Flashmaps AreaSelector Developer's Guide

Version 3.3.2

flashmaps

pushing the limits of mapping

Index

1. Overview

Basically, AreaSelector is an interactive high-scale map to integrate in your website.

As you will learn in this guide, AreaSelector is fully customizable, offering many ways to interact with your website and with your database. For example, it allows you to color code areas depending on your database (e.g., total sales per state, for example) or plot your international branches on a world map.

In fact, AreaSelector is not only a piece of software, but also a turn-key combination of software and data (mainly maps). At this stage the AreaSelector series includes:

- World: continents, countries, states/regions/provinces, etc.
- US Administrative Areas: states, counties, Upper and Lower House districts, Congress districts, townships, MSAs (Metropolitan Statistical Areas), etc.
- Zip codes
- Area codes
- Custom areas like sales regions, markets, assistance areas, etc.

If you need to map any other kind of area, please ask our representatives. AreaSelector is designed to support any kind of large areas.

Think about AreaSelector as a "map machine" available to provide your databases with mapping capabilities.

Basically, AreaSelector is an Adobe Flash object which offers an API consisting of a series of JavaScript/Flash functions. These functions will allow you to send commands to the AreaSelector, like loading a new map, focusing on a specific area, etc. In a nutshell, this API (Application Programming Interface) and the XML files permit a complete integration of AreaSelector into your existing or new web pages in a flexible, dynamic and easy way.

AreaSelector configures the areas and POIs (Points of Interest) on the map by reading the corresponding XML feeds, which may be plain XML files or dynamic pages (php, asp, etc.) returning XML code. You can set up area colors, icons to display the POIs, actions to take place when the user clicks on an area, and many other things.

This guide provides complete information about AreaSelector, and the best practices to integrate it into your website.

1.1. Builder Care

All technologies inside AreaSelector have been developed at Flashmaps from scratch. This enables us to implement virtually every maps or functionalities you may need to fit your requirements.

Please email us at support@flashmaps.com or call us toll free at **1-866-392-0071** to share your most ambitious mapping wishes with us, or let us know if we can assist you in any way.

1.2. Files

AreaSelector comes with a set of pre-defined files which are listed and shortly described in the following table:

Under fmASMap folder:

fmASToolbar.swf	AreaSelector's basic interface.
fmASEngine.swf	AreaSelector's engine.
fmASEngine.xml	AreaSelector's configuration.
fmASIconLibrary.fla	Icon library (source).
fmASIconLibrary.swf	Icon library (binary).
fmASContainer.fla	Container embedding the map (source).
fmASContainer.swf	Container embedding the map (binary).

Under areas (or similar) folder:

fm-us.xml, fm-world.xml, etc.	Theme configuration.
area_categories.xml (or similar)	Area categories.
areas.xml,.php, etc.	Areas to be displayed.

Under POIs (or similar) folder:

poi_categories.xml (or similar)	POI categories.
pois.xml, .php, etc.	POIs (Points of Interest).

Legend with the color-coding of the file names:

In red	Files you should not change at all.
In green	Files that you may change, but you will always need.
In blue	Files that may vary depending on the project.

* Usually our Flashmaps AreaSelector solutions are optimized for our customer's wishes and needs, which means that, depending upon your individual requirements, your project may include more or less customizations. This, however, sometimes implicates that there might be additional files or alternative versions of some files (e.g. different fmASEngine.xml configuration files) which, therefore, are not listed here in the table and may differ in their naming.

1.3. AreaSelector Architecture

Basically, AreaSelector is built up of cartography, themes, areas and POIs (Points of interest). Find below a description of these concepts, their role inside AreaSelector and some examples of how they need to be defined.

Find in this section a brief description and samples of the main components of AreaSelector. Find the full description of all the XML feeds in the *XML Feeds Reference* section later in this guide.

1.3.1. Cartography

The flashmaps AreaSelector uses flash movie clips (swf) as map cartography. Depending upon your chosen product, this can be just one file or many. Generally, a Flashmaps AreaSelector consisting of two map levels needs just one flash cartography file; for instance, AreaSelector US states (level 1: US view, level 2: state view). AreaSelectors with a third map level usually loads child maps. For instance, the Flashmaps AreaSelector US States + Counties displays a flash cartography file with the US and the state divisions at map level 1. Zooming into map level 2, the child flash cartography files will be loaded, displaying the respective US state with the county divisions.

These cartography files contain areas which form the basis of Flashmaps AreaSelector and can be customized in many ways. Find more information in the Areas section below.

The cartography files in your project depend upon the product version, and are normally located in the folder fmASMap/area and subfolders or in the folder fmASMap/maps and subfolders. For example, a plain AreaSelector World will have cartography files completely different from an AreaSelector USA with Area Codes.

1.3.2. Themes

AreaSelector works with themes. The theme is an XML feed (normally an XML file) which provides the necessary information to display a specific map, such as: what area to display, colors, POIs (Points of interest), etc. Think of a AreaSelector as a word processing software, and think of a theme as document you use under it, which can be completely different to another (different areas, different colors, different POIs, different behaviors, etc.).

If a child map will be loaded when zooming into an area, a new theme needs to be loaded in order to configure the new map view.

This is what a theme looks like:

```
<theme id="us">
        <map file="areas/fm-us.swf" borderColor="0xCCCCCC" />
        <areas xmlAreas="areas/areas.xml" xmlCategories="areas/area_categories.xml" />
        <pois xmlPOIs="pois/pois.xml" xmlCategories="pois/poi_categories.xml" />
</theme>
```

As you can see, the theme specifies: a map (a SWF file, a Flash movieclip); the corresponding set of areas to display and interact with (an XML feed); and, optionally, a set of POIs (Points of Interest) to display and interact with, too (another XML feed).

You can make AreaSelector load different themes, corresponding to different information which can be displayed in a dynamic way. For example, you may have a theme displaying the map of USA color-coding the states by your last year sales, and another one color-coding the states by population and including the top 100 cities in the country.

Find the full description of theme XML feed in the *XML Feeds Reference* section of this guide.

1.3.3. Areas

An AreaSelector map always contains a set of areas. One of the most popular features of the system is clicking on an area to dynamically zoom on it.

Areas have a set of properties which can be configured dynamically (according to database values), like changing their colors, displaying area's tooltip, or actions to launch on mouse over, click, etc.

Areas are grouped in categories to ease the application of their properties and events.

Inside the theme you need to specify the URLs for the set of areas and their categories.

In addition to set colors and events to a full category, you can set them to each area individually.

All the information about areas and their categories is stored in static XML files or dynamic pages that output the corresponding XML format:

Area Categories XML Feed

This XML feed describes the colors, actions and some other information about each areas category. For example:

```
<areaCategories>
        <category id='cat1' zoom='true' enabled='true'
tooltip='popups/area.swf' poitextField='state'
poitextCategory='state' >
                <color up='0xCEDBFC' over='0xFEDF74'
down='0xFEDF74' />
        </category>
        <category id='cat2' zoom='true' enabled='true'
tooltip='popups/area.swf' poitextField='state'
poitextCategory='state' >
                <color up='0xA167C0' over='0xFEDF74'
down='0xFEDF74' />
        </category>
</areaCategories>
```

Areas XML Feed

This XML feed describes the ID, label and category of each area. For example:

```
<areas>
        <area id='us_az' category='cat1' label='Arizona' state='AZ' />
        <area id='us_ca' category='cat2' label='California' state='CA' />
        <area id='us_co' category='cat1' label='Colorado' state='CO' />
        ...
</areas>
```

Find the full description of areas and area categories XML feeds in the *XML Feeds Reference* section of this guide.

1.3.4. POIs (Points of Interest)

POIs (Points of Interest) let you show on the map any kind of locations, such as company offices, stores, event sites, etc.

You can display the same set of POIs on all zoom levels (states and counties, for example), or make AreaSelector load a new set of POIs when you enter an area; for example, when you click and zoom into a state. This is especially useful when you have a very large number of POIs, and displaying them all in the initial view could make the map too confusing.

The POI location is defined by its real coordinates (latitude and longitude). Coordinates are always given in decimal form (GGG.GGGG), rather than sexagesimal or base-sixty (GGG.MMSS).

In addition to setting up the icon and events to a full category of POIs, you can set them individually to each POI.

All the information about POIs and their categories are stored in static XML files or dynamic pages that output the corresponding XML format:

POI Categories XML Feed

This XML feed describes the icon, actions and some other information about each POIs category. For example:

```
<POICategories>
	<category id='eastern' enabled='true' iconUrl='eastern'
		iconEmbedded='true' tooltip='popups/team.swf' >
		<action event='onRelease' target='_MC'
			url='popups/team_info.swf' />
	</category>
	<category id='western' enabled='true' iconUrl='western'
		iconEmbedded='true' tooltip='popups/team.swf' >
		<action event='onRelease' target='_MC'
			url='popups/team_info.swf' />
	</category>
</POICategories>
```

POIs XML Feed

This XML feed describes the ID, label, category and coordinates of each POI. For example:

```
<POIs>
	<POI id='2' category='eastern' label='Boston Celtics'
		lat='42.36391' lon='-71.06285' />
	<POI id='20' category='eastern' label='New York Knicks'
		lat='40.74941' lon='-73.99218' />
	<POI id='7' category='western' label='Denver Nuggets'
		lat='39.74549' lon='-105.00982' />
	<POI id='17' category='western' label='Minnesota Timberwolves'
		lat='44.97958' lon='-93.27487' />
</POIs>
```

Find the full description of POIs and POI categories XML feeds in the *XML Feeds Reference* section of this guide.

On the map, a POI is represented by a little icon. Normally you want to display a different icon per POI category, although you may also specify the icon individually for each POI.

For optimal display performance, the icons are normally compiled into the Icon Library, which is a Flash file (fmASIconLibrary.fla) where you can easily change the icons we provide or add new ones. Go to chapter *5.2,. Insert or modify an icon in the library,* for detailed instructions on this topic.

You may also use external icons if you need to plot icons created on the fly, if you enable users to add new icons or if, for any reason, you cannot predict which icons need to be displayed. Please note that, since Microsoft Internet Explorer will not cache the file, external icons have to be loaded for each POI that uses them, which might lead to slower performance of the map if displaying a high number of POIs.

MultiPOIs

When several POIs have the same coordinates, one of them would be on top of the rest, preventing the user from seeing or clicking on them. In order to avoid this, a special category for such POIs might be created to merge all the coincident POIs into a "multiPOI".

If you need a MultiPOIs solution for your Flashmaps AreaSelector, please share your specific needs with us. The most popular solution is to have a special MultiPOI icon with a special MultiPOI behavior. When you hover over a multiPOI, AreaSelector will warn you about the multiplicity of POIs there. When you click on a multiPOI, AreaSelector pops up a list of the members, letting you select any of them.
You can change that icon by editing a symbol named 'multipoi' in the icon library (fmASIconLibrary.fla).

1.3.5. Polylines

You can display multiple-point lines (polylines) over your maps in AreaSelector, to indicate routes or any kind of relationship between areas or POIs.

You can display the same set of polylines on all zoom levels (states and counties, for example), or make AreaSelector load a new set of polylines when you enter an area; for example, when you click and zoom into a state.

The polyline's vertexes are defined by their real coordinates (latitude and longitude).

In addition to setting up the look and events to a full category of polylines, you can set them individually to each polyline.

All the information about polylines and their categories is stored in static XML files or dynamic pages that output the corresponding XML format:

Polyline Categories XML Feed

This XML feed describes the color, thickness, actions and some other information about each polylines category. For example:

```
<PolylineCategories>
        <category id="city" enabled="true" color="0xFF0000" thickness="2"
                tooltip="popup.swf" >
                <action event="onRelease" target="_blank" url="default.html" />
        </category>
</PolylineCategories>
```

Polylines XML Feed

This XML feed describes the ID, label, category and vertexes coordinates of each polyline. For example:

```
<polylines>
        <polyline id="1" category="line" label="Line 1" >
                <vertex lat="43.12" lon="-73.12" />
                <vertex lat="44.12" lon="-73.12" />
```

```
            ...
        </polyline>
        ...
    </polylines>
```

Find the full description of polylines and polyline categories XML feeds in the *XML Feeds Reference* section of this guide.

2. XML Feeds Reference

2.1. fmASEngine.xml

fmASEngine.xml is AreaSelector's general configuration file. It needs to be located in the same directory as fmASEngine.swf.

Editing this file could affect the normal operation of AreaSelector, so make a backup copy before making any changes.

This is an example of what fmASEngine.xml looks like:

```
<fmEngine>
	<version>003.003.001</version>
	<mapSize>63074184-36925814-24792479-25703581</mapSize>
	<serialNumber>fmAS-0014-6267</serialNumber>
	<affiliateID></affiliateID>
	<theme>test/fm-us.xml</theme>
	<map mapWidth="1000" mapHeight="600" mapScale="20100000"
		mapScaleMax="100000" centerLat="-0.31" centerLon="0.47"
		transactionSteps="5" />
	<mapInitial initialX="1355" initialY="510" initialScale="100" />
	<mapSelect color="0xFFF000" divider="," />
	<mapConfig backLabel="Back to main map" showLabel="true" areasAlpha="40"
		popupCloseWhenClick="true" />
	<mapZoom zoom="true" zoomMouse="true" pan="true" />
	<levels poisSendParams="true" areasSendParams="true" randomLoad="false">
		<level id="1" showPOIText="false" showAreaTooltip="true"
			showAreaSelectedTooltip="true"  showPOITooltip="true"
			poisLoad="true" poisShow="true" poisDelay="0" filterId=""
			poisRefreshOnViewChange="true" />
		<level id="2" showPOIText="true"  showAreaTooltip="true"
			showAreaSelectedTooltip="false" showPOITooltip="true"
			poisLoad="false" poisShow="true" poisDelay="0" filterId=""
			poisRefreshOnViewChange="true" />
	</levels>
	<categories>
		<filter id="glow" filterType="glow" alpha="1" blurX="4" blurY="4"
			color="0x000000" quality="1" strength="1" />
		<filter id="shadow" filterType="shadow" angle="60" distance="3"
			alpha="0.7" blurX="4" blurY="4" color="0x333333" />
		<filter id="blur" filterType="blur"  blurX="4" blurY="4"
			quality="1" />
		<filter id="bevel" filterType="bevel"  distance="4" angle="90"
			highlightColor="0xFFFFFF" highlightAlpha="0.5"
			shadowColor="0x666666" shadowAlpha="0.5" blurX="4" blurY="4"
			type="outer" />
		<text id="state" filterId="none" fontName="_sans" fontSize="11"
			fontBold="no" fontItalic="no" fontAlign="center"
			fontColor="0x999999" />
	</categories>
```

```
</fmEngine>
```

Let us look inside each of the attributes and nodes of this XML feed:

```
<version>003.003.001</version>
```

Version node is the version of your AreaSelector system.

```
<theme>areas/fm-us.xml</theme>
```

Theme node is the theme to load initially. If you set the parameter 'customTheme' when instancing AreaSelector, this node will override it. Note that you are able to load another theme later on.

```
<map mapWidth="1000" mapHeight="600" mapScale="20100000" mapScaleMax="4000" centerLat="-0.31" centerLon="0.47" transactionSteps="5" />
```

Map node includes some important information about your system:

mapWidth: AreaSelector's width in pixels. **Do not change this value.**

mapHeight: AreaSelector's height in pixels. **Do not change this value.**

mapScale: Map's internal true scale. **Do not change this value.**

mapScaleMax: Maximum scale (closest view) a map is displayed at. It is a percentage of the real size. The bigger value, the closer you can zoom in.

transactionSteps: Number of steps when dynamically zooming in or out.

```
<mapInitial initialX="1355" initialY="510" initialScale="100" />
```

mapInitial node indicates the initial view of your system:

initialX: Start-up horizontal location.

initialY: Start-up vertical location.

initialScale: Start-up scale.

```
<mapSelect color="0xFFF000" divider="," />
```

mapSelect node sets the behavior when using AreaSelector in 'Selection' mode.

color: Color to fill the areas as they are selected.
divider: Separation character when returning the IDs of the selected areas as a single string.

```
<mapConfig backLabel="Back to main map" showLabel="true" areasAlpha="40"
popupCloseWhenClick="true" />
```

mapConfig node sets up several general behavior aspects:

backLabel: Tooltip to show when hovering over the back button.

showLabel: Shows or hides the current area label on the upper right corner.

areasAlpha: When you zoom in an area, the opacity of the parent map.

popupCloseWhenClick: Closes popups when user clicks anywhere in the map.

```
<mapZoom zoom="true" zoomMouse="true" pan="true" />
```

mapZoom node sets up several zooming aspects:

zoom: Displays the zoom tool buttons.

zoomMouse: Enables or disables zooming in and out when user rolls mouse wheel.

pan: Enables or disables the pan navigation tool.

```
<levels poisSendParams="true" areasSendParams="true" randomLoad="false">
```

levels node sets up several aspects of zoom levels (nation, state, county, etc.)

poisSendParams: Send ID, label and category as parameters when launching an action, like opening a different URL in the browser window.

areasSendParams: Send ID as a parameter when launching an action, like opening a different URL in the browser window.

randomLoad: Add a random parameter at the end of the URL, to prevent the Flash player from caching the output of dynamic pages.

```
<level id="1" showPOIText="false" showAreaTooltip="true"
    showAreaSelectedTooltip="true"  showPOITooltip="true"
    poisLoad="true" poisShow="true" poisDelay="0" filterId=""
    poisRefreshOnViewChange="true"
    polylinesLoad="true" polylinesShow="true" />
```

where level child node sets up several aspects of each of the zoom levels (nation, state, county, etc.)

id: Level's ID.

showPOIText (true|false): Shows or hides area texts on their centroids on this level. By default, it is false.

showAreaTooltip (true|false): Displays a tooltip with area's name when you hover over one of them on this level. By default, it is true.

showAreaSelectedTooltip (true|false): Displays a tooltip with area's name when you have zoomed in it. By default, it is true.

showPOITooltip (true|false): Displays a tooltip with POI's name when you hover over one of them on this level. By default, it is true.

poisLoad (true|false): Loads a new set of POIs when you enter this zoom level. New level, current area and parent area will be sent as parameters in the POI's URL, so you can load just the relevant POIs. By default, it is true.

poisShow (true|false): Shows or hides POIs on this level. By default, it is false.

poisDelay: Sets a delay (in seconds) between POIs while plotting them over the map. By default, it is "0" (no delay).

filterId: Use a graphic filter for the areas on this level. Filters are defined in fmASEngine.xml, under categories node.

poisRefreshOnViewChange (true|false): Reload POIs when you change the view by zooming in, out or panning.

polylinesLoad (true|false): Load a new set of polylines when you enter this zoom level. The new level, the current area and the parent area will be sent as parameters in the polylines URL, so you can load just the relevant polylines. By default, it is true.

polylinesShow (true|false): Show or hide polylines at this level. By default, it is true.

2.2. Theme XML Feed

The theme XML feed describes the general make-up of the theme, namely: the map itself, areas, POIs and polylines to be loaded within AreaSelector.

Normally, the theme XML feed will consist of a plain XML file, but it can also be created on the fly by a dynamic URL (php, asp, etc.).

This is a what a theme XML feed looks like:

```
<theme id="theme1">
        <map file="fm-us.swf" borderColor="0x666666" />
        <areas xmlAreas="areas.xml" xmlCategories="area_categories.xml" />
        <pois xmlPOIs="pois.xml" xmlCategories="poi_categories.xml" />
</theme>
```

It includes the following attributes:

id: Theme's ID.

And the following nodes:

```
<map file="fm-us.swf" borderColor="0x666666" showPOIText="true" />
```

where map node describes the initial map to load and some of its properties:

file: Initial map's URL, relative to the page that instances AreaSelector's Flash object.

borderColor: Area bounderies' line color.

showPOIText: Show or hide areas' texts on their corresponding centroids.

```
<areas xmlAreas="areas.xml" xmlCategories="area_categories.xml" />
```

where areas node defines where to load areas and area categories from:

xmlAreas: URL to load initial areas from.

xmlCategories: URL to load initial area categories from.

```
<pois xmlPOIs="pois.xml" xmlCategories="poi_categories.xml" />
```

where pois node defines where to load POIs and POI categories from:

xmlPOIs: URL to load initial POIs from.

xmlCategories: URL to load initial POI categories from.

```
<polylines xmlPolylines="lines.xml" xmlCategories="lines_categories.xml" />
```

where polylines node defines where to load polylines from:

xmlPolylines: URL to load polylines from.

xmlCategories: URL to load polyline categories from.

2.3. Area Categories XML Feed

The area categories XML feed describes all the categories in use by the areas on the map. You will normally want to group several areas in one category, when you need them to look and behave in the same way. For example, you may want to create a category 'blue' to fill a group of areas with that color.

The area categories XML feed may be a plain XML file or can be created on the fly by a dynamic URL (php, asp, etc.).

This is what an area categories XML feed looks like:

```
<areaCategories>
        <category id="state" zoom="true" enabled="true" tooltip="popup.swf">
                <color up="0xFF0000" over="FFCC66" down="FFCC66" />
                <action event="onRelease" target="_blank" url="info.php" />
        </category>
</areaCategories>
```

Area categories XML feed is built up of **category** nodes.

2.3.1. category Node

Area categories XML feed is built up of category nodes:

```
<category id="state" zoom="true" enabled="true" tooltip="popup.swf">
        <color up="0xFF0000" over="FFCC66" down="FFCC66" />
        <action event="onRelease" target="_blank" url="info.php" />
</category>
```

category node describes each of the area categories. The following attribute is required:

id: Area category's ID.

The following attributes are optional:

zoom (true|false): Zoom or not over an area when you click on it.

enabled (true|false): Areas will trigger onRollover, onRollout and onRelease events (true) or not (false). By default, it is true.

tooltip (true|false|popup): When you hover over an area, make AreaSelector display a standard tooltip (true), display nothing (false) or display a custom popup (specify Flash movieclip's URL).

poitextField: Area's attribute to display as a text in the middle of the area. It refers to the attribute of the respective area.xml file; e.g. poitextField='state' will display the values of the attribute named 'state' in the respective area.xml file.

poitextCategory: The text category (style) for the text in the middle of the area. It refers to the text categories (nodes) in the file fmASEngine.xml; e.g. poitextCategory='state' will display the text in the font style specified in the text node, with the id='state' in the fmASEngine.xml.

poitextScaleMin: If you will display texts in the middle of areas, set the minimum scale to a value where the texts will be visible. By default, it is 0 (no minimum scale restriction).

poitextScaleMax: If you will display texts in the middle of areas, set the maximum scale to a value where the texts will be visible. By default, it is the maximum scale (no maximum scale restriction).

category node includes the following child nodes:

2.3.1.1 color Child Node

```
<color up="0xFF0000" over="FFCC66" down="FFCC66" />
```

color child node sets area colors. The following attributes are required:

up: Area's regular color.

over: Color when you hover over the area.

down: Color when you are clicking over the area.

The following attributes are optional:

alpha: Regular color opacity (percentage).

filterId: Set a graphic filter to the areas of this category. It refers to the attribute 'filterId' in the nodes levels of the file fmASEngine.xml. The available flash filters are listed as filter nodes in the same file.

gradient: Set a color gradient, from minimum to maximum. Each area will be colored according to its colorAlpha attribute's value. The two colors are separated with a comma. For Example: **gradient="ffffff,880000"**

2.3.1.2 action Child Node

```
<action event="onRelease" target="_blank" url="default.html" />
```

action child node describes an action to trigger when you hover over an area, move out of it or click on it. The same event may trigger several actions, not just one.

The following attributes are required:

event (onRelease|onRollover|onRollout): The event when the action will be triggered. Valid values are:

onRelease: The user has clicked on the area.

onRollover: The user has hovered over the area.

onRollout: The user has moved out of the area.

target (_mc|_js|_as|_blank|_self|_top): The type of action that will be launched. Valid values are:

_mc: Pop up a Flash movieclip (.swf) over the map. This is a popular way to display extended information about a certain area when you click on it.

_js: Call a JavaScript function.

_as: Call an ActionScript function named fmAreaToOutside in the container (the Flash application that contains AreaSelector; only for cases when the Flashmaps AreaSelector is embedded within a parent Flash application rather than directly in the HTML).

_blank: Open a web page on a new browser window.

_self: Open a web page on the current browser window or frame.

_top: If AreaSelector is displayed inside a frame, open a web page on current browser window.

url: The specific action to launch. Depending on the target:

_mc: Flash movieclip's URL. All the area's attributes are sent to the movieclip, so you will be able to display it on the popup or query for additional information using area's ID. The URL can either be: relative to the file fmASToolbar.swf (normally in the folder fmASMap); absolute in the same subdomain, starting with /; or on a different subdomain. starting with http://.

_js: JavaScript function to call. Event, ID and label are sent automatically as string parameters to the JavaScript function, so you can display them or load any of additional information by using AJAX technics. The function is referenced just by the function name (no brackets):

```
<action event='onRollover' target='_JS' url='myFunction' />
```

This will call the function myFunction() on the main page. It will automatically receive the above mentioned string parameter, e.g.:

```
function myFunction(string1, string2, string3) {
        alert('event: ' + string1 +' areaID: ' + string2 +' label: ' + string3);
       }
```

_as: A string to send one or more parameters to fmAreaToOutside ActionScript function. All the following parameters are sent, namely: event, area's ID, URL (this attribute) and area's label.

_blank, **_self** or **_top**: The URL of the web page to display. AreaSelector adds an **id** parameter to send over area's ID, so the page can display information relevant to that specific area, if required.

If, for any reason, you do not want to send the ID and the rest of parameters, you can set poisSendParams attribute to false inside fmASEngine.xml.

You can also make AreaSelector send certain attributes as parameters with targets **_blank**, **_self** and **_top**, by specifying the attribute name inside brackets (referring to the attributes of the respective areas.xml).

For example, if you click on Utah:

```
url="info.php?itemlabel=[label]"
```

this will make AreaSelector call:

```
info.php?itemlabel=Utah
```

The URL can either be: relative (to fmASToolbar.swf or fmASMap, respectively); absolute, inside the same subdomain, if starting with /, and on a different subdomain, if starting with http://.

2.4. Areas XML Feed

Areas XML feed describes the set of areas to be displayed inside the map. Each area will behave as an independent object, which may be colored or trigger actions when you hover over or click on it.

This is what an areas XML feed looks like:

```
<areas>
    <area id="us_ca" category="state" label="California" />
    <area id="us_fl" category="state" label="Florida" />
    ...
</areas>
```

Areas XML feed is built up of several area nodes:

2.4.1. area Node

```
<area id="us_ca" category="state" label="California" />
```

area node describes each of the areas. The following attributes are required:

id: Area's ID.

category: Area's category. Normally all the areas of the same category look and behave the same.

label: Area's label. The label is usually displayed as a tooltip when you hove over an area.

The following attributes are optional:

child: New theme to load when entering this area. It is necessary to load new themes in maps with more than 2 map levels as, depending upon the map level, new map files have to be loaded. E.g. the Flashmaps AreaSelector USA & States with only two levels (USA view showing the states inside and one-state view when clicking on it) contains only one map file and does not need to load child themes. Instead, when you click on a state, the Flashmaps AreaSelector USA & States & Counties loads a new map file with the counties inside (usually not displayed at US view). Therefore a child theme has to be loaded if the map zooms into a state which configures the new cartography.

zoom (true|false): The area will be zoomed in when you click on it (true) or not (false). By default, it is true.

enabled (true|false): The area will trigger onRollover, onRollout and onRelease events (true) or not (false). By default, it is true.

tooltip (true|false|popup): When you hover over an area, make AreaSelector display a standard

tooltip (true), display nothing (false) or display a custom popup (specify Flash movieclip's URL).

colorAlpha: Set fill color opacity (percentage). If a gradient color was set for this category, this value indicates which color to use within the minimum and maximum colors in the gradient (0 means the first color and 100 the second one).

poitextField: Area's attribute to display as a text in the middle of the area.

poitextCategory: The text category (style) for the text in the middle of the area.

poitextLat, poitextLon: Instead of letting AreaSelector place the text in the middle of the area , set a specific location for the text. Remember that coordinates are always decimal (GGG.GGGG), never sexagesimal or base-sixty (GGG.MMSS).

poitextScaleMin: If you display texts in the middle of areas, set the minimum scale where the text will be visible. By default, it is 0 (no minimum restriction).

poitextScaleMax: If you display texts in the middle of areas, set the maximum scale where the text will be visible. By default, it is the maximum scale (no restriction).

As explained earlier, area colors and events are normally set up for a full category, in the area categories XML feed. However, if you need to assign specific colors and/or actions to just one area, you can set them up inside the *area* node in the same way you do in the *category* node. For example:

```
<area id="us_ca" category="state" label="California" >
        <color up="0xFF0000" over="FFCC66" down="FFCC66" />
        <action event="onRelease" target="_blank" url="infoCA.html" />
</area>
```

Besides the standard nodes and attributes described in this section, you can include in the *area* node any additional information you want to use in the popups or send to a URL as parameters when clicking the area, for example:

```
<area id="us_ca" category="state" label="California" branches="3" sales="$4.3M"/>
```

2.5. POI Categories XML Feed

The POI categories XML feed describes all the categories in use by the POIs (Points of Interest) plotted over the map. You normally want to group several POIs in one category when you need them to look and behave the same. For example, you may want to create a category 'branches' to display a certain icon, and 'distributors' to display a different one.

The POI categories XML feed may be a plain XML file, or it can be created on the fly by a dynamic URL (php, asp, etc.).

This is what a POI categories XML feed looks like:

```
<POICategories>
    <category id="city" enabled="true" iconUrl="city" iconEmbedded="true"
        tooltip="popup.swf" >
        <action event="onRelease" target="_blank" url="default.html" />
    </category>
</POICategories>
```

POI categories XML feed is built up of **category** nodes:

2.5.1. category Node

```
<category id="city" enabled="true" iconUrl="city" iconEmbedded="true" tooltip="popup.swf" >
    <action event="onRelease" target="_blank" url="default.html" />
</category>
```

category node describes each of the POI categories. The following attributes are required:

id: POI category's ID.

iconEmbedded (true|false): Indicates if this category uses an icon in the internal library (true) or an external one (false). Please read section *2.3.3.POIs (Points of Interest)* for extended information about POIs.

iconUrl: If iconEmbedded is true, the icon's ID within the library. If iconEmbedded is false, the icon's URL. Please read section *2.3.3.POIs (Points of Interest)* for extended information about POIs.

iconWidth: Only required if iconEmbedded is false, the icon's width in pixels.

iconHeight: Only required if iconEmbedded is false, the icon's height in pixels.

The following attributes are optional:

enabled (true|false): POIs will trigger onRollover, onRollout and onRelease events (true) or not (false). By default, it is true.

tooltip (true|false|popup): When you hover over a POI, make AreaSelector display a standard

tooltip (true), display nothing (false) or display a custom popup (specify Flash movieclip's URL).

textScaleMin: If the POI includes a text to display (iconLabel attribute), the minimum scale for the texts to be visible. By default, it is 0 (no restriction).

textScaleMax: If the POI includes a text to display (iconLabel attribute), the maximum scale for the texts to be visible. By default, it is the maximum scale (no restriction).

textScalePOI (false|true): Applies text scale restrictions to POI icons as well as POI texts.

rollover: When you hover over a POI, it displays another icon from the library over the POI to highlight it. For example, you may want to display a radiant circle around the POI.

menu (true|false): Enables (true) or disables (false) the right-click context menu.

category node includes the following child nodes:

2.5.1.1 action Child Node

```
<action event="onRelease" target="_blank" url="/info/poiinfo.php" />
```

action child node describes an action to trigger when you hover over a POI, move out of it or click on it. Check out *action Node* in section *3.3. Area Categories XML Feed.*

2.5.1.2 menu Child Node

```
<menu label="About Us..." action="_self,www.bestcompany.com" />
```

menu child node adds a line to the context menu when you right-click on the POI. Remember you need to set to *true* the attribute *menu* inside the POI category node.

The following attributes are required:

label: The text that will appear in the context menu.

action: When the user clicks on the new context menu option, AreaSelector invokes a function called *fmFromASPOI* in the container (the Flash movieclip that contains AreaSelector in case that the AreaSelector is embedded in a parent flash environment). This action attribute is send to the function as a parameter, so from there you can do whatever you need.

2.6. POIs XML Feed

POIs XML feed describes the set of POIs (Points of Interest) to be plotted over the map. Each POI will behave as an independent object, displaying an icon and triggering certain actions when you hover over or click on it.

This is what a POIs XML feed looks like:

```
<POIs>
<poi id="1" category="city" label="New York" lat="43.43" lon="-73.1" />
...
</POIs>
```

POIs node may include an attribute called *focusOnCategory* that makes AreaSelector focus over the POIs. For example:

```
<POIs focusOnCategory="*" >
```

focusOnCategory (*|category): After loading the set of POIs, make AreaSelector focus on the full set (*) or on the ones pertaining to a certain category (specify the name of the category).

POIs XML feed consists of several *poi* nodes:

2.6.1. poi Node

```
<poi id="1" category="city" label="New York" lat="43.43" lon="-73.1" />
```

poi node describes each of the POIs.

The following attributes are required:

id: POI's ID.

category: POI's category. Normally all the POIs of the same category look and behave the same.

label: POI's label. The label is usually displayed as a tooltip when you hove over a POI.

lat: POI's latitude in decimal degrees.

lon: POI's longitude in decimal degrees.

The following attributes are optional:

enabled (true|false): The POI will trigger onRollover, onRollout and onRelease events (true) or not (false). By default, it is true.

tooltip (true|false|popup): When you hover over a POI, make AreaSelector display a standard tooltip (true), display nothing (false) or display a custom popup (specify Flash movieclip's URL).

textScaleMin: If you display texts in the middle of areas, set the minimum scale for the texts to be visible. By default, it is 0 (no restriction).

textScaleMax: If you will texts in the middle of areas, set the maximum scale where the texts will be visible. By default, it is the maximum scale (no restriction).

textScalePOI (false|true): Apply text scale restrictions, to POI icons as well as POI texts,.

rollover: When you hover over a POI, display another icon from the library over the POI to highlight it. For example, you may want to display a radiant circle around the POI.

As explained earlier, POI icons and events are normally set up for a full category, in the POI categories XML feed. But, if you need to assign specific icon and/or actions to one single POI, you can set them up inside the *poi* node just as you would in the *category* node. For example:

```
<poi id="1" category="city" label="New York" lat="43.43" lon="-73.1">
    <action event="onRelease" target="_blank" url="/info/newyork.html" />
</poi>
```

Besides the standard nodes and attributes described in this section, you can include in the *poi* node any additional information you want to use in the popups or send to a URL as parameters when clicking the POI. For example:

```
<poi id="1" category="city" label="New York" lat="43.43" lon="-73.1"
    employees="34" sales="$3.6M" />
```

2.7. Polyline Categories XML Feed

The polyline categories XML feed describes all the categories in use by the polylines you can draw over the map. You will normally want to group several polylines in one category when you need them to look and behave in the same way. For example, you may want to create a category 'routes' to the corresponding polylines, with a certain color and width.

The polyline categories XML feed may be a plain XML file, or it can be created on the fly by a dynamic URL (php, asp, etc.).

This is what a polyline categories XML feed look like:

```
<PolylineCategories>
    <category id="route" enabled="true" color="0xFF0000" thickness="2"
        tooltip="routepopup.swf" >
        <action event="onRelease" target="_blank" url="default.html" />
    </category>
</PolylineCategories>
```

2.7.1. category Node

Polyline categories XML feed consists of category nodes:

```
<category id="route" enabled="true" color="0xFF0000" thickness="2"
    tooltip="routepopup.swf" >
    <action event="onRelease" target="_blank" url="routeinfo.php" />
</category>
```

category node describes each of the polyline categories. The following attributes are required:

id: Polyline category's ID.

color: Polyline's color.

colorAlpha: Polyline's opacity (percentage). By default, it is 100 (100%, fully opaque).

thickness: Polyline's thickness in pixels. By default, it is 1.

The following attributes are optional:

enabled (true|false): Polylines will trigger onRollover, onRollout and onRelease events (true) or not (false). By default, true.

tooltip (true|false|popup): When you hover over a polyline, make AreaSelector display a standard tooltip (true), display nothing (false) or display a custom popup (specify Flash movieclip's URL).

arrowSize: You may display an arrow (or any other symbol) in the middle of the polyline to indicate from where to where it goes. This attribute defines the size of the symbol. The symbol itself is defined in the icon library (fmASIconLibrary.fla): under the symbols folder named *areaselector*, the symbol named *arrow*. In the library, the arrow needs to be pointing to the left.

category node includes the following child nodes:

2.7.1.1 action Child Node

```
<action event="onRelease" target="_blank" url="/lineinfo.php" />
```

action child node describes an action to trigger when you hover over a polyline, move out of it or click on it. Check out *action Node* in section *3.3. Area Categories XML Feed.*

2.8. Polylines XML Feed

Polylines XML feed describes the set of polylines to be displayed over the map. Each polyline will behave as an independent object, which may be colored or trigger actions when you hover over or click on it.

This is how a polylines XML feed looks like:

```
<polylines>
    <polyline id="1" category="line" label="Line 1" >
        <vertex lat="43.12" lon="-73.12" />
        <vertex lat="44.12" lon="-73.12" />
        ...
    </polyline>
    ...
</polylines>
```

Polylines XML feed consists of several **polyline** nodes:

2.8.1. polyline Node

```
<polyline id="1" category="line" label="Line 1" >
    <vertex lat="43.12" lon="-73.12" />
    <vertex lat="44.12" lon="-73.12" />
    ...
</polyline>
```

polyline node describes each of the polylines.

The following attributes are required:

id: Polyline's ID.

category: Polyline's category. Normally all the polylines of a same category look and behave the same.

label: Polyline's label. The label is usually displayed as a tooltip when you hover over a polyline.

The following attributes are optional:

color: Polyline's color.

colorAlpha: Polyline's opacity (percentage). By default, it is 100 (100%, fully opaque).

thickness: Polyline's thickness in pixels. By default, it is 1.

enabled (true|false): Polylines will trigger onRollover, onRollout and onRelease events (true) or not (false). By default, it is true.

tooltip (true|false|popup): When you hover over a polyline, make AreaSelector display a standard tooltip (true), display nothing (false) or display a custom popup (specify Flash movieclip's URL).

arrowSize: You may display an arrow (or any other symbol) in the middle of the polyline to indicate from where to where it goes. This attribute defines the size of the symbol. The symbol itself is defined in the icon library (fmASIconLibrary.fla): under the symbols folder named *areaselector*, the symbol named *arrow*. In the library, the arrow has to be pointing to the left.

curve (true|false): Display the polyline as a curve (true) or as it is (false).

curveAngle: If *curve* is true, this attribute indicates the curvature (percentage).

curveHighest: If *curve* is true, this attribute indicates where in the line the curve is highest (percentage).

polyline node has to include, at least, two *vertex* child nodes:

2.8.1.1 vertex Child Node

```
<vertex lat="43.12" lon="-73.12" />
```

vertex child node describes each of the points of the polyline. Please note that the polyline will be created through the vertex in the order that you declare them.

The following attributes are required:

lat: Vertex's latitude.

lon: Vertex's longitude.

As explained earlier, polyline events are normally set up for a full category, in the polyline categories XML feed. However, if you need to assign specific actions to just one polyline, you can set them up inside the *polyline* node in the same way as in the *category* node. For example:

```
<polyline id="1" category="line" label="Line 1" >
        <action event="onRelease" target="_blank" url="/info/line1.html" />
</polyline>
```

3. API reference

AreaSelector can be embedded into two different environments: HTML code or a parent Flash movie clip, a Flash container. Find in this section a reference to the API (Application Programming Interface) AreaSelector offers for both environments.

3.1. JavaScript API reference

If you are embedding AreaSelector into HTML, PHP, ASP, ASP.NET, ColdFusion or any other technology that returns HTML code, AreaSelector offers a powerful JavaScript API to let you communicate with the map.

3.1.1. fmThemeLoad

Load a theme.

Function
fmThemeLoad(theme_str)

Parameter
theme_str (string) XML URL containing the theme to load. Can be an XML file or a dynamic page.

Example (load USA theme)
```
<a href="JavaScript:fmThemeLoad('USA.xml');">Load USA</a>
```

3.1.2. fmThemeReloadAreas

Reload a collection of areas.

Function
fmThemeReloadAreas(areas_xml)

Parameter
areas_xml (string) XML URL containing the areas to reload. Can be an XML-file or a dynamic page.

Example (reload the areas of USA)
```
<a href="JavaScript:fmThemeReloadAreas('states.asp');"> Reload states</a>
```

3.1.3. fmThemeReloadPOIs

Reload a collection of POIs.

Function
fmThemeReloadPOIs(pois_xml)

Parameter
pois_xml (string) XML URL containing the POIs to reload. Can be an XML-file or a

dynamic page.

Example (reload a set of POIs)

```
<a href="JavaScript:fmThemeReloadPOIs('pois.asp');"> Reload POIs</a>
```

3.1.4. fmInitialView

Display the initial view.

Function

fmInitialView()

Example (display the initial view)

```
<a href="JavaScript:fmInitialView();">Initial view</a>
```

3.1.5. fmMapBackLevel

Go back one level. For example, if displaying one state with its counties, get back to country map.

Function

fmMapBackLevel()

Example (get back one level)

```
<a href="JavaScript:fmMapBackLevel();">Back one level</a>
```

3.1.6. fmAreaCenter

Focus on a certain area in the current theme. Depending on the current level, AreaSelector focuses with or without the need to zoom.

Function

fmAreaCenter(area_str)

Parameter

area_str (string) ID of the area to focus on.

Example (Focus on California)

```
<a href="JavaScript:fmAreaCenter('us_ca');">California</a>
```

3.1.7. fmAreaZoomIn

Dynamically zoom in a certain area running through higher levels.

Function

fmThemeReloadAreas(areas_array)

Parameter

areas_array (array) Array of area IDs to run through while zooming in and center.

Example (Zoom on Northamerica, USA and, finally, California)

```
<a href="JavaScript:fmAreaZoomIn('NA,US,us_ca');">California</a>
```

3.1.8. fmAreaCenterLatLon

Focus on a point, inside a certain area, at a specific scale.

Function
fmAreaCenterLatLon(area_str, lat, lon, scale)

Parameter

area_str (string)	ID of the area to focus on.
lat (numeric)	Latitude in decimal degrees.
lon (numeric)	Longitude in decimal degrees.
scale (numeric)	Scale percentage. The higher the value, the closer the view.

Example (focus on a point inside NY)

```
<a href="JavaScript:fmAreaCenterLatLon('us_ny', 40.742272, -73.987907, 2000);">New
York</a>
```

Comentario [T1]: ¿El area id se utiliza en esta función? A mí no me hace ninguna diferencia si utilizo 'NY' o 'us_ny' …

3.1.9. fmAreaEnabled

Enable or disable specific areas. When areas are disabled, these can not be selected.

Function
fmAreaEnabled(area_str, enabled_str)

Parameter

area_str (string)	ID of the area to enable/disable.
enabled_str (string)	Enable or disable area (true or false)

Example (disable California state)

```
<a href="JavaScript:fmAreaEnabled('us_ca', 'false');">Disable California</a>
```

3.1.10. fmAreaColor

Change the colors of a specific area.

Function
fmAreaColor(area_str, colorNormal, colorOver, colorPress, colorText)

Parameter

area_str (string)	ID of the area to change colors.
colorNormal (string)	Area color in normal state (mouse pointer outside the area).
colorOver (string)	Area color when hovering over it.
colorPress (string)	Area color when clicking it.
colorText (string)	Text color.

Example (change Oregon's colors)

```
<a href="JavaScript:fmAreaColor('us_or', '0x4a6e93', '0xFF0000', '0xFFF000', '0x000000');">
        Color Oregon</a>
```

3.1.11. fmMapModeZoom

Set AreaSelector to zoom mode. When a user selects an area, the system zooms in on it. This is the default mode; however, you will normally want to get back to it after using the Select Mode.

Function
fmMapModeZoom()

Example (enter zoom mode)
```
<a href="JavaScript:fmMapModeZoom();">Mode Zoom</a>
```

3.1.12. fmMapModeSelect

Set AreaSelector to selection mode. When a user selects an area, it is either selected or deselected.

Function
fmMapModeSelect()

Example (enter select mode)
```
<a href="JavaScript:fmMapModeSelect();">Mode Select</a>
```

3.1.13. fmMapModeExportListAreas

Get a list of selected areas, sorted by levels.

Function
fmMapModeExportListAreas():String

Return
areas_str (string) List of selected areas.

Example (get the list of selected areas)
```
<a href="JavaScript:alert(fmMapModeExportListAreas());">Get Selected Areas</a>
```

3.1.14. fmMapModeCleanAreas

Clear selected areas.

Function
fmMapModeCleanAreas()

Example (clear selected areas)
```
<a href="JavaScript:fmMapModeCleanAreas();">Clear</a>
```

3.1.15. fmPOIsShowCategory

Show POIs pertaining to one or every category.

Function
fmPOIsShowCategory(category_str)

Parameter

category_str (string)	ID of the category to show, or "*" for every one.

Example (show all categories)
```
<a href="JavaScript:fmPOIsShowCategory('*');">Show all</a>
```

3.1.16. fmPOIsHideCategory

Hide POIs pertaining to one or every category.

Function
fmPOIsHideCategory(category_str)

Parameter

category_str (string)	ID of the category to hide, or "*" for every one.

Example (hide hotels)
```
<a href="JavaScript:fmPOIsShowCategory('hotel');">Hide Hotels</a>
```

3.1.17. fmPOIAddEvent

Execute a concrete event.

Function
fmPOIAddEvent(event_str, target_str, url_str, id_str)

Parameters

event_str (string)	Event when action will be launched. Can have the following values: **onRelease**: invoked when clicking over an area. **onRollOver**: invoked when hovering over an area. **onRollOut**: invoked when the mouse leaves the area.
target_str (string)	Type of action to launch. Can have the following values: **_MC**: pop up a Flash movieclip. **_JS**: call a JavaScript function. **_AS**: call an AreaSelector internal function. **_blank**, **_self**, **_top**: open a web page.
url_str (string)	Action to launch. Its value depends on the type of action. **_MC**: path to the movieclip to load.

_JS: name of the JavaScript function to call. The event and area's ID are sent as parameters.
_AS: Action name to be launched.
_blank, **_self**, **_top**: web page's URL. The ID is sent as poi_id parameter. This URL can either be: relative to the file fmASToolbar.swf (normally in the folder fmASMap); absolute in the same subdomain, starting with /, or on a different subdomain, starting with http://.

Id_str (string) POI ID.

Example (open a popup)

```
<a href="JavaScript:fmPOIAddEvent('onRelease','_MC', 'popups/team_info.swf',
'12');">Show popup</a>
```

3.1.18. fmPOIRollOver

Execute the rollover event of a POI

Function
fmPOIRollOver(id_str)

Parameter
Id_str (string) POI ID.

Example (launch rollover event on a POI)

```
<a href="JavaScript:fmPOIRollOver('12');">Rollover event on POI 12</a>
```

3.1.19. fmPOIRollOut

Execute the rollout event of a POI

Function
fmPOIRollOut(id_str)

Parameter
Id_str (string) POI ID.

Example (launch rollout event on a POI)

```
<a href="JavaScript:fmPOIRollOut('12');">Rollout event on POI 12</a>
```

Comentario [T2]: No me funcionan las dos funciones (fmPOIRollOver & RollOut) probando en versión 003.003.003 y 003.001.001

3.1.20. fmShowAlert

Show an alert window with a custom message over AreaSelector.

Function
fmShowAlert(title_str, text_str)

Parameters
title_str (string) Title of the alert window.
text_str (string) Text inside the alert window.

Example (show alert window)
```
<a href="JavaScript:fmShowAlert('Error','Please select a state');">Show Alert</a>
```

3.1.21. fmHideAlert

Hide the alert window.

Function
fmHideAlert()

Example (hide alert window)
```
<a href="JavaScript:fmHideAlert();">Hide Alert</a>
```

3.1.22. fmShowCrossHair

Focus on a certain point and display a cross hair over it.

Function
fmShowCrossHair()

Example (focus and show crosshair)
```
<a href="JavaScript:fmShowCrossHair('40.35', '-90.56', '2000');">Focus and show
crosshair</a>
```

Comentario [T3]: El crosshair pone en los coordenadas. Pero el scale utiliza para escalar el icono del crosshair y no el mapa . Probando en versión 003.003.003 y 003.001.001

3.1.23. fmHideCrossHair

Hide the cross hair.

Function
fmHideCrossHair()

Example (hide crosshair)
```
<a href="JavaScript:fmHideCrossHair();">Hide crosshair</a>
```

3.1.24. fmShowPOIText

Show texts in the middle of areas.

Function
fmShowPOITexts()

Example (show texts in the middle of the areas)
```
<a href="JavaScript:fmShowPOIText();">Show texts in areas</a>
```

3.1.25. fmHidePOIText

Hide texts in the middle of areas.

Function
fmHidePOITexts()

Example (hide texts in the middle of the areas)
```
<a href="JavaScript:fmHidePOIText();">Hide texts in areas</a>
```

3.1.26. fmPrint

Print the current map view.

Function
fmPrint()

Example (print the current view)
```
<a href="JavaScript:fmPrint();">Print Map</a>
```

3.2. Flash API Reference

3.2.1. fmThemeLoad

Load a theme.

Function
fmThemeLoad(theme_str, area_str)

Parameter

theme_str (string) XML URL containing the theme to load. Can be an XML file or a dynamic page.

area_str (string) ID of the initial area to center.

Example (load USA states theme)
```
container_mc.fmThemeLoad("areas/fmUSA.xml");
```

3.2.2. fmThemeReloadAreas

Reload a collection of areas.

Function
fmThemeReloadAreas(areas_str)

Parameter

Areas_str (string) XML URL containing the areas to load. Can be an XML file or a dynamic

page.

Example (reload areas from **fmStates.php**)

```
container_mc.fmThemeReloadAreas("areas/fmStates.php");
```

3.2.3. fmThemeReloadPOIs

Reload a collection of POIs.

Function

```
fmThemeReloadPOIs(pois_str)
```

Parameter

pois_str (string) XML URL containing the POIs to load. Can be an XML file or a dynamic page.

Example (reload POIs from **pois.php**)

```
container_mc.fmThemeReloadPOIs("pois/pois.php");
```

3.2.4. fmThemeReloadPolylines

Reload a collection of polylines.

Function

```
fmThemeReloadPolylines(polylines_str)
```

Parameter

polyliness_str (string) XML URL containing the polylines to load. Can be an XML file or a dynamic page.

Example (reload polylines from **polylines.php**)

```
container_mc.fmThemeReloadPolylines("polylines/polylines.php");
```

3.2.5. fmInitialView

Display the initial view.

Function

```
fmInitialView()
```

Example (display the initial view)

```
container_mc.fmInitialView();
```

3.2.6. fmMapBackLevel

Go back one level. For example, if displaying a state with its counties, get back to country map.

Function

```
fmMapBackLevel()
```

Example (go back one level)

```
container_mc.fmMapBackLevel();
```

3.2.7. fmAreaCenter

Focus on a certain area in the current theme. Depending on the current view level, AreaSelector focuses with or without the need to zoom.

Function
fmAreaCenter(area_str, filter_str, same_level)

Parameter

area_str (string)	ID of the area to focus on.
filter_str (string)	Apply a filter to the POIs to load
same_level (bool)	If you want to center in an area on this level, or zoom in an area of lowest level.

Example (focus on California)

```
container_mc.fmAreaCenter("us_ca", "", false);
```

3.2.8. fmAreaBackAndCenter

Zoom back one level and focus on a certain area. Depending on the current view level, AreaSelector focuses with or without the need to zoom.

Function
fmAreaBackAndCenter(area_str)

Parameter

area_str (string)	ID of the area to focus on.

Example (go back one level and focus on California)

```
container_mc.fmAreaBackAndCenter("us_ca");
```

3.2.9. fmAreaZoomIn

Dynamically zoom on a certain area running through higher levels.

Function
fmAreaZoomIn(areas_array)
Parameter

areas_array (array)	Array of area IDs to run through while zooming in and center.

Example (from a world map, focus on Northamerica, USA and California)

```
var areas_array = new Array("NA", "US", "us_ca");
container_mc.fmAreaZoomIn(areas_array);
```

3.2.10. fmAreaCenterLatLon

Focus on a point inside a certain area at a specific scale.

Function
fmAreaCenterLatLon(area_str, lat, lon, scale, load_pois, map_back)

Parameters

area_str (string)	ID of the area to focus on.
lat (number)	Latitude in decimal degrees.
lon (number)	Longitude in decimal degrees.
scale (number)	Scale percentage. The higher the value, the closer the view.
load_pois (bool)	Whether one wants to load POIs when centering in the new area or not. By default, it is true.
map_back (bool)	Whether one wants to go back to lowest level and zoom in. By default, it is true.

Example (zoom in an area and focus on an specific point)

```
container_mc.fmAreaCenterLatLon("us_ny", 40.7422, -73.987, 2000, true, true);
```

3.2.11. fmAreaEnabled

Enable or disable a specific area. When disabled, areas are not selectable.

Function

```
fmAreaEnabled(area_str, bEnabled)
```

Parameters

Area_str (string)	ID of the area to enable/disable.
bEnabled (boolean)	Enable or disable area (true or false)

Example (disable California)

```
container_mc.fmAreaEnabled("us_ca", false);
```

3.2.12. fmAreaColor

Change the colors of a specific area.

Function

```
fmAreaColor(area_str, colorUp, colorOver, colorDown, colorText)
```

Parameters

area_str (string)	ID of the area.
colorUp (string)	Area color in normal state (mouse pointer outside the area).
colorOver (string)	Area color when hovering over it.
colorDown (string)	Area color when clicking it.
colorText (string)	Text color.

Example (set new colors for Oregon)

```
container_mc.fmAreaColor("us_or", "0x4a6e93", "0xFF0000", "0xFFF000", "0x000000");
```

3.2.13. fmObjectShow

Show an object (* for all).

Function

```
fmObjectShow(obj_str)
```

Parameter
obj_str (string) ID of the object.

Example (show all objects)
```
container_mc.fmObjectShow("*");
```

3.2.14. fmObjectHide

Hide an object (* for all).

Function
```
fmObjectHide(obj_str)
```

Parameter
obj_str (string) ID of the object.

Example (hide all objects)
```
container_mc.fmObjectHide("*");
```

3.2.15. fmMapModeActivate

Set zoom or selection mode.

Function
```
fmMapModeActivate(mode_str)
```

Parameter
mode_str (string) Mode to set. The valid options are:
zoom: activate zoom mode.
select: activate select mode.

Example (activate zoom select mode)
```
container_mc.fmMapModeActivate("select");
```

3.2.16. fmMapModeExport

Get the list of selected areas sorted by level.

Function
```
fmMapModeExport():string
```

Return
(string) List of selected areas.

Example (get the selected areas)
```
var areas_str = container_mc.fmMapModeExport();
```

3.2.17. fmMapModeClean

Clear selected areas.

Function
fmMapModeClean()

Example (clear selecte areas)
container_mc.fmMapModeClean();

3.2.18. fmPOIsShowCategory

Show POIs pertaining to one or every category.

Function
fmPOIsShowCategory(category_str)

Parameter
category_str (string) ID of the category to show or * for every one.

Example (show hotels)
container_mc.fmPOIsShowCategory("hotel");
Example (show all POI categories)
container_mc.fmPOIsShowCategory("*");

3.2.19. fmPOIsHideCategory

Hide POIs pertaining to one or every category.

Function
fmPOIsHideCategory(category_str)

Parameter
category_str (string) ID of the category to hide or "*" for every one.

Example (hide hotels)
container_mc.fmPOIsHideCategory("hotel");
Example (hide all POI categories)
container_mc.fmPOIsHideCategory("*");

3.2.20. fmPOIAddEvent

Execute a specific event.

Function
fmPOIAddEvent(event_str, target_str, url_str, id_str)

Parameter
event_str (string) Event when action will be launched. Can have the following values:
onRelease: invoked when clicking over an area.

onRollOver: invoked when hovering over an area.
onRollOut: invoked when the mouse pointer leaves the area.

target_str (string) Type of action to launch. Can have the following values:
_MC: pop up a Flash movieclip.
_JS: call a JavaScript function.
_AS: call an AreaSelector internal function.
_blank, **_self**, **_top**: open a web page.

url_str (string) Action to launch. Its value depends on the type of action.
_MC: path to the movieclip to load.
_JS: name of the JavaScript function called. The event and area IDs are sent as parameters.
_AS: Action name to be launched.
_blank, **_self**, **_top**: web page's URL. The ID is sent as **poi_id** parameter.

Id_str (string) POI ID.

Example (when object 12 is clicked, open an URL on a new browser window)

```
container_mc.fmPOIAddEvent("onRelease", "_blank", "http://www.flashmaps.com", "12");
```

3.2.21. fmPOIRollOver

Launch onRollover event of a specific POI.

Function

```
fmPOIRollOver(id_str)
```

Parameter

Id_str (string) POI's ID.

Example (launch the onRollOver event on POI 12)

```
container_mc.fmPOIRollOver("12");
```

3.2.22. fmPOIRollOut

Launch onRollout event of a specific POI.

Function

```
fmPOIRollOut(id_str)
```

Parameter

id_str (string) POI's ID.

Example (launch the onRollOut event on POI 12)

```
container_mc.fmPOIRollOut("12");
```

3.2.23. fmPOIHighlight

Highlight a specific POI.

Function
fmPOIHighlight(id_str)

Parameter
Id_str (string) POI's ID.

Example (highlight POI 12)
```
container_mc.fmPOIHighlight("12");
```

3.2.24. fmPOIUnhighlight

Unighlight all POIs.

Function
fmPOIUnhighlight()

Example (unhighlight all POIs)
```
container_mc.fmPOIUnhighlight();
```

3.2.25. fmShowAlert

Show an alert window with a custom message over AreaSelector.

Function
fmShowAlert(title_str, text_str)

Parameter
title_str (string) Title of the alert window.
text_str (string) Text inside the alert window.
Example (show an alert window
```
container_mc.fmShowAlert("Warning!", "Please type in a zip code.");
```

3.2.26. fmHideAlert

Hide the alert window.

Function
fmHideAlert()

Example (hide the alert window)
```
container_mc.fmHideAlert();
```

3.2.27. fmShowCrossHair

Focus on a certain point and display a cross hair over it.

Function

```
fmShowCrossHair(lat, lon, scale)
Example (show a crosshair and focus on 40.35, -90.56)
container_mc.fmShowCrossHair("40.35", "-90.56", "2000");
```

3.2.28. fmHideCrossHair

Remove the cross hair from the map.

```
Function
fmHideCrossHair()
Example (hide the crosshair)
container_mc.fmHideCrossHair();
```

3.2.29. fmPOITextShow

Show texts in the middle of areas.

```
Function
fmShowPOIText()

Example (show texts in the middle of areas)
container_mc.fmShowPOIText();
```

3.2.30. fmPOITextHide

Hide texts in the middle of areas.

```
Function
fmHidePOIText()

Example (hide texts in the middle of areas)
container_mc.fmHidePOIText();
```

3.2.31. fmFromASReady (Event)

AreaSelector calls this ActionScript function in the container when a theme has been completely loaded (areas, POIs and polylines), or when areas and POIs have been refreshed after zooming in an area or zooming out of it (change of map level).

```
Event
function fmFromASPOIsLoaded() {
  // Your code here
}
```

3.2.32. fmFromASArea (Event)

AreaSelector calls this ActionScript function in the container when the user has zoomed in an area (by clicking it) or out of it (by clicking *back* button).

It is also called when an area event type '*_AS*' is triggered. Get more details on this in section *2.3. Area Categories XML Feed*.

Event

```
function fmFromASArea(event_str:String, id_str:String, url_str:String, label_str:String) {
  // Your code here
}
```

Parameters

event_str: The event that has triggered the action: **'ZoomIn'**, **'ZoomOut'**, **'Move'**, **'onRollover'**, **'onRollout'** and **'onRelease'**.
id_str: Area's ID.
url_str: A string containing information about the area.
label_str: Area's label.

3.2.33. fmFromASAreaLatLon (Event)

AreaSelector calls this ActionScript function in the container when an area event type '*_AS*' which *url* attribute is '*getLatLon*'. Get more details on this in section *2.3. Area Categories XML Feed*.

This function lets you now the coordinates where the user has clicked.

Event

```
function fmFromASAreaLatLon(lat:Number, lon:Number) {
  // Your code here
}
```

Parameters

lat: Decimal latitude (DDD.DDDD, not DDD.MMSS) of the point where the user has clicked.
lon: Decimal longitude of the point where the user has clicked.

3.2.34. fmFromASAreaLatLonScale (Event)

AreaSelector calls this ActionScript function in the container when an area event type '*_AS*' which *url* attribute is '*getLatLonScale*'. Get more details on this in section *2.3. Area Categories XML Feed*.

This function lets you now the coordinates, scale and area where the user has clicked.

Event

```
function fmFromASAreaLatLonScale(id_str:String, lat:Number, lon:Number, scale:Number) {
  // Your code here
}
```

Parameters

id_str: Area's ID.
lat: Decimal latitude (DDD.DDDD, not DDD.MMSS) of the point the user has clicked.
lon: Decimal longitude (DDD.DDDD, not DDD.MMSS) of the point the user has clicked.
scale: Current view's scale.

3.2.35. fmFromASPOI (Event)

AreaSelector calls this ActionScript function in the container when a POI event type '*_AS*' is triggered. Find more details in section *2.5. POI Categories XML Feed*.

Event

```
function fmFromASPOI(event_str:String, id_str:String, url_str:String, label_str:String) {
  // Your code here
}
```

Parameters

event_str: The event that has triggered the action: **'onRollover'**, **'onRollout'** and **'onRelease'**.
id_str: POI's ID.
url_str: A string containing information about the POI.
label_str: POI's label.

3.2.36. fmFromASPOIsLoaded (Event)

AreaSelector calls this ActionScript function in the container when a set of POIs has been completely loaded, to let the container know the number of POIs.

Event

```
function fmFromASPOIsLoaded(total:Number) {
  // Your code here
}
```

Parameters

total: The number of POis that have been loaded.

3.2.37. fmFromASPolyline (Event)

AreaSelector calls this ActionScript function in the container when a polyline event type "*_AS*' is triggered. Get more details on this in section *2.7. Polyline Categories XML Feed*.

Event

```
function fmFromASPolyline(event_str:String, id_str:String, url_str:String, label_str:String) {
  // Your code here
}
```

Parameters

event_str: The event that has triggered the action: **'onRollover'**, **'onRollout'** and **'onRelease'**.
id_str: Polyline's ID.
url_str: A string containing information about the polyline.
label_str: Polyline's label.

4. Instancing AreaSelector

AreaSelector object can receive a set of parameters when instanced from within HTML or a Flash container to perform certain tasks as it loads:

4.1. customTheme

Set the initial theme.

Example
fmASToolbar.swf?customTheme=areas/fm-us.php

4.2. customArea

Focus initially on a certain area.

Note that, if the area you want to focus on is not on the current map, but on a child one, you will need to include the full path of IDs, using the comma character as separator. For example, while loading a map of the USA, setting customArea to 'us_ca,06075' will make AreaSelector load the child map CA (California) and, inside it, focus on county 06075 (San Francisco).

Examples
On the USA map, focus on California:
fmASToolbar.swf?customArea=us_ca
On the world map, focus on Northamerica and then on Canada:
fmASToolbar.swf?customArea=na,ca

4.3. ASConfig

Load an alternative configuration XML file, overriding the default one (fmASEngine.xml).

Example
fmASToolbar.swf?ASConfig=configs/fmASEngine01.xml

4.4. absPath

Set the path to AreaSelector files. The path is relative to the HTML page that contains the map.

If not specified, AreaSelector files are expected in the same folder as fmASToolbar.swf.

Example
fmASToolbar.swf?absPath=Flashmaps/areaselector/

4.5. focusOnMap

Make AreaSelector focus on the map loaded initially.

Example
fmASToolbar.swf?focusOnMap=yes

4.6. focusOnPOIs

Make AreaSelector focus on the initial set of POIs.

Example
fmASToolbar.swf?focusOnPOIs=yes

5. How to

5.1. Insert AreaSelector into your HTML code

This is the way you should insert your AreaSelector Flash object inside your HTML code:

```
fmObjectActivateWrite("fmASEngine","fmASMap/fmASToolbar.swf", "500", "300",
"true", "high", "#FFFFFF", ".");
```

Note that the toolbar's URL (**fmASMap/fmASToolbar.swf**) may be changed to send parameters like **customTheme**, **focusOnMap**, etc. If your Flashmaps AreaSelector comes with a parent flash container, the container's file name replaces the toolbar's one in the URL of the activation code (fmASMap/fmASContainer,swf). The toolbar will be loaded subsequently by the flash container.

However, we recommend to copy the entire code lines, starting with <!-- FLASHMAPS - MAP OBJECT - BEGIN --> up to the line <!-- FLASHMAPS - MAP OBJECT - END -->, from the <body> section of the template pages (usually default.html/php/asp etc., depending upon the version) of your product into the body section of your page where you want to display the map. That way you will also include the flash player detection script and maybe individual customizations of your product (if any). Your evaluation pack or final project includes working examples of inserting Flashmaps AreaSelector into not only HTML, but also the server scripting technology you chose.

You also need to include two JavaScript modules before instancing the Flash object (inside <head> section is a good place to do it):

```
<script language="JavaScript" src="include/fmActivate.js"></script>
<script language="JavaScript" src="include/fmASAPI.js"></script>
```

Again, you can copy the code from the head section of the template pages from <!-- FLASHMAPS - FUNCTION LIBRARIES - BEGIN --> to <!-- FLASHMAPS - FUNCTION LIBRARIES - END -->.
Depending upon the individual requested customizations, there might be additional JavaScript functions or HTML code (AJAX results tables, etc.) to copy.

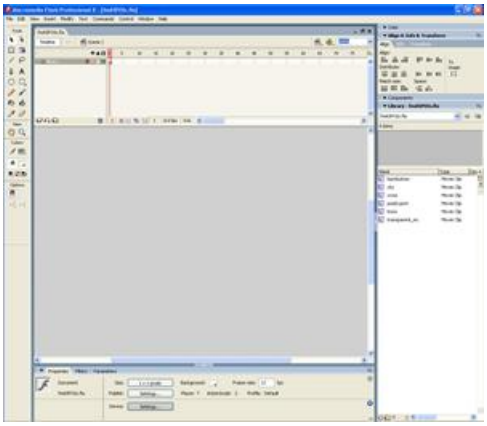
To put your Flashmaps AreaSelector application to work, you will also need to copy the folders 'fmASMap' and 'include', with all their files and subfolders, into the directory of your page with the map (or alternatively change the links to these files). The directory 'images' includes images and stylesheets for the template pages of the product, and are not necessary for using the Flashmaps AreaSelector in your new pages.

5.2. Insert or modify an icon in the library

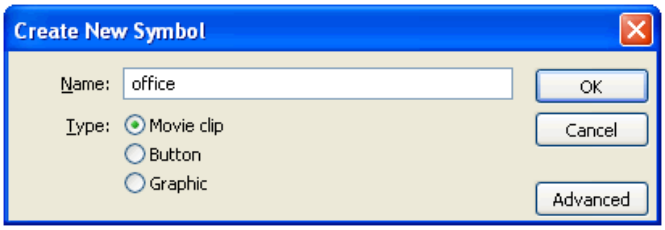
To add or modify POI icons in the icon library, you must edit the fmASIconLibrary.fla with Adobe Flash 8 (or later).

The steps to add an icon are the following:

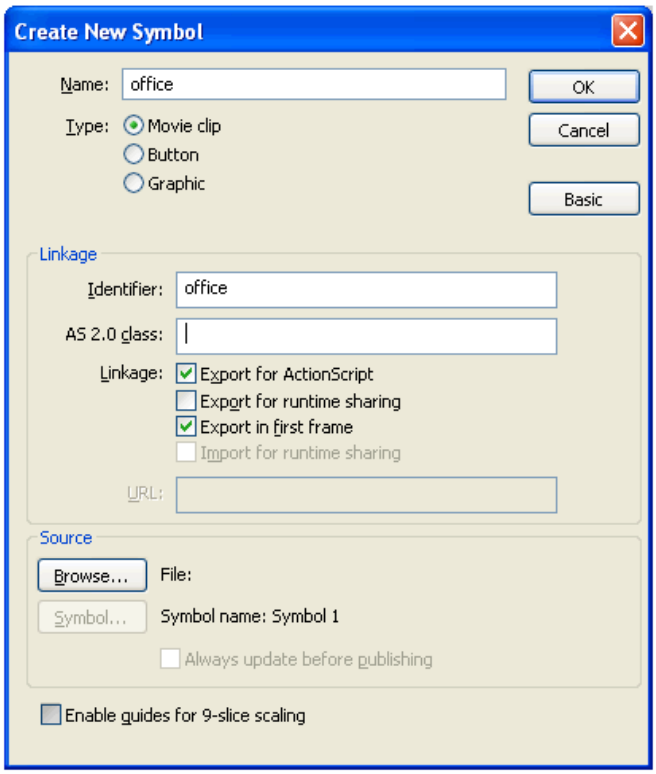
- Open fmASLibrarys.fla with Adobe Flash 8 (or later).



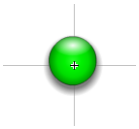
- Insert a new symbol (Insert menu → New Symbol) and name the new icon. Select the type: Movie clip.



- Press the Advanced button and select the checkboxes "Export for Action Script" and "Export in first frame". Use the same icon name in the Identifier textbox.



- Draw the icon in this area. Note that the icon will be centered at the position 0, 0, where the cross is displayed.



- Once you have compiled your movie clip (Ctrl + Enter), your new icon will be inserted in your icon library and ready to use. Feel free to add as many icons as you wish.

You can now reference this new POI icon from the POI categories XML feed and from the POIs XML feed. Find full information in sections *4.5 POI Categories XML Feed* and *4.6 POIs XML Feed.*

Make sure to clear your browser's cache (remove Internet temporary files) before you check out the results.