the size of the font used by the text symbol object.

Syntax
int getFontSize()

Arguments
none

Returned Value

int	The size of the font specified.

See Also
Symbol
TextSymbol.setFont

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
var fontSize = sym.getFontSize();
```

TextSymbol.getGlowColor

Description
Returns the 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
IMSMap
Symbol
TextSymbol.setGlowColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var rgb = sym.getGlowColor();
```

TextSymbol.getInterval

Description
Returns the interval of the TextSymbol.

Syntax
double getInterval()

Arguments
none

Returned Value

double	A valid double value.

See Also

IMSMap
Symbol
TextSymbol.setInterval

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var interval = sym.getInterval();
```

TextSymbol.getOutlineColor

Description
Returns the 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
TextSymbol.setOutlineColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
var rgb = sym.getOutlineColor();
```

TextSymbol.getPrintMode

Description
Returns the PrintMode of the symbol object.

Syntax
int getPrintMode()

Arguments
none

Returned Value
int The default value for PrintMode is 0, none.

0—None: No change is made to the label.
1—Proper Case: The first letter of each word in a label is uppercase and everything else is lowercase.
2—Uppercase: All letters are uppercase.
3—Lowercase: All letters are lowercase.

See Also
Symbol
TextSymbol.setPrintMode
ArcXML Programmer's Reference Guide

Example
```
//This sample demonstrates setting the printmode to always use UpperCase
var sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
var ret = sym.getPrintMode();
```

TextSymbol.getShadowColor

Returns the shadow color as a comma-delimited string containing the red, green, and blue color values.

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

IMSMap
Symbol
TextSymbol.setShadowColor

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
sym.setShadowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
var rgb = sym.getShadowColor();
// rgb = 255,0,0
```

TextSymbol.getTransparency

Description

Returns the transparency value set on the TextSymbol object.

Syntax

double getTransparency()

Arguments

none

Returned Value

double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also

IMSMap
Symbol
TextSymbol.setTransparency

Example

```
var sym = parent.mapFrame.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 antialiasing value used by the TextSymbol 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 enabled)

Arguments
enabled	Use “true” if antialiasing should be used, “false” otherwise.

Returned Value
boolean

See Also
Symbol
TextSymbol.getAntialiasing

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol (“TEXT_SYMBOL”);
sym.setAntialiasing (“true”);
```

TextSymbol.setBlockoutColor

Description
Sets the color value used by the Blockout effect of the text symbol object.

Syntax
boolean setBlockoutColor(Color newColor)

Arguments
newColor	Any valid Color object created using IMSMap.createColor(r,g,b).

Returned Value
boolean

See Also
Color
IMSMap
Symbol
TestSymbol.getBlockoutColor

Example
```
var  sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var ret = sym.setBlockoutColor(parent.mapFrame.IMSMap.createColor(255,255,255));
```

TextSymbol.setFont

Description

Sets the font for the text symbol.

Syntax

boolean setFont(String fontname, int fontsize)

Arguments

fontname	The name of the font to use.
fontsize	The size of the font to use.

Returned Value

boolean

See Also

IMSMap
Symbol
TextSymbol.getFontName
TextSymbol.getFontSize

Example

```
var myLayer = parent.mapFrame.IMSMap.getSelectedLayer():
var rend = parent.mapFrame.IMSMap.getLayerRenderer(myLayer);
var sym = rend.getSymbol();
if (!sym.setFont(“times”, 12) )
   //report the failure - do something…
```

TextSymbol.setFontColor

Description

Sets the color of the font drawn for the label of the feature.

Syntax

boolean setFontColor(Color newColor)

Arguments

newColor	Valid Color object obtained from the IMSMap.createColor (r,g,b) method.

Returned Value

boolean

See Also

Color
IMSMap
Symbol
TextSymbol.getFontColor

Example

```
var  sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var ret = sym.setFontColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TextSymbol.setGlowColor

Description
Sets the color used for the glowing effect.

Syntax
boolean setGlowColor(Color newColor)

Arguments

newColor	A valid Color object obtained from the IMSMap.createColor(r,g,b) method.

Returned Value
boolean

See Also
Color
IMSMap
Symbol
TextSymbol.getGlowColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var ret = sym.setGlowColor(parent.mapFrame.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

n	A valid double value.

Returned Value
boolean

See Also
Symbol
TextSymbol.getInterval

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var ret = sym.setInterval(1.0);
```

TextSymbol.setOutlineColor

Description
Sets the outline color of this object.

Syntax
boolean setOutlineColor(Color colorValue)

Arguments
colorValue	A valid Java Color object.

Returned Value
boolean

See Also
Symbol
TextSymbol.getOutlineColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
var ret = sym.setOutlineColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TextSymbol.setPrintMode

Description
Sets how the text of a label is displayed.

Syntax
boolean setPrintMode(int value)

Arguments
value	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 uppercase and everything else is lower case. 2—Uppercase: All letters are uppercase. 3—Lowercase: All letters are lowercase.

Returned Value
boolean

See Also
IMSMap
Symbol
TextSymbol.getPrintMode
ArcXML Programmer's Reference Guide

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
if (sym.setPrintMode(1))alert(“Success”);
```

TextSymbol.setShadowColor

Description

Sets the color value for the shadow effect on the TextSymbol. The shadow effect methods provide for a mirrored image or reflection to appear behind and below the text as it is labeled. The shadow will be drawn using a 0.5 transparency.

Syntax

boolean setShadowColor(Color colorValue)

Arguments

colorValue	A valid color value retrieved from the IMSMap.createColor() method.

Returned Value

boolean

See Also

Color
IMSMap
Symbol
TextSymbol.getShadowColor

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
sym.setShadowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TextSymbol.setTransparency

Description

Sets the transparency value of the text symbol.

Syntax

boolean setTransparency(double n)

Arguments

n	The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value

boolean

See Also

IMSMap
Symbol
TextSymbol.getTransparency

Example

```
var sym = parent.mapFrame.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 scalable vector fonts that 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 that it 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, Antialiasing, and Blockout.

TrueTypeMarkerSymbol.getAngle

Description

Returns the angle of rotation of the symbol. 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

IMSMap
Symbol
TrueTypeMarkerSymbol.setAngle
ArcXML Programmer's Reference Guide

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var angle = sym.getAngle();
```

TrueTypeMarkerSymbol.getAntialiasing

Description
Returns the antialiasing value of the symbol. Antialiasing is the process of adding pixels along lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax
boolean getAntialiasing()

Arguments
none

Returned Value
boolean

See Also
Symbol

TrueTypeMarkerSymbol.setAntialiasing

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var val = sym.getAntialiasing();
```

TrueTypeMarkerSymbol.getBlockoutColor

Description
Returns the color value for the blockout effect of the symbol.

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
IMSMap
Symbol
TrueTypeMarkerSymbol.setBlockoutColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var rgb = sym.getBlockoutColor();
```

TrueTypeMarkerSymbol.getCharacter

Description
Returns the character code in the font associated with the symbol.

Syntax
String getCharacter()

Arguments
none

Returned Value

String	The string representation of the character.

See Also
IMSMap
Symbol
TrueTypeMarkerSymbol.setCharacter

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var char = sym.getCharacter();
alert(“The character represented in the TrueTypemarker is:” + char);
```

TrueTypeMarkerSymbol.getFontColor

Description
Returns 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
IMSMap
Symbol
TrueTypeMarkerSymbol.setFontColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var rgb = sym.getFontColor();
```

TrueTypeMarkerSymbol.getFontName

Description
Returns the name of the font used by the symbol.

Syntax
String getFontName()

Arguments
none

Returned Value

String	The name of the font used.

See Also
IMSMap
Symbol
TrueTypeMarkerSymbol.setFont

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var fontName = sym.getFontName();
alert(“The font name is:” + fontName);
```

TrueTypeMarkerSymbol.getFontSize

Description
Returns the size of the font for the symbol.

Syntax
int getFontSize()

Arguments
none

Returned Value

int	The size of the font specified.

See Also
Symbol
TrueTypeMarkerSymbol.setFont

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var fontSize = sym.getFontSize();
```

TrueTypeMarkerSymbol.getGlowColor

Description
Returns the color of the glow effect for the symbol.

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
IMSMap
Symbol
TrueTypeMarkerSymbol.setGlowColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var rgb = sym.getGlowColor();
```

TrueTypeMarkerSymbol.getOutlineColor

Description
Returns the color for the outline 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
Color
IMSMap
Symbol
TrueTypeMarkerSymbol.setOutlineColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var rgb = sym.getOutlineColor();
```

TrueTypeMarkerSymbol.getShadowColor

Description
Returns the shadow color for the symbol.

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 bebetween 0 and 255
G	defines green part of the RGB color value, should bebetween 0 and 255
B	defines blue part of the RGB color value, should bebetween 0 and 255

See Also
IMSMap
Symbol
TrueTypeMarkerSymbol.setShadowColor

Example
```
var Symbol = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var newColor = parent.mapFrame.IMSMap.createColor(255,0,0);
Symbol.setShadowColor(newColor);

if (Symbol == null)
   alert("Error: Symbol is null");
else alert ("the shadow color of the TrueTypeMarkerSymbol is" +
Symbol.getShadowColor());
```

TrueTypeMarkerSymbol.getTransparency

Description
Returns the transparency value of the symbol.

Syntax
double getTransparency()

Arguments
none

Returned Value
double	The default value is 1.0. The valid range is from 0.0 (transparent) to 1.0 (opaque).

See Also
Symbol
TrueTypeMarkerSymbol.setTransparency

Example
```
var sym = parent.mapFrame.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 angle)

Arguments

angle	A valid double number in the range of 0.0–360.0; the default value is 0.

Returned Value

boolean

See Also

IMSMap
Symbol
TrueTypeMarkerSymbol.getAngle
ArcXML Programmer's Reference Guide

Example

The following example demonstrates setting the rotation angle at 240 degrees.

```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var ret = sym.setAngle(240);
```

TrueTypeMarkerSymbol.setAntialiasing

Description

Sets the antialiasing value used by the symbol. Antialiasing is the process of adding pixels along lines to smooth the jagged appearance. Antialiasing is off by default.

Syntax

boolean setAntialiasing(String enabled)

Arguments

enabled	“True” to turn this option on, “false” to turn it off.

Returned Value

boolean

See Also

Symbol
TextSymbol.getAntialiasing

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
boolean ret = sym.setAntialiasing(“true”);
```

TrueTypeMarkerSymbol.setBlockoutColor

Description
Sets the color value used by the blockout effect of the symbol.

Syntax
boolean setBlockoutColor(Color newColor)

Arguments

newColor	Any valid color object created using the IMSMap.createColor(r,g,b) method.

Returned Value
boolean

See Also
Color
IMSMap
Symbol
TrueTypeMarkerSymbol.getBlockoutColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var ret = sym.setBlockoutColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setCharacter

Description
Sets the character code in the font associated with the TrueTypeMarkerSymbol.

Syntax
boolean setCharacter(String character)

Arguments

character	A valid string/index value to represent the character.

Returned Value
boolean

See Also
IMSMap
Symbol
TrueTypeMarkerSymbol.getCharacter

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var ret = sym.setCharacter(240);
```

TrueTypeMarkerSymbol.setFont

Description
Sets the font of the symbol.

Syntax
boolean setFont(String fontname, int fontsize)

Arguments

fontname	The name of the font to use.
fontsize	The size of the font to use.

Returned Value
boolean

See Also
Symbol
TrueTypeMarkerSymbol.getFontName
TrueTypeMarkerSymbol.getFontSize

Example
```
var sym = parent.mapFrame.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 symbol.

Syntax
boolean setFontColor(Color newColor)

Arguments

newColor	Any valid Color object created using the IMSMap.createColor(r,g,b) method.

Returned Value
boolean

See Also
Color
Symbol
TrueTypeMarkerSymbol.getFontColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol(“TRUETYPE_MARKER_SYMBOL”);
var ret = sym.setFontColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setGlowColor

Description
Sets the color for the glow effect of the symbol.

Syntax
boolean setGlowColor(Color newColor)

Arguments

newColor	Any valid Color object created using the IMSMap.createColor(r,g,b) method.

Returned Value
boolean

See Also
Color
IMSMap
Symbol
TrueTypeMarkerSymbol.getGlowColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret = sym.setGlowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setOutlineColor

Description
Sets the outline color of the symbol.

Syntax
boolean setOutlineColor(Color colorValue)

Arguments

colorValue	Any valid Color object created using the IMSMap.createColor(r,g,b) method.

Returned Value
boolean

See Also
Color
IMSMap
Symbol
TrueTypeMarkerSymbol.getOutlineColor

Example
```
var sym = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_MARKER_SYMBOL");
var ret = sym.setOutlineColor(parent.mapFrame.IMSMap.createColor(255,0,0));
```

TrueTypeMarkerSymbol.setShadowColor

Description

Sets the shadow effect on the symbol. The shadow effect is 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 TrueTypeMarkerSymbol. The shadow will be drawn using a 0.5 transparency.

Syntax

boolean setShadowColor(Color colorValue)

Arguments

color Value	A valid Color object using the IMSMap.createColor(r,g,b) method.

Returned Value

boolean

See Also

Color
IMSMap
Symbol
TrueTypeMarkerSymbol.getShadowColor

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_MARKER_SYMBOL");
if (sym.setShadowColor(parent.mapFrame.IMSMap.createColor(255,0,0)) ) {
   //do something.
}
```

TrueTypeMarkerSymbol.setTransparency

Description

Sets a level of transparency on the symbol.

Syntax

boolean setTransparency(double n)

Arguments

n	The valid range is from 0.0 (transparent) to 1.0 (opaque).

Returned Value

boolean

See Also

Symbol
TrueTypeMarkerSymbol.getTransparency

Example

```
var sym = parent.mapFrame.IMSMap.createSymbol("TRUETYPE_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

Returns the 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

IMSMap
Renderer
ValueMapLabelRenderer.setSeparator

Example

```
var myLayer =
   parent.mapFrame.IMSMap.getSelectedLayer(); // Layer with valueMapLabelRenderer
var valueMapLabelRenderer;
valueMapLabelRenderer = parent.mapFrame.IMSMap.getLayerLabelRenderer (myLayer);
var separatorChar = valueMapLabelRenderer.getSeparator();
alert("The separator used is:" + separatorChar);
```

ValueMapLabelRenderer.setDefaultSymbol

Description

Sets the default symbol renderer object 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 sym)

Arguments

sym	A text symbol object that controls how text is rendered.

Returned Value

boolean

See Also

Renderer
Symbol
ArcXML Programmer's Reference Guide

Example

```
var sym, valuemapLabelRenderer;
var range;
var myLayer =
   parent.mapFrame.IMSMap.getSelectedLayer();  // Layer with valueMapLabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
valuemapLabelRenderer =
   parent.mapFrame.IMSMap.createRenderer(“VALUE_LABEL_RENDERER”);
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
sym.setFontColor(parent.mapFrame.IMSMap.createColor(255,255,255));
valuemapLabelRenderer.setDefaultSymbol(sym);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “0”, “200”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(178,176,0));
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);
range = parent.map.Frame.IMSMap.createValueRange(myLayer, fieldName, “201”, “400”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “401”, “600”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(0,0,245));
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, valuemapLabelRenderer);
```

ValueMapLabelRenderer.setField

Description

Sets a field to draw text on a feature.

Syntax

boolean setField(Layer layer, String fieldName)

Arguments

layer	The layer object to apply the field to.
fieldName	The field name used to store values to draw text on a feature.

Returned Value

boolean

See Also

IMSMap
Layer
Renderer

Example

```
var sym, valueMapLabelRenderer;
var myLayer =
  parent.mapFrame.IMSMap.getSelectedLayer();  // Layer with valueMapLabelRenderer
sym = imsmap.createSymbol(“TEXT_SYMBOL”);
valueMapLabelRenderer=parent.mapFrame.IMSMap.createRenderer(“VALUE_LABEL_RENDERER”);
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
valueMapLabelRenderer.setSymbol(sym);
valueMapLabelRenderer.setField(myLayer, “field1”);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, valueMapLabelRenderer);
```

ValueMapLabelRenderer.setFields

Description

Sets multiple fields to draw text on a feature. Use the setSeparator method to identify the string used in the display between fields.

Syntax

boolean setFields(Layer layer, Collection col)

Arguments

layer	The layer object to apply the fields on.
col	The collection object that contains the field names to use.

Returned Value

boolean

See Also

Collection
Layer
Renderer
ArcXML Programmer's Reference Guide

Example

```
var col;
var sym, valueMapLabelRenderer;
var myLayer =
   parent.mapFrame.IMSMap.getSelectedLayer();  // Layer with valueMapLabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
valueMapLabelRenderer =
parent.mapFrame.IMSMap.createRenderer("VALUE_LABEL_RENDERER");
col = parent.mapFrame.IMSMap.createCollection();
col.addStringElement("field1");
col.addStringElement("field2");
sym.setFont("times", 12);
sym.setAntialiasing("true");
valueMapLabelRenderer.setSymbol(sym);
valueMapLabelRenderer.setSeparator(":");
valueMapLabelRenderer.setFields(myLayer, col);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, valueMapLabelRenderer);
```

ValueMapLabelRenderer.setSeparator

Description
Sets a separator character when multiple fields are used to display labels on a feature.

Syntax
boolean setSeparator(String separator)

Arguments

separator	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

See Also
IMSMap
Renderer
ValueMapLabelRenderer.getSeparator

Example
```
var col;
var sym, valueMapLabelRenderer;
var myLayer =
  parent.mapFrame.IMSMap.getSelectedLayer();  // Layer with valueMapLabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol(“TEXT_SYMBOL”);
valueMapLabelRenderer = parent.mapFrame.IMSMap.createRenderer(“VALUE_LABEL_RENDERER”);
col = parent.mapFrame.IMSMap.createCollection();
col.addStringElement(“field1”);
col.addStringElement(“field2”);
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
valueMapLabelRenderer.setSymbol(sym);
valueMapLabelRenderer.setSeparator(“:”);
valueMapLabelRenderer.setFields(myLayer, col);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, valueMapLabelRenderer);
```

ValueMapLabelRenderer.setSymbolForRangeValue

Description

Sets range value and a symbol to be used for that range. A range value is used to represent a way of classifying features into categories or classes by drawing different symbols for text specified by the 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

symObject	A symbol object that is used to control how text is rendered.
range	A range that defines a pair of comparable values.

Returned Value

boolean

See Also

IMSMap
Renderer
Symbol
ValueRange

Example

```
var sym, valuemapLabelRenderer;
var range;
var myLayer =
   parent.mapFrame.IMSMap.getSelectedLayer();  // Layer with valueMapLabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
valuemapLabelRenderer =
parent.mapFrame.IMSMap.createRenderer("VALUE_LABEL_RENDERER");
sym.setFont("times", 12);
sym.setAntialiasing("true");
sym.setFontColor(parent.mapFrame.IMSMap.createColor(255,255,255));
valuemapLabelRenderer.setDefaultSymbol(sym);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, "0", "200");
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(178,176,0));
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, "201", "400");
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, "401", "600");
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(0,0,245));
valuemapLabelRenderer.setSymbolForRangeValue(sym, range);
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, valuemapLabelRenderer);
```

ValueMapLabelRenderer.setSymbolForUniqueValue

Description

Sets unique values to the LabelRenderer using a field obtained from the layer and the value to assign the symbol. This method is a way of symbolizing text to be drawn on features of a layer by drawing a symbol for each unique value.

Syntax

boolean setSymbolForUniqueValue(Symbol sym, Layer layer, String fieldname, String value)

Arguments

sym	A symbol object that is used to control how text is rendered.
layer	The layer object to apply unique value.
fieldname	The name of the field obtained from the layer.
value	Unique value of the field to apply this symbol.

Returned Value

boolean

See Also

IMSMap
Layer
Renderer
Symbol

Example

```
var col;
var sym, valuemapLabelRenderer;
var myLayer =
   parent.mapFrame.IMSMap.getSelectedLayer();  // Layer with valueMapLabelRenderer
sym = parent.mapFrame.IMSMap.createSymbol("TEXT_SYMBOL");
sym.setFont("times", 12);
sym.setAntialiasing("true");
valuemapLabelRenderer = parent.mapFrame.IMSMap.createRenderer("VALUE_LABEL_RENDERER");
valuemapLabelRenderer.setSymbolForUniqueValue(sym, myLayer, "field1", "200");
valuemapLabelRenderer.setSymbolForUniqueValue(sym, myLayer, "field1", "300");
valuemapLabelRenderer.setSymbolForUniqueValue(sym, myLayer, "field1", "400");
valuemapLabelRenderer.setSymbol(sym);
valuemapLabelRenderer.setField(myLayer, "field1");
parent.mapFrame.IMSMap.setLayerLabelRenderer(myLayer, valuemapLabelRenderer);
```

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

Returns 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

IMSMap
Layer
Renderer
ValueMapRenderer.setField

Example

```
var valRenderer, field;
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
valRenderer = parent.mapFrame.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 sym)

Arguments

sym	A symbol object that controls how features are rendered.

Returned Value

boolean

See Also

Renderer
Symbol
ArcXML Programmer's Reference Guide

Example

```
var sym, valRenderer, range;
var myLayer =
  parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
valRenderer = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
sym.setFontColor(parent.mapFrameIMSMap.createColor(255,255,255));
valRenderer.setDefaultSymbol(sym);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “0”, “200”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(0,0,34));
valRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “201”, “400”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
valRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “401”, “600”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(100,200,0));
valRenderer.setSymbolForRangeValue(sym, range);
valRenderer.setField(myLayer, fieldName);
parent.mapFrame.IMSMap.setLayerRenderer(myLayer, valRenderer);
```

ValueMapRenderer.setField

Description
Sets a single field to classify the features.

Syntax
boolean setField(Layer layer, String fieldName)

Arguments

layer	The layer object to apply the field to.
fieldName	The Field name used to store values used to draw text on a feature.

Returned Value
boolean

See Also
IMSMap
Layer
Renderer

Example
```
var sym, valRenderer;
var myLayer =
  parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
valRenderer = parent.mapFrame.IMSMap.createRenderer("VALUEMAP_RENDERER");
sym.setFont("times", 12);
sym.setAntialiasing("true");
valRenderer.setSymbol(sym);
valRenderer.setField(myLayer, "field1");
parent.mapFrame.IMSMap.setLayerRenderer (myLayer, valRenderer);
```

ValueMapRenderer.setSymbolForRangeValue

Description
Sets the range value and a symbol to be used for that range. 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. When the Display Engine draws the feature, it will use the correct symbol for the value of the field.

Syntax
boolean setSymbolForRangeValue(Symbol sym, ValueRange range)

Arguments

sym	A symbol object that is used to control how text is rendered.
range	A range defines a pair of comparable values.

Returned Value
boolean

See Also
IMSMap
Renderer
Symbol
ValueRange

Example
See the next page.

ValueMapRenderer.setSymbolForRangeValue

Example

```
var sym, valRenderer, range;
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
valRenderer = parent.mapFrame.IMSMap.createRenderer(“VALUEMAP_RENDERER”);
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
sym.setFontColor(parent.mapFrame.IMSMap.createColor(255,255,255));
valRenderer.setDefaultSymbol(sym);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “0”, “200”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(0,0,34));
valRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “201”, “400”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(255,0,0));
valRenderer.setSymbolForRangeValue(sym, range);
range = parent.mapFrame.IMSMap.createValueRange(myLayer, fieldName, “401”, “600”);
sym.setGlowColor(parent.mapFrame.IMSMap.createColor(100,200,0));
valRenderer.setSymbolForRangeValue(sym, range);
valRenderer.setField(myLayer, fieldName);
parent.mapFrame.IMSMap.setLayerRenderer(myLayer, valRenderer);
```

ValueMapRenderer.setSymbolForUniqueValue

Description

Sets unique values to the Renderer using a field obtained from the layer and the value to assign the symbol. This method represents a way of symbolizing features of a layer by drawing a symbol for each unique data value. When the Display Engine draws the feature, it will use the correct symbol for the value of the field.

Syntax

boolean setSymbolForUniqueValue(Symbol sym, Layer layer, String fieldName, String value)

Arguments

sym	A symbol object that is used to control how text is rendered.
layer	The layer object to apply a unique value.
fieldName	The name of the field obtained from the layer.
value	Unique value of the field to apply this symbol.

Returned Value

boolean

See Also

IMSMap
Layer
Renderer
Symbol

Example

See the next page.

ValueMapRenderer.setSymbolForUniqueValue

Example

```
var col;
var sym, ValueMapRenderer ;
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
sym = parent.mapFrame.IMSMap.createSymbol();
sym.setFont(“times”, 12);
sym.setAntialiasing(“true”);
ValueMapRenderer  = parent.mapFrame.IMSMap.createRenderer(“VALUE_LABEL_RENDERER”);
ValueMapRenderer.setSymbolForUniqueValue(sym, myLayer, “field1”, “200”);
ValueMapRenderer.setSymbolForUniqueValue(sym, myLayer, “field1”, “300”);
ValueMapRenderer.setSymbolForUniqueValue(sym, myLayer, “field1”, “400”);
ValueMapRenderer.setSymbol(sym);
ValueMapRenderer.setField(myLayer, “field1”);
parent.mapFrame.IMSMap.setLayerRenderer(myLayer, ValueMapRenderer);
```

ValueRange

A range defines a pair of comparable values that can be used with a ValueMapRenderer.

ValueRange.getLower

Description

Returns the lower range value as defined by the setLower method.

Syntax

String getLower()

Arguments

none

Returned Value

String	The lower value; the literal “None” is returned if no field is specified.

See Also

Symbol
ValueRange.setLower
ArcXML Programmer's Reference Guide

Example

```
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
var range = parent.mapFrame.IMSMap.createValueRange (myLayer, fieldName, “0”,
“200”);
var val=range.getLower();
```

ValueRange.getUpper

Description
Returns the upper range value as defined by the setUpper method.

Syntax
String getUpper()

Arguments
none

Returned Value
String	The upper value; the literal “None” is returned if no field is specified.

See Also
Symbol
ValueRange.setUpper
ArcXML Programmer's Reference Guide

Example
```
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
var range = parent.mapFrame.IMSMap.createValueRange (myLayer, fieldName, “0”,
“200”);
var val = range.getUpper();
```

ValueRange.inRange

Description
Determines 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 layer, String fieldName, String value)

Arguments
layer	A valid layer object from the IMSMap.
fieldName	Field Name to use within the layer.
value	The upper value to use in this range.

Returned Value
boolean

See Also
IMSMap
Symbol

Example
See the next page.

ValueRange.inRange

Example

```
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
var range = parent.mapFrame.IMSMap.createValueRange (myLayer, fieldName, “0”,
“200”);
if (range.inRange(myLayer, “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 value)

Arguments

value	The lower value to use in this range, the literal “None” is returned if no field is specified.

Returned Value

boolean

See Also

IMSMap
Symbol
ValueRange.getLower

Example

```
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
var range = parent.mapFrame.IMSMap.createValueRange (myLayer, fieldName, “0”,
“200”);
var 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 layer, String fieldName, String value)

Arguments

layer	A valid layer object from the IMSMap.
fieldName	Field Name to use within the layer.
value	The upper value to use in this range.

Returned Value

boolean

See Also

IMSMap
Symbol
ValueRange.getUpper

Example

```
var myLayer =
   parent.mapFrame.IMSMap.getSelected Layer(); //Layer with ValueMapRenderer
var range = parent.mapFrame.IMSMap.createValueRange (myLayer, fieldName, “0”,
“200”);
If(range.setUpper(myLayer, “name”, 100) ) {
   alert(“Upper range was set successfully”);
} else {
   alert(“Setting the upper range failed”);
}
```