



## AUTO PILOT MODE

## 1. Introduction and scope

The Auto Pilot Mode (APM) controls some of the Map Intelligence Tools (which we shall refer to in this topic as plugins) depending on the document being manipulated by the Dashboard Client. For example, based on the name of a business intelligence document, a Layer Designer can control:

- what plugins must be excluded;
- what plugins must be deactivated, and
- what plugins must be activated and how.

 An excluded plugin will not even appear in the list of plugins usually visible when the user clicks the **Tools** menu button in the Map Intelligence **Mapping Viewer**.

 Auto Pilot mode requires some programming and access to the Map Intelligence server.

## 2. External elements of the APM

The default behavior of the Auto Pilot Mode, is defined by the property `plugin.automation` in the properties file `plugins.common.properties` in the directory `WEB-INF/properties` of a running Map Intelligence webapp. Here is how it is documented in the properties file:


```
# Indicate whether or not plugin automation settings should be applied to new  
  
# incoming requests to the Map Intelligence Server. Acceptable values for this  
  
# property are 'true' to enable the plugin automation scheme, while any other  
  
# value will disable that scheme. The default value is 'true'.
```

```
plugin.automation=true
```

When this property is set to TRUE, after a successful submission of a request from a client the **Info Bar** of the Map Intelligence **Mapping Viewer** in a web browser will show the following **Info Bar** display.




Fig 54. The Auto Pilot Mode set turned on.

Notice the Auto Pilot Mode icon  at the right end of the display. Hovering over it should display a tool-tip message that tells the user to click it in order to turn OFF the Auto Pilot Mode.

The End User can click on Auto Pilot Mode icon to toggle the Auto Pilot Mode state; i.e. If it was enabled – as is the case in the figure above– then clicking it will turn the Auto Pilot Mode OFF. When it is OFF, the icon changes to the following:




Fig 55. The Auto Pilot Mode turned off.

-  End Users can toggle this button as many times as they wish. The state of the APM will change every time in the Browser only. The new state of the APM only be taken into consideration when an operation is submitted to the Map Intelligence Server.

### 3. Internal elements of the APM

The behavior of the Auto Pilot Mode is controlled first by a file called `plugin-automation.map`

and second by a collection of files that have a `.pap` extension (also called *PAP* files short for *Plugin Automation Profile*). All these files live in `WEB-INF/data/profiles/plugin` of a running Map Intelligence webapp. The following is an exhaustive explanation of what these files look like and what they do.

-  Please note that all the text in the following documents is case sensitive.

## The `plugin-automation.map` file

This file is a global map between document names and (a) the plugins that must be excluded and (b) the list of profiles that must be considered, when the request submitted from the client hits the Map Intelligence Server. It is an XML file that looks like:

```
<?xml version="1.0" encoding="UTF-8"?>

<document-profiles>

  <document-entry name-expression=".*Crimes analysis.*">

    <profile-name>rsn1</profile-name>

    <profile-name>rsn2</profile-name>

    <excluded-plugin>forge.gradient</excluded-plugin>

  </document-entry>

  <document-entry name-expression=".*Property listings.*">

    <profile-name>rsn3</profile-name>

    <excluded-plugin>forge.webServices.mapImage</excluded-plugin>

  </document-entry>

  <document-entry name-expression=".*">

    <profile-name>default</profile-name>

  </document-entry>

</document-profiles>
```

As you can see the file is the representation of a single `document-profiles` element which is a collection of `document-entry` elements.

## The `document-entry` element

Each such element indicates to the Automaton (of the APM) what to do if/when the document's name, in the submitted request, matches the name indicated in this element.

## The `name-expression` attribute

This attribute is a *Regular Expression* that the Automaton uses as a *Pattern* to match the document's name from the submitted request. For example, the regular expression of the first `document-entry` element in the example above means that any document that contains the string “Crimes analysis” anywhere in its name is a match.

The syntax of this attribute MUST be a valid string that the `java.util.regex.Pattern` class can understand, and interpret. *Annex A* at the end of this section, contains a summary of regular expression constructs that the `Pattern` class, and by extension the Automaton, can handle.

 A good tutorial on using regular expressions in Java can be found at:

<http://www.regular-expressions.info/java.html>.

## The `profile-name` element

Each of these elements shall denote the name of a PAP file (a file with a `.pap` extension in the folder `WEB-INF/data/profiles/plugin` of a running Map Intelligence webapp).

## The `excluded-plugin` element

Each of these elements denotes the unique identifier of a plugin that will be excluded from the user's session. As mentioned earlier, an excluded plugin will not even appear in the list of plugins displayed when the user clicks on the **Tools** menu button in the **Mapping Viewer**.

## Business rules applicable to the `plugin-automation.map` file

- There MUST be one and only one such file in a running Map Intelligence webapp.
- The Automaton will respect the order of the `document-entry` elements in the map file. It will not process the second element before processing the first one and so on.
- The APM Automaton will select the first `document-entry` element that has a `name-expression` regular expression matching the document's name in the request. Using the same

example as above, for a document named “Monthly\_Crimes analysis\_2.1.4,” the APM Automaton will select the first document-entry in the list.

- The APM Automaton ensures that an excluded plugin is not visible in the list of plugins in the current session. The list of available plugins, visible when the user clicks the **Tools** button of the **Mapping Viewer** does not contain the name of an excluded plugin. For example, if the first (in the list) document-entry is processed, the Point Gradient plugin's name does not appear (as given in the image below) in the list of available plugins.

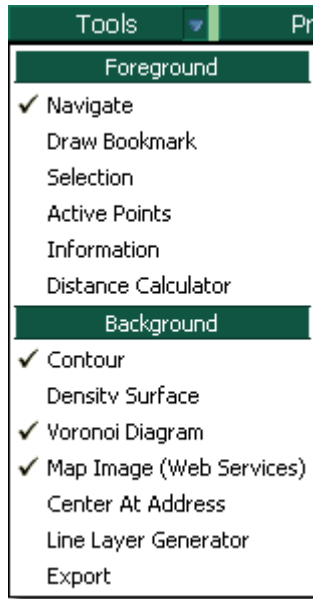


Fig 56. The Tools menu list with the Point Gradient tool excluded.

## The PAP files

The Plugin Automation Profile, or Profile, or PAP file, is an XML file that tells the Automaton when and what to do with regard to one or more Map Intelligence plugins. Here is an example of a PAP file:

```
<?xml version="1.0" encoding="UTF-8"?>

<automation-profile>

  <activation-condition><![CDATA[

    #view.getMapName() == 'nsw.mdf'
```

```
&& #view.getZoom() >= 3.0

&& #view.getZoom() <= 4.0

]]></activation-condition>

<plugin-setting plugin-id="forge.contour">

  <parameter-setting name="forge.contour.point.layer.option.list"

    value="literal:Oracle Crimes"/>

  <parameter-setting name="contour.point.layer.column.options_Oracle Crimes"

    value="literal:VALUE"/>

</plugin-setting>

<plugin-setting plugin-id="forge.voronoi">

  <parameter-setting name="forge.voronoi.point.layer.option.list"

    value="literal:Police Stations"/>

  <parameter-setting name="forge.voronoi.opacity"

    value="literal:80"/>

</plugin-setting>

<plugin-setting plugin-id="forge.webServices.mapImage">

  <parameter-setting name="forge.webServices.mapImage.data.source"

    value="literal:Streets"/>

</plugin-setting>
```

```
<parameter-setting name="forge.webServices.mapImage.data.source_Streets.child"
    value="literal:MDS.Streets.AS (4)"/>
</plugin-setting>
</automation-profile>
```

As you can see there is always a single automation-profile element in such a file.

### The `activation-condition` element

This element (or more correctly its contents) is a representation of a boolean expression which the Automaton will evaluate, at run-time, to decide whether, or not, to apply the rest of the profile; i.e. Deactivate or activate the plugin.

The expression itself is written in OGNL (Object Graph Navigation Language) –an expression language for getting and setting properties of Java objects.



A comprehensive Language Guide (for the version used in the current implementation of the APM) is available at <http://www.ognl.org/2.6.7/Documentation/html/LanguageGuide/>.

The current implementation of the APM offers the writer of such expressions access to two of the core Map Intelligence Server top-level objects: the `View` and the `ParameterMap`.

References to the `View` object are made using the syntax `#view` while references to the `ParameterMap` instance are made using the syntax `#params`. Having access to those objects means that the writer of an `activation-condition` can refer to all the public methods available on these objects.

To illustrate, the example above will be interpreted by the Automaton as follows:

*if the return value of the `getMapName()` method, invoked on the `View` object, is equal to the string literal "nsw.mdf," AND the return value of the `getZoom()` method, invoked on the same `View` object, is between 3.0 and 4.0 inclusive, then return `TRUE`, otherwise return `FALSE`.*

## The `plugin-setting` element

Every such element tells the Automaton which plugin to activate (when the `activation-condition` evaluates to TRUE) and how to setup that plugin for it to do what it is supposed to do.

## The `plugin-id` attribute

This attribute contains the unique identifier of the Map Intelligence plugin/tool to automate.

## The `parameter-setting` element

Each such element tells the Automaton, for the plugin identified by the `plugin-id` attribute, what plugin parameter to set and what value to set it to.

## The `name` attribute

The value of this attribute is a plugin-specific property name.

## The `value` attribute

The value of this attribute is a plugin specific value represented as a Java string. A prefix at the beginning of this string will tell the Automaton how to consider the rest. Possible values of the prefix are:

- `literal`:

to mean that what follows should be passed as is to the plugin's property,

- `ognl`:

to mean that what follows should be evaluated as an OGNL expression, and the result (as a string) should be passed to the plugin's property.

*Annex B* at the end of this section, lists all *programmable* plugins and the names of their settable parameters.

## Conflict resolution and automation algorithm

As mentioned earlier, once a document-entry element is selected (because its `name-expression` matched the document's name) the Automaton will consider all the PAPs referenced by the `profile-name` elements found in this document-entry element. It was also mentioned earlier that when processing a profile, the Automaton will activate a plugin if the profile's `activation-condition` evaluates to TRUE, and will deactivate it if the same condition evaluates to FALSE. But what happens if the same plugin (a) is referenced in two profiles, and (b) in one profile the



activation condition evaluates to FALSE while it evaluates to TRUE in the other? Will the plugin be activated or deactivated?

Furthermore, what if the same plugin is referenced in two profiles that belong to the same document-entry? Both profiles have their activation condition evaluate to TRUE, but one profile sets a parameter with one value, while the other sets the same parameter to another value. Which value will that parameter ultimately have: the one in the first profile or the second one?

These two cases describe the possible type of conflicts that may arise from the scheme we described so far.

### **To activate or not to activate... that is the question**

The APM Automaton will guarantee that a plugin that is activated by one Profile will not be deactivated by another when both profiles are referenced in one document-entry element. It does so by keeping track of two lists: `pluginsAlreadyActivated` and `pluginsToDeactivate`. When a plugin is to be activated, it is first removed from the `pluginsToDeactivate` list (if it is already there) and then added to the list of `pluginsAlreadyActivated` after it is effectively activated. On the other hand, if a plugin is to be deactivated, the Automaton will check if it is not already present in the `pluginsAlreadyActivated`. Only when it is not there, will it get added to the `pluginsToDeactivate` list. After processing all the Profiles, the Automaton then deactivates all the plugins that are left in the `pluginsToDeactivate` list.

### **Plugin parameter values and evaluation order**

The Automaton will guarantee that it will process the automation profiles in the order they are given in the `document-entry` element it picks for a document. It is then left to the writer of the automation profiles to (a) encode the plugins and their desired parameter values in as many profiles she wants, and (b) order those profiles in a way that ensures predictability of the outcome.

Furthermore, the Automaton will not overwrite plugin parameters that have not been set in a profile thus allowing a user to change their values in the GUI without fear of them being over-ridden by the profile.

### **Rules of thumb to improve predictability**

The following is a list of some rules of thumb that a profile writer can adopt and follow in order to predict the outcome of plugin automation.

1. When writing `document-entry` elements, specify as much as possible the full document name.
2. Order your document-entry elements from the SPECIFIC to the GENERAL; i.e. Put those with fully

named documents before those which have a regular expression pattern. Furthermore, put the ones with fewer regular expression constructs before the ones with more constructs.

3. Try describing the plugins in separate automation profiles: i.e. reduce the number of plugins referred to in PAP files. You can ultimately dedicate one profile per plugin.
4. Organise your PAP files that refer to the same plugin, in exclusive conditions: i.e. reduce the occurrences where two profiles will have their activation-condition evaluated to TRUE but which set the plugin's parameter(s) differently.
5. Set as few plugin parameters as possible. In other words **DO NOT SET ALL THE PLUGIN'S PARAMETERS** unless the default values provided by the plugin do not suit your needs. This is because a plugin parameter that is not set in an automation profile can be changed by the user (from the GUI) , without it being over-written by the Automaton.

## Annex A: Summary of regular-expression constructs

Construct	Matches
<b>Characters</b>	
<code>x</code>	The character <code>x</code>
<code>\\</code>	The backslash character
<code>\\0n</code>	The character with octal value <code>0n</code> ( $0 \leq n \leq 7$ )
<code>\\0nn</code>	The character with octal value <code>0nn</code> ( $0 \leq n \leq 7$ )
<code>\\0mnn</code>	The character with octal value <code>0mnn</code> ( $0 \leq m \leq 3, 0 \leq n \leq 7$ )
<code>\\xhh</code>	The character with hexadecimal value <code>0xhh</code>
<code>\\uhhhh</code>	The character with hexadecimal value <code>0xhhhh</code>
<code>\\t</code>	The tab character ( <code>"\u0009"</code> )
<code>\\n</code>	The newline (line feed) character ( <code>"\u000A"</code> )
<code>\\r</code>	The carriage-return character ( <code>"\u000D"</code> )
<code>\\f</code>	The form-feed character ( <code>"\u000C"</code> )
<code>\\a</code>	The alert (bell) character ( <code>"\u0007"</code> )
<code>\\e</code>	The escape character ( <code>"\u001B"</code> )
<code>\\cx</code>	The control character corresponding to <code>x</code>
<b>Character classes</b>	
<code>[abc]</code>	<code>a</code> , <code>b</code> , or <code>c</code> (simple class)
<code>[^abc]</code>	Any character except <code>a</code> , <code>b</code> , or <code>c</code> (negation)
<code>[a-zA-Z]</code>	<code>a</code> through <code>z</code> or <code>A</code> through <code>Z</code> , inclusive (range)
<code>[a-d[m-p]]</code>	<code>a</code> through <code>d</code> , or <code>m</code> through <code>p</code> : <code>[a-dm-p]</code> (union)
<code>[a-z&amp;&amp;[def]]</code>	<code>d</code> , <code>e</code> , or <code>f</code> (intersection)
<code>[a-z&amp;&amp;[^bc]]</code>	<code>a</code> through <code>z</code> , except for <code>b</code> and <code>c</code> : <code>[ad-z]</code> (subtraction)
<code>[a-z&amp;&amp;[^m-p]]</code>	<code>a</code> through <code>z</code> , and not <code>m</code> through <code>p</code> : <code>[a-lq-z]</code> (subtraction)
<b>Predefined character classes</b>	
<code>.</code>	Any character (may or may not match <a href="#">line terminators</a> )
<code>\\d</code>	A digit: <code>[0-9]</code>
<code>\\D</code>	A non-digit: <code>[^0-9]</code>
<code>\\s</code>	A whitespace character: <code>[ \\t\\n\\x0B\\f\\r]</code>
<code>\\S</code>	A non-whitespace character: <code>[^\\s]</code>
<code>\\w</code>	A word character: <code>[a-zA-Z_0-9]</code>
<code>\\W</code>	A non-word character: <code>[^\\w]</code>
<b>POSIX character classes (US-ASCII only)</b>	
<code>\\p{Lower}</code>	A lower-case alphabetic character: <code>[a-z]</code>
<code>\\p{Upper}</code>	An upper-case alphabetic character: <code>[A-Z]</code>
<code>\\p{ASCII}</code>	All ASCII: <code>[\\x00-\\x7F]</code>
<code>\\p{Alpha}</code>	An alphabetic character: <code>[\\p{Lower}\\p{Upper}]</code>
<code>\\p{Digit}</code>	A decimal digit: <code>[0-9]</code>
<code>\\p{Alnum}</code>	An alphanumeric character: <code>[\\p{Alpha}\\p{Digit}]</code>

<code>\p{Punct}</code>	Punctuation: One of !"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~
<code>\p{Graph}</code>	A visible character: <code>[\p{Alnum}\p{Punct}]</code>
<code>\p{Print}</code>	A printable character: <code>[\p{Graph}\x20]</code>
<code>\p{Blank}</code>	A space or a tab: <code>[ \t]</code>
<code>\p{Cntrl}</code>	A control character: <code>[\x00-\x1F\x7F]</code>
<code>\p{Xdigit}</code>	A hexadecimal digit: <code>[0-9a-fA-F]</code>
<code>\p{Space}</code>	A whitespace character: <code>[\t\n\x0B\f\r]</code>

Construct	Matches
<b>java.lang.Character classes (simple java character type)</b>	
<code>\p{javaLowerCase}</code>	Equivalent to <code>java.lang.Character.isLowerCase()</code>
<code>\p{javaUpperCase}</code>	Equivalent to <code>java.lang.Character.isUpperCase()</code>
<code>\p{javaWhitespace}</code>	Equivalent to <code>java.lang.Character.isWhitespace()</code>
<code>\p{javaMirrored}</code>	Equivalent to <code>java.lang.Character.isMirrored()</code>
Classes for Unicode blocks and categories	
<code>\p{InGreek}</code>	A character in the Greek block (simple block)
<code>\p{Lu}</code>	An uppercase letter (simple category)
<code>\p{Sc}</code>	A currency symbol
<code>\P{InGreek}</code>	Any character except one in the Greek block (negation)
<code>[\p{L}&amp;&amp;[^\p{Lu}]]</code>	Any letter except an uppercase letter (subtraction)
<b>Boundary matchers</b>	
<code>^</code>	The beginning of a line
<code>\$</code>	The end of a line
<code>\b</code>	A word boundary
<code>\B</code>	A non-word boundary
<code>\A</code>	The beginning of the input
<code>\G</code>	The end of the previous match
<code>\Z</code>	The end of the input but for the final <a href="#">terminator</a> , if any
<code>\z</code>	The end of the input
<b>Greedy quantifiers</b>	
<code>X?</code>	X, once or not at all
<code>X*</code>	X, zero or more times
<code>X+</code>	X+ X, one or more times
<code>X{n}</code>	X, exactly n times
<code>X{n,}</code>	X, at least n times
<code>X{n,m}</code>	X, at least n but not more than m times
<b>Reluctant quantifiers</b>	

<code>X??</code>	X, once or not at all
<code>X*?</code>	X, zero or more times
<code>X+?</code>	X, one or more times
<code>X{n}?</code>	X, exactly <i>n</i> times
<code>X{n,}?</code>	X, at least <i>n</i> times
<code>X{n,m}?</code>	X, at least <i>n</i> but not more than <i>m</i> times

**Possessive quantifiers**

<code>X?+</code>	X, once or not at all
<code>X*+</code>	X, zero or more times
<code>X++</code>	X, one or more times
<code>X{n}+</code>	X, exactly <i>n</i> times
<code>X{n,}+</code>	X, at least <i>n</i> times
<code>X{n,m}+</code>	X, at least <i>n</i> but not more than <i>m</i> times

**Logical operators**

<code>XY</code>	X followed by Y
<code>X Y</code>	Either X or Y
<code>(X)</code>	X, as a <a href="#">capturing group</a>

Construct	Matches
<b>Back references</b>	
<code>\n</code>	Whatever the <i>n</i> th <a href="#">capturing group</a> matched
<b>Quotation</b>	
<code>\</code>	Nothing, but quotes the following character
<code>\Q</code>	Nothing, but quotes all characters until <code>\E</code>
<code>\E</code>	Nothing, but ends quoting started by <code>\Q</code>
<b>Special constructs (non-capturing)</b>	
<code>(?:X)</code>	X, as a non-capturing group
<code>(?idmsux-idmsux)</code>	Nothing, but turns match flags on – off
<code>(?idmsux-idmsux:X)</code>	X, as a non-capturing group with the given flags on - off
<code>(?=X)</code>	X, via zero-width positive lookahead
<code>(?!X)</code>	X, via zero-width negative lookahead
<code>(?&lt;=X)</code>	X, via zero-width positive lookbehind
<code>(?&lt;!X)</code>	X, via zero-width negative lookbehind
<code>(?&gt;X)</code>	X, as an independent, non-capturing group

## Annex B: Programmable plugins and their parameter names

### Contour – `forge.contour`

- `forge.contour.point.layer.option.list`: The [Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `contour.point.layer.column.options_xxx`: The [Column] drop-down list, where xxx is the name of the layer selection from the previous drop-down list; e.g. If “Oracle Crimes” is the name of a valid layer that the user can select from the Layer dropdown list, then a valid name for the column property is

`contour.point.layer.column.options_Oracle Crimes`. Possible values include the column names of the layer in question.

- `forge.contour.unit.option.list`: The [Unit] drop-down list. Possible values include the strings which reference the units in that drop-down list. In the English version of Map Intelligence, this list consists of the following values: “kilometer”, “meter”, “mile” and “yard”.
- `forge.contour.cell.size`: The [Cell Size in Units] slider value. Possible values are string representation of integers ranging from 1 to 1000.
- `forge.contour.search.radius`: The [Search Radius in Cells] slider value. Possible values are string representations of integers ranging from 0 to 100.
- `forge.contour.level.count`: The [Number of Levels] slider value. Possible values are string representations of integers ranging from 2 to 9.
- `forge.contour.logarithmic.scale`: The [Logarithmic Scale] checkbox value. Possible values are “true” (for checked) and “false” (for unchecked).
- `forge.contour.area`: The [Outline v/s Paint Area] checkbox value. Possible values are “true” (for checked) and “false” (for unchecked).
- `forge.contour.show.labels`: The [Show Labels] checkbox value. Possible values are “true” (for checked) and “false” (for unchecked).
- `contour_startColour`: The [Color Range Start color] hexadecimal value. Possible values are “#rrggbb” where the pairs “rr”, “gg” and “bb” are hexadecimal digits in the range 00 to FF. The pairs “rr”,

“gg” and “bb” represent the “Red”, “Green” and “Blue” components of the colour respectively.

- `contour_endColour`: The [Color Range End color] hexadecimal value. Possible values are “#rrggbb” where the pairs “rr”, “gg” and “bb” are hexadecimal digits in the range 00 to FF. The pairs “rr”, “gg” and “bb” represent the “Red”, “Green” and “Blue” components of the colour respectively.
- `forge.contour.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.

## Density Surface – `forge.density.surface`

- `forge.densitySurface.point.layer.option.list`: The [Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `densitySurface.point.layer.column.options_XXX`: The [Column] drop-down list, where XXX is the name of the layer selection from the previous drop-down list; e.g. If “Oracle Crimes” is the name of a valid layer that the user can select from the Layer drop-down list, then a valid name for the column property is

`densitySurface.point.layer.column.options_Oracle Crimes`. Possible values include the column names of the layer in question.

- `forge.density.surface.unit.option.list`: The [Unit] drop-down list. Possible values include the strings which reference the units in that drop-down list. In the English version of Map Intelligence, this list consists of the following values: “kilometer”, “meter”, “mile” and “yard”.
- `forge.density.surface.cell.size`: The [Cell Size in Units] slider value. Possible values are string representation of integers ranging from 1 to 1000.
- `forge.density.surface.search.radius`: The [Search Radius in Cells] slider value. Possible values are string representations of integers ranging from 0 to 100.
- `forge.density.surface.class.count`: The [Number of Classes] slider value. Possible values are string representations of integers ranging from 2 to 9.
- `densitySurface_startColour`: The [Color Range Start color] hexadecimal value. Possible values are “#rrggbb” where the pairs “rr”, “gg” and “bb” are hexadecimal digits in the range 00 to FF. The pairs “rr”, “gg” and “bb” represent the “Red”, “Green” and “Blue” components of the colour respectively.

- `densitySurface_endColour`: The [Color Range End color] hexadecimal value. Possible values are “#rrggbb” where the pairs “rr”, “gg” and “bb” are hexadecimal digits in the range 00 to FF. The pairs “rr”, “gg” and “bb” represent the “Red”, “Green” and “Blue” components of the colour respectively.
- `forge.density.surface.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.

## Line Layer Generator - `forge.linelay`

- `forge.linelay.source.layer.list`: The [Source Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `forge.linelay.source.layer.list_XXX.child`: The [Column] drop-down list, where XXX is the name of the layer selection from the previous drop-down list; e.g. If “Agents” is the name of a valid layer that the user can select from the Source Layer drop-down list, then a valid name for the column property is

`forge.linelay.source.layer.list_Agents.child`. Possible values include the column names of the layer in question.

- `forge.linelay.dest.layer.list`: The [Destination Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `forge.linelay.dest.layer.list_XXX.child`: The [Column] drop-down list, where XXX is the name of the layer selection from the previous drop-down list; e.g. If “House Listings” is the name of a valid layer that the user can select from the Source Layer drop-down list, then a valid name for the column property is

`forge.linelay.dest.layer.list_House Listings.child`. Possible values include the column names of the layer in question.

- `forge.linelay.thickness`: The [Line Thickness] input field. Possible values are string representation of integers greater than 0.
- `forge.linelay.visible`: The [Visible] checkbox value. Possible values are “true” (for checked) and “false” (for unchecked).
- `forge.linelay.startColour`: The [Color Range Start color] hexadecimal value. Possible values are “#rrggbb” where the pairs “rr”, “gg” and “bb” are hexadecimal digits in the range 00 to FF. The pairs



“rr”, “gg” and “bb” represent the “Red”, “Green” and “Blue” components of the colour respectively.

- `forge.linelayer.endColour`: The [Color Range End color] hexadecimal value. Possible values are “#rrggbb” where the pairs “rr”, “gg” and “bb” are hexadecimal digits in the range 00 to FF. The pairs “rr”, “gg” and “bb” represent the “Red”, “Green” and “Blue” components of the colour respectively.

### Point Gradient – `forge.gradient`

- `forge.gradient.point.layer.option.list`: The [Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `gradient.point.layer.column.options_XXX`: The [Themes] drop-down list, where xxx is the name of the layer selection from the previous drop-down list.
- `forge.gradient.square.size.option.list`: The [Reach] drop-down list. Possible values are “LOW”, “MEDIUM” and “HIGH”.
- `forge.gradient.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.

### Voronoi – `forge.voronoi`

- `forge.voronoi.point.layer.option.list`: The [Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `forge.voronoi.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.

### Web Services Map Image – `forge.webServices.mapImage`

- `forge.webServices.mapImage.data.source`: The Service [Type] drop-down list. Possible values include: “Default”, “Cadastre”, “Census”, “Relief”, “Satellite”, “Streets”, “Terrain”, “Topo”, “Traffic”, “Weather”, and “Other”.

- `forge.webServices.mapImage.data.source_XXX.child`: The Data [Source] drop-down list, where xxx is the name of the Service Type selection from the previous dropdown list; e.g. If “Satellite” is the name of a valid Service Type that the user can select from the Service Type drop-down list, then a valid name for this property could be

`forge.webServices.mapImage.data.source_Satellite.child.`

- `forge.webServices.mapImage.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.

### **Web Services Drive Time - `forge.webServices.driveTime`**

- `forge.webServices.driveTime.point.layer.option.list`: The [Layer] drop-down list. Possible values include the strings which reference the layers visible in that dropdown list.
- `forge.webServices.driveTime.point.limit`: The [Maximum number of points to analyze] slider value. Possible values are string representations of integers ranging from 1 to 10.
- `forge.webServices.driveTime.polygons.per.point`: The [Number of polygons / point] slider value. Possible values are string representations of integers ranging from 1 to 4.
- `forge.webServices.driveTime.units.option.list`: The [Units] drop-down list. Possible values include: “Miles”, “Minutes” and “Kilometers”.
- `forge.webServices.driveTime.interval`: The [Interval] slider value. Possible values are string representations of integers ranging from 1 to 50.
- `forge.webServices.driveTime.inter.polygon.interval`: The [Inter-polygon Interval] slider value. Possible values are string representations of integers ranging from 1 to 15.
- `forge.webServices.driveTime.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.

## Charts On Regions - `forge.regionChart`

- `forge.regionChart.multi.point.layer.option.list`: The Region-Relationship [Layer] drop-down list. Possible values include the strings which reference the layers visible in that drop-down list.
- `regionChart_xxx.point.layer.column.options_yyy`: The [Classification column] dropdown list, where xxx is the name of the Point/Facts Layer of the selected Region-Relationship Layer from the previous drop-down list and yyy is the name of the column selected from that layer, where spaces in the names are replaced by underscores; e.g. If “Crimes by Suburb” is the name of a valid region-relationship layer that the user has selected from the [Layer] drop-down list, and “Crimes” is the Points/Facts Layer of that region-relationship layer, then the valid name for the corresponding classification column parameter is

`regionChart_Crimes_by_Suburb.point.layer.column.options_Crimes`. Possible values include the column names of the layer in question.

- `forge.regionChart.class.count`: The [Number of classes] input field. Possible values are string representations of integers.
- `regionChart_xxx.point.layer.aggregation.column.options_yyy`: The [Aggregation column] drop-down list, where xxx is the name of the Point/Facts Layer of the selected Region-Relationship Layer from the previous drop-down list and yyy is the name of the column selected from that layer, where spaces in the names are replaced by underscores; e.g. If “Crimes by Suburb” is the name of a valid regionrelationship layer that the user has selected from the [Layer] drop-down list, and “Crimes” is the Points/Facts Layer of that region-relationship layer, then a valid name for the aggregation column parameter is

`regionChart_Crimes_by_Suburb.point.layer.aggregation.column.options_Crimes`. Possible values include the column names of the layer in question.

- `forge.regionChart.aggregation.numeric.type.options`: The [Aggregation type] dropdown list. Possible values for the English US variant of Map Intelligence are: “Count”, “Average”, “Sum”, “Minimum” and “Maximum”.
- `forge.regionChart.chart.type.options`: The [Type] drop-down list in the [Chart parameters] section. Possible values for the English US variant of Map Intelligence are: “Bar”, and “Pie”.
- `forge.regionChart.chart.count`: The [Count] input field in the [Chart parameters] section. Possible values are string representations of integers.

- `forge.regionChart.chart.size`: The [Size] input field in the [Chart parameters] section. Possible values are string representations of integers.
- `forge.regionChart.opacity`: The [Opacity] slider value. Possible values are string representations of integers ranging from 1 to 100.