

# Installation & Configuration manual



## **ProcessMap for SharePoint**

Version 3.1

## Copyright

This manual is protected by copyright law. Changes in content or partly copying are not allowed without authorization from the copyright holder.



Installation & Administration Manual for Process Map version 3.1

Created September 1, 2006

Last edited February 16, 2011

©Meridium Kalmar

## Content

<b>1.</b>	<b>Introduction</b>	<b>2</b>
1.1.	Prerequisites	2
1.2.	Manual conventions	2
1.3.	Relation to other manuals	2
<b>2.</b>	<b>Requirements</b>	<b>3</b>
2.1.	Server	3
2.2.	Client	3
<b>3.</b>	<b>Installation</b>	<b>4</b>
3.1.	Installation Step 1 – Adding the solution	4
3.2.	Installation Step 2 – Deploying the solution	4
3.3.	Installation Step 3 – Activating the feature	5
3.4.	Installation Step 4 – License	5
3.5.	Installation Step 5 – Configuration	5
<b>4.</b>	<b>Editor configuration</b>	<b>7</b>
<b>5.</b>	<b>Security configuration modifications</b>	<b>13</b>
5.1.	Server settings	13
5.2.	Client settings	13
<b>6.</b>	<b>Authentication modes and security settings</b>	<b>14</b>
6.1.	Local windows user is not the same as the SharePoint user	14
6.2.	License	14
<b>7.</b>	<b>Configuration</b>	<b>15</b>
7.1.	ProcessMapExtension section	15
7.2.	ConfigSections	15
7.3.	Modules and Handlers	15
7.4.	Application Settings	16
<b>8.</b>	<b>Create new symbols</b>	<b>17</b>
8.1.	Symbol xml definition	17
<b>9.</b>	<b>Property Rules</b>	<b>20</b>
9.1.	Conditions	21
<b>10.</b>	<b>Web Part API</b>	<b>22</b>
<b>11.</b>	<b>Logging</b>	<b>23</b>

## 1. Introduction

Welcome!

ProcessMap is a plug-in for SharePoint. It is used to visualize processes and to connect the various steps of a process to e.g. documents, web pages or other sources of information. The ProcessMap plug-in is fully integrated into the SharePoint environment and can utilize standard SharePoint functions like search and version control. Process maps are easily created with the built in and easy to use drawing tool.

This manual is aimed towards an administrator or developer.

### 1.1. Prerequisites

The administrator will need a thorough understanding of SharePoint administration. The developer will need thorough understanding of xml and asp.net.

### 1.2. Manual conventions

Certain typographic conventions are used in this manual.

Running text is presented in the times font. Notes, tips and warnings are presented in bold.

Code is written in the courier typeface

```
print "Hello world"
```

**Note!** A note. Highlights important information.

**Tips!** A tip. Contains an advice or an easier way to do something.

**Warning!** A warning! Highlights a problem that might occur and how to avoid it.

### 1.3. Relation to other manuals

This manual is part of a series of two manuals. The other manual is ProcessMap: Editor manual. This manual is sufficient reading for an administrator or developer.

## 2. Requirements

### 2.1. Server

ProcessMap works with the following versions of the .NET Framework and SharePoint:

- .NET 3.5
- Windows SharePoint Services v3 and Microsoft Office SharePoint Server 2007

Make sure you have the installation package that suits your needs.

### 2.2. Client

The client that should view the process maps requires support for javascript but other than that follows SharePoint recommendations for client hardware and software.

Clients that should edit the process maps must have Internet Explorer 6 or later or Firefox with a plugin that allows Firefox to run .Net ClickOnce applications.

Clients must have Microsoft .NET framework runtime version 3.5 or later installed.

Vista clients must have the process map website added as a trusted site.

Windows XP and Windows Vista are supported client operating systems.

### 3. Installation

ProcessMap is installed as a standard SharePoint solution. This will copy the needed files, and makes the necessary modifications of the system configuration.

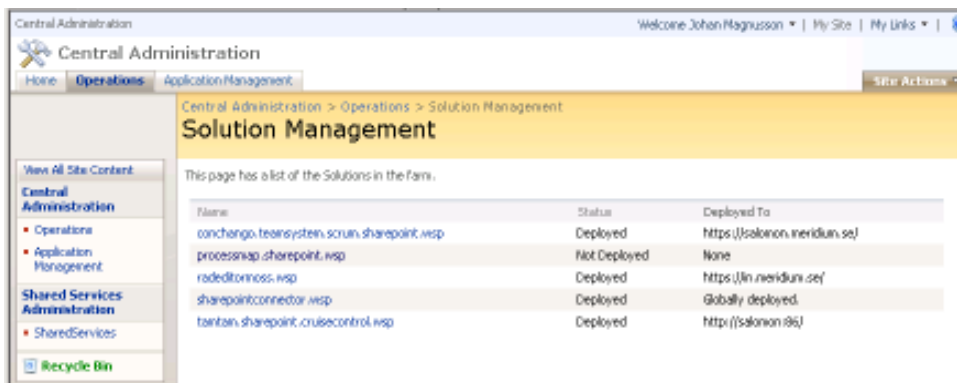
#### 3.1. Installation Step 1 – Adding the solution

The first step in installing ProcessMap is adding the solution to SharePoint. This is done with stsadm using the command below:

```
stsadm -o addsolution -filename ProcessMap.SharePoint.wsp -f
```

**Note! Stsadm is usually installed at this location: C:\Program Files\Common Files\Microsoft Shared\Web Server Extensions\12\BIN\stsadm.exe**

After the command has executed the solution will be available in SharePoint Central Administration under Central Administration > Operations > Solution Management



#### 3.2. Installation Step 2 – Deploying the solution

Next step is to deploy the solution. ProcessMap is deployed globally, meaning it will be available on all sites once deployed.

To deploy the solution click on the solution in Solution Management and then on the Deploy Solution link.

Or use:

```
stsadm -o deploysolution -name ProcessMap.sharepoint.wsp
-allowGacDeployment -immediate
```

This will schedule the solution for deployment. In most cases the deployment will be executed almost immediately but if the job doesn't seem to execute it can be triggered manually by the following command:

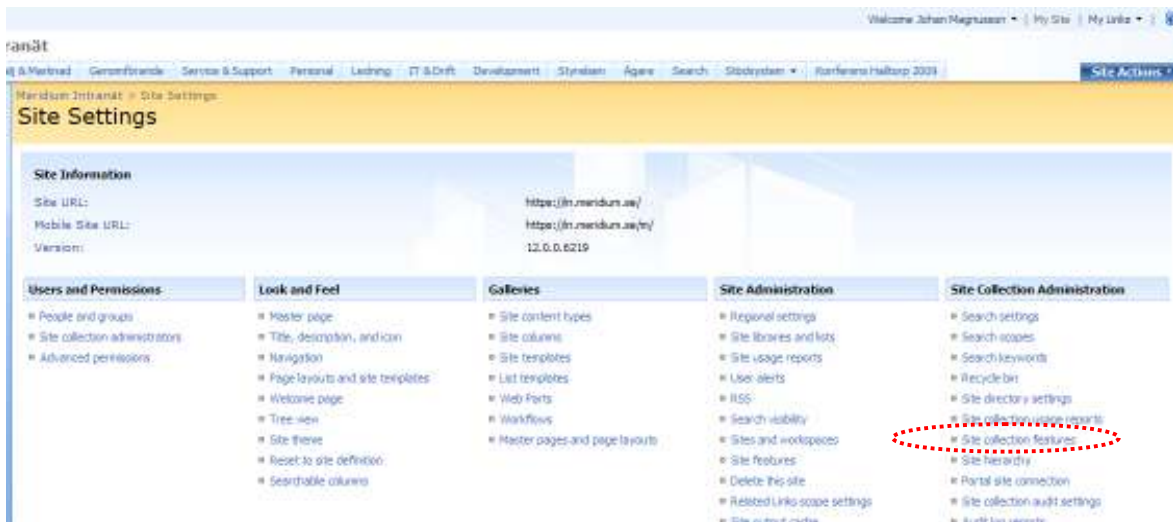
```
stsadm -o execadmsvcjobs
```

### 3.3. Installation Step 3 – Activating the feature

Once the solution is deployed it must be activated on the sites where it should be used.

Open up Site Settings for the site.

Then click on “Site collection features” under “Site Collection Administration”



Then click on the “Activate” button after the ProcessMap feature. Once it is activated status will show as “Active”.

Or use the command line version:

```
stsadm -o activatefeature -name ProcessMap -url URL_to_your_site
```

Where URL\_to\_your\_site should be replaced with the URL to the site where you want to activate the ProcessMap feature.

### 3.4. Installation Step 4 – License

Copy the license file meridiumLicense.config to the root folder of the site.

E.g. C:\Inetpub\wwwroot\wss\VirtualDirectories\80\

Make sure that Everyone has read access to it.

### 3.5. Installation Step 5 – Configuration

After installation there are some configuration that need to done.

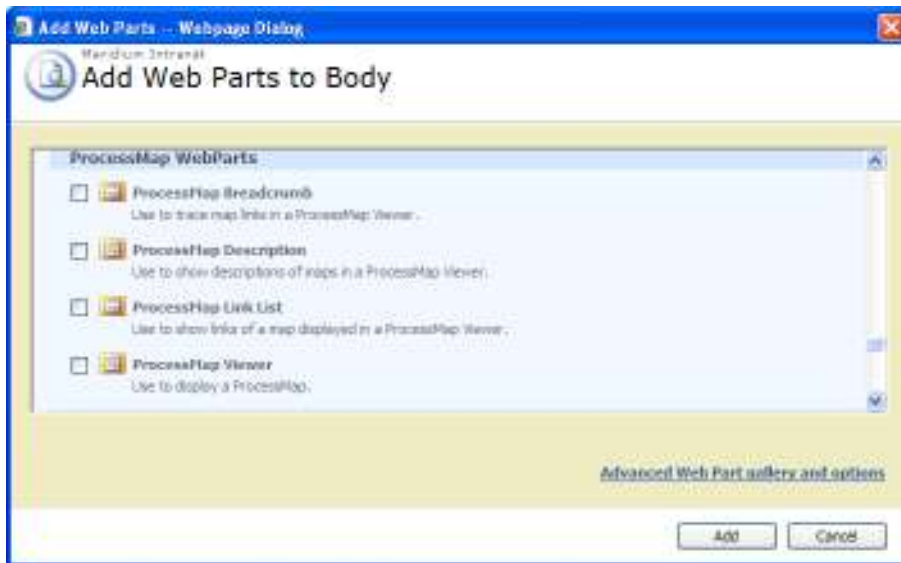
- Create a list

Create a list where the processmaps can be created. The “ProcessMap List” is available under “Custom Lists”, and is created as any other SharePoint list.

- Create a Page

In order to publish the processmaps created in the new ProcessMap List a web part page is required. Simply create a standard “Web Part Page”.

The ProcessMap web parts are available in the standard dialog for adding web parts.



Add the web part “ProcessMap Viewer” and optionally any of the others.

If you add any of “ProcessMap BreadCrumb”, “ProcessMap Description” and “ProcessMap Link List”, these should be connected to the “Viewer” using Connections on the edit web part dropdown.



If there is any processmaps created they can be published by selecting “Select Map” from edit on the “Viewer” web part.

Save the page and note the url.

This url should be inserted into the settings config file as specified in chapter 4.

See PreviewPageUrl

- Configure the editor  
As described in chapter 4.

## 4. Editor configuration

It is possible to configure the behavior of the ProcessMap editor.

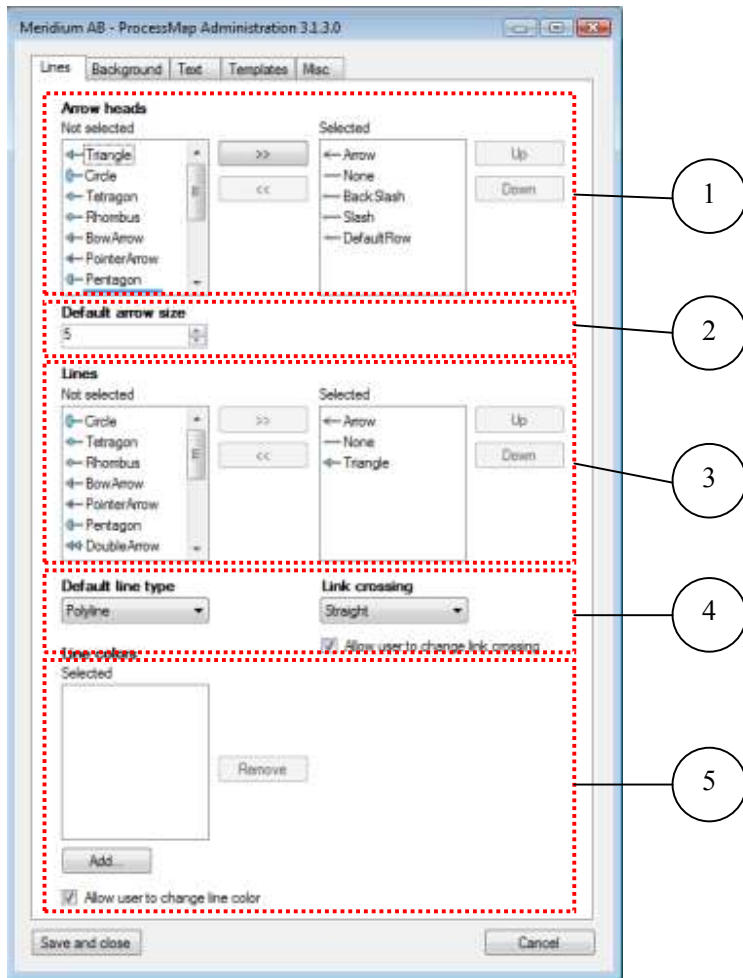
All settings are stored in the file EditorSettings.xml which can be found in the list

<https://host/ProcessMapSymbols/Settings/>.

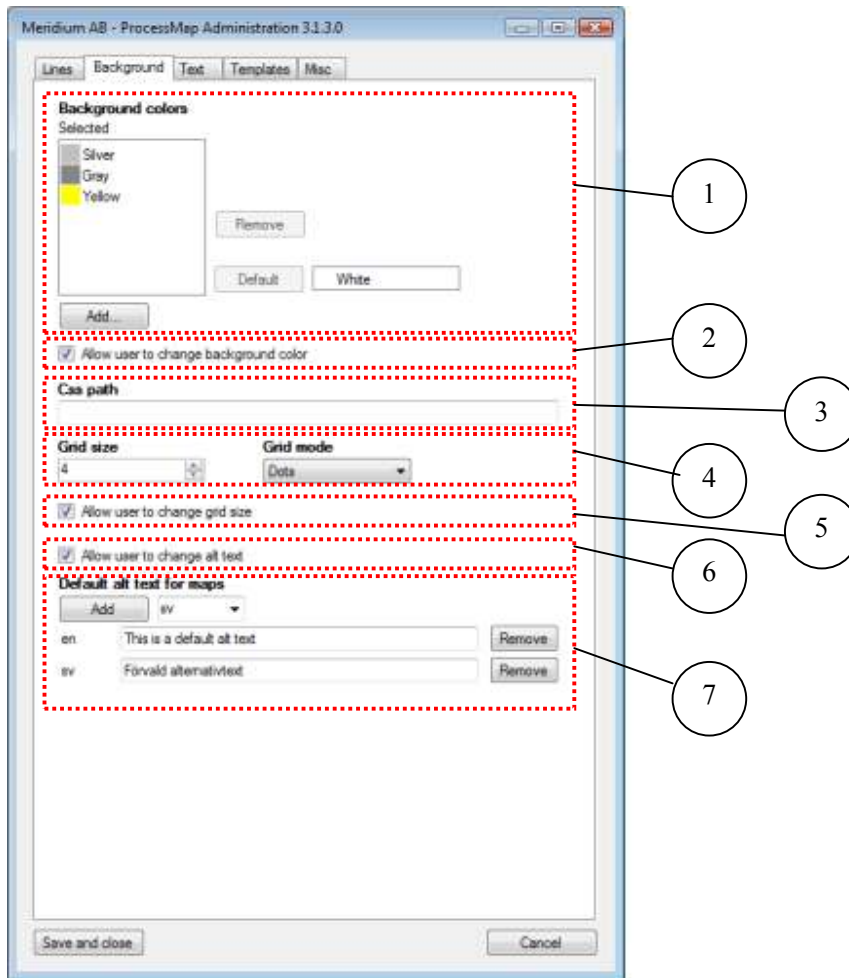
This file will be created the first time ProcessMap runs and is not included in the installation package. It is possible to make changes to the configuration by downloading and editing the file and then upload it to the list again once all changes are made, but the easiest way of making changes is by using the admin client that can be found under Site Collection Administration.

**Warning! The web application needs to be restarted before the changes made to the configuration file takes effect, unless the properties GeneratedDataIsValidSince and Version is also updated.**

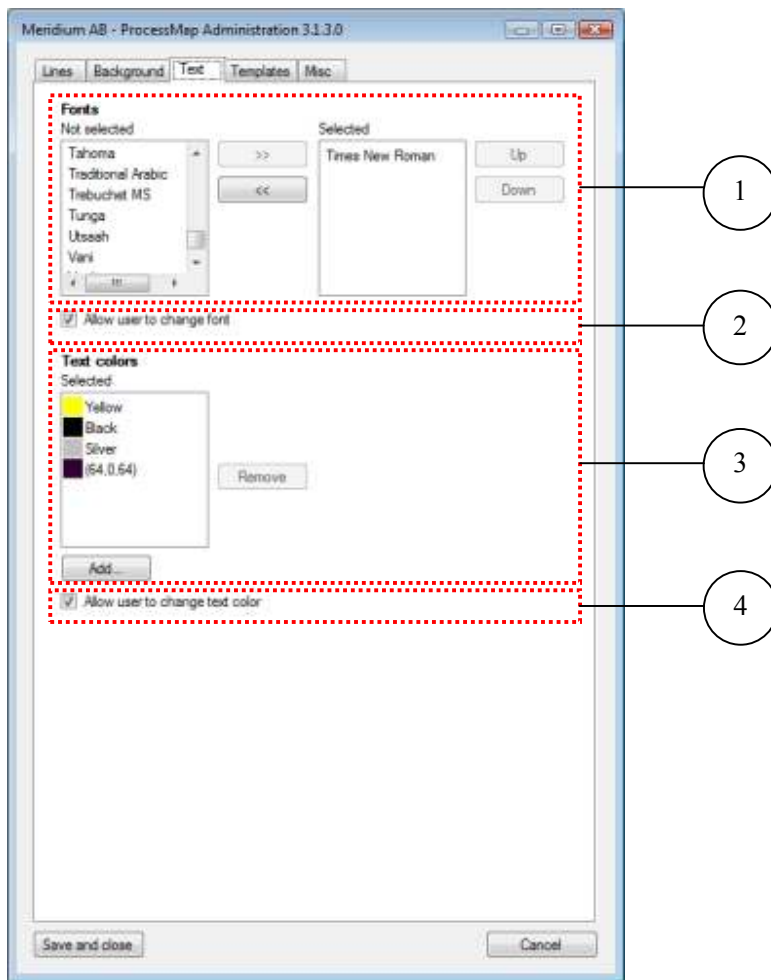
Below are a description of each setting and the name of the element in the configuration xml file.



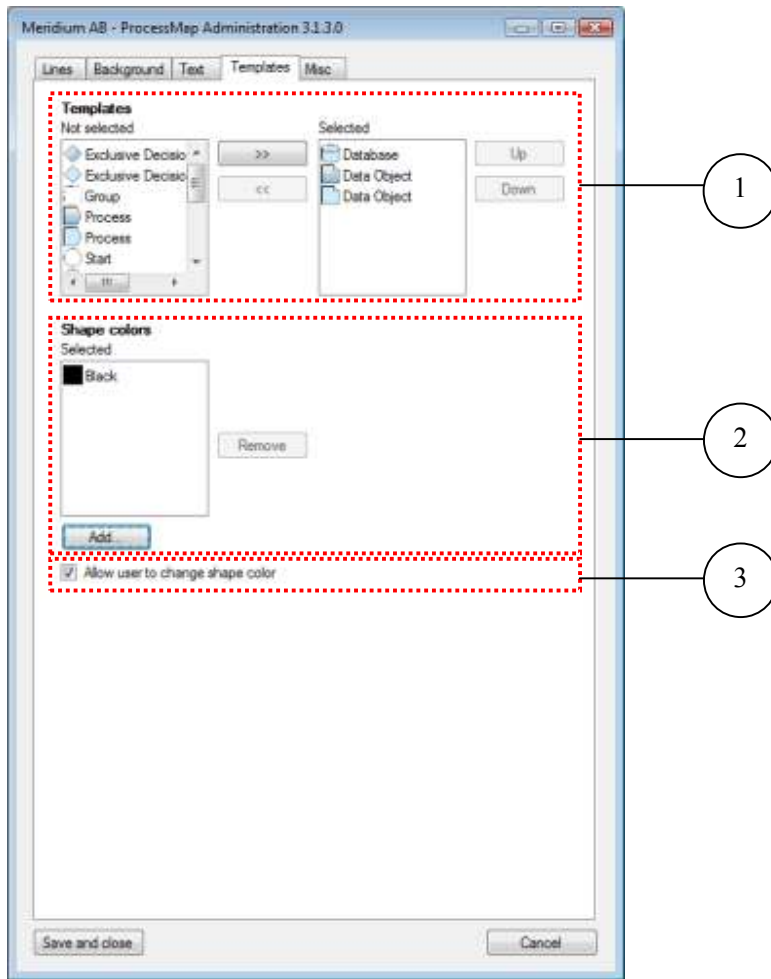
Name	Description
1. ArrowHeads	Controls which arrow heads that can be selected for a connector line. If omitted all arrow head will be available.
2. ArrowSize	The default size on arrows.
3. SelectedLines	This setting controls which connector lines that are available in the ProcessMap editor. See ArrowHeads.
4. DefaultLineType	Default line type. Polyline, Bezier or Cascading.
LinkCrossing	Default connector line crossing. Straight, Arcs or Cut.
ChangeLinkCrossing	If changing line crossing is allowed.
5. LineColors	Controls which colors that can be used for connector lines. See BackgroundColors. If omitted all colors will be available.
ChangeLineColor	If changing line color is allowed.



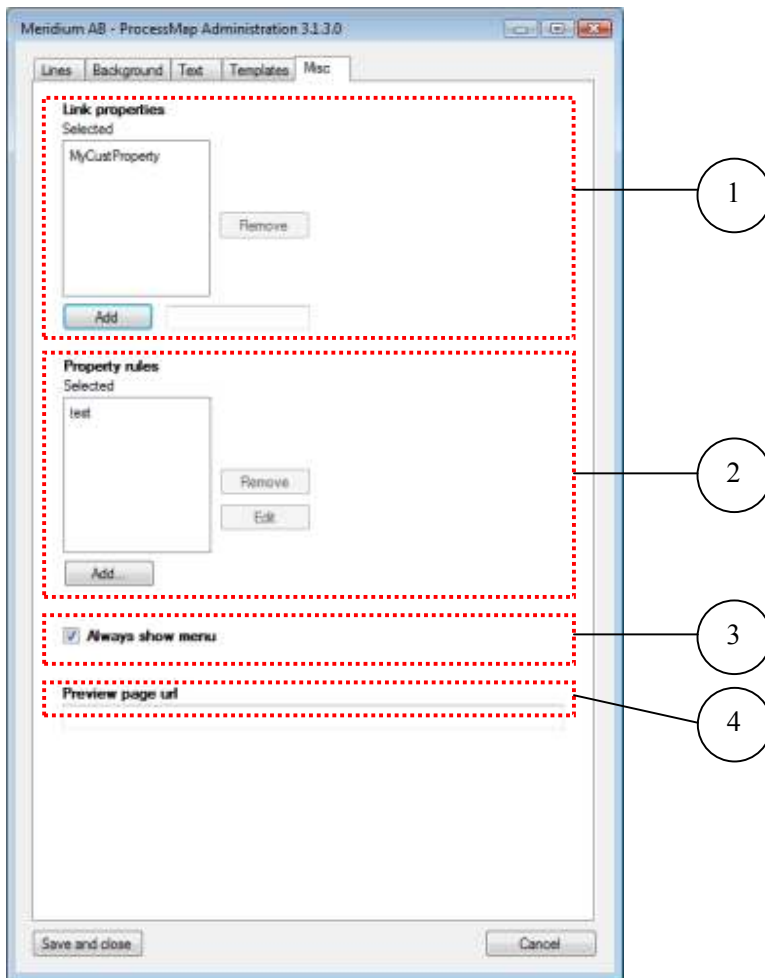
Name	Description
1. BackgroundColors DefaultBackgroundColor	Controls which colors that can be used as background. Colors can be specified as RGB, ARGB or by name.  Specifies the diagrams default background.
2. ChangeBackgroundColor	If changing background color is allowed.
3. CssPath	A custom css file that will be used when displaying the ProcessMaps.
4. GridSize GridMode	Sets the grid size.  Grid mode. None, Hidden, Dots or Lines.
5. ChangeGrid	If changing grid size and grid mode in the editor is allowed.
6. ChangeAltText	If editors should be allowed to change alt-text.
7. AltTextDictionary	A dictionary containing the default alt-text in different languages.



Name	Description
1. SelectedFonts	Which fonts that should be available in the editor. Make sure that the selected fonts exists both on the server and on the client.
2. ChangeFont	If changing font is allowed.
3. TextColors	Which colors that should be available for text in the editor.
4. ChangeTextColor	If changing text color is allowed.



Name	Description
1. SelectedTemplates	This setting controls which symbols that are available in the ProcessMap editor.
2. ShapeColors	Which colors that should be available for symbols in the editor. See BackgroundColors.
3. ChangeShapeColor	Which colors that should be available for text in the editor.



Name	Description
1. LinkProperties	Defines what properties that can be selected for a link.
2. PropertyRules	Rules for the link properties. See chapter 8.
3. MenuAlwaysShow	If the link dropdown should be displayed even if there is only one link.
4. PreviewPageUrl	The page specified is used for previewing processmaps in SharePoint. Only used in the SharePoint version!

## 5. Security configuration modifications

The final step of configuring the ProcessMap for usage includes the following steps:

### 5.1. Server settings

If you experience errors regarding security access please read chapter 6, "Authentication modes and security settings", for further assistance.

### 5.2. Client settings

- .Net framework

For the ProcessMapEditor to work properly each client has to have Microsoft .NET framework 3.5 or later installed.

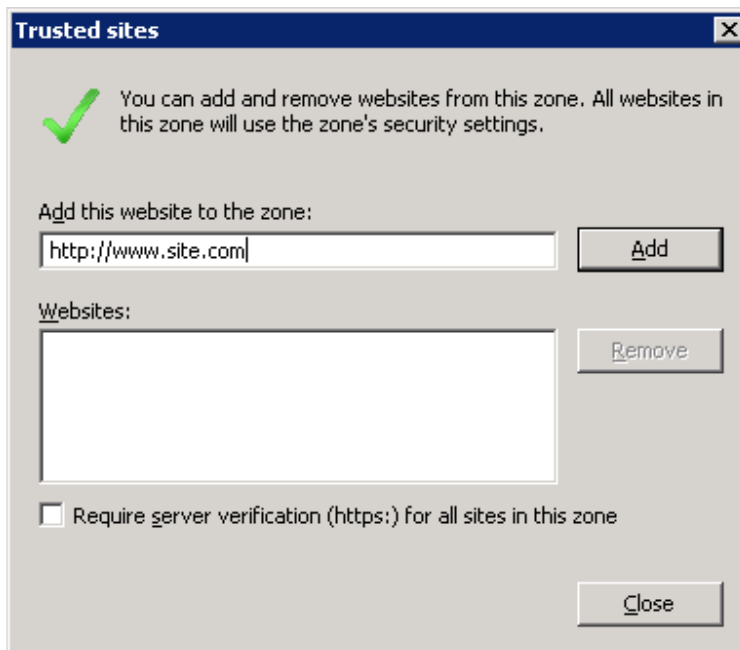
For more information about Microsoft .NET: <http://www.microsoft.com/net>

- Internet Explorer Security Zone

If you use Windows Vista you also have to make sure that the SharePoint server is added as a trusted site. Check the setting in the browser system tray.

Proceed in the following way to add the server url to the Trusted Sites:

Select the internet explorer menu "Tools->Internet Options-> Security->Trusted Sites->Sites".



Enter the server URL in the upper field, click add followed by Close.

The new setting can now be seen in the right bottom corner of Internet Explorer:



## 6. Authentication modes and security settings

To be able to use the ProcessMap editor there are some security considerations that must be made. Since ProcessMap is distributed as a ClickOnce application it must be accessible to the client.

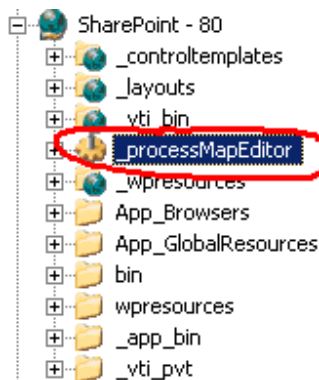
### 6.1. Local windows user is not the same as the SharePoint user

When different user accounts is used on the client and against SharePoint, the ClickOnce application will fail to load.

The solution is to create a new virtual folder in the web site pointing at the folder `_layouts/ProcessMap/Client`. This virtual folder must allow anonymous access and be configured as an own application to prevent SharePoint from blocking access. Though it can use the same application pool if desired.

After this is done the following changes should be made to the web.config.

```
<configuration>
  <se.meridium>
    <ProcessMap>
      <appSettings>
        <add key="ProcessMapEditorBasePath" value="/_processMapEditor"/>
      </appSettings>
    </ProcessMap>
  </se.meridium>
</configuration>
```



In this example we have created the virtual folder `_processMapEditor` in the root of the web site. When the editor is started the user will be asked to login to the ProcessMap server.

### 6.2. License

Everyone needs read access to the license file.

`C:\inetpub\wwwroot\wss\VirtualDirectories\80\meridiumLicense.config`

## 7. Configuration

This section describes the configuration options that are available for ProcessMap. This section also describes all changes made to the web.config file upon installation.

### 7.1. ProcessMapExtension section

The ProcessMap has an own configuration section in the web.config and is located in:

```
configuration/se.meridium/processMapExtension
```

Below follows all configuration keys that exist in the section and a description of the usage and configuration possibilities.

### 7.2. ConfigSections

System settings, do not change.

```
<configSections>
  <sectionGroup name="se.meridium">
    <section name="ProcessMap"
      type=
        "ProcessMap.Server.Configuration.ProcessMapConfigurationSection,
        ProcessMap.Server, Version=1.0.0.0, Culture=neutral,
        PublicKeyToken=01f4840270d48493" />
    </sectionGroup>
  </configSections>
```

### 7.3. Modules and Handlers

System settings for IIS 5 and IIS 6,

```
<httpHandlers>
  <add verb="GET,HEAD" path="*/ProcessMapImage.axd"
    validate="false"
    type="ProcessMap.Server.ProcessMapImageHandler,
    ProcessMap.Server, Version=1.0.0.0, Culture=neutral,
    PublicKeyToken=01f4840270d48493" />
</httpHandlers>
```

For IIS 7

```
<system.webServer>
  <handlers>
    <add name="*/ProcessMapImage.axd_GET,HEAD" verb="GET,HEAD"
      path="*/ProcessMapImage.axd"
      type="ProcessMap.Server.ProcessMapImageHandler,
      ProcessMap.Server, Version=1.0.0.0, Culture=neutral,
      PublicKeyToken=01f4840270d48493" />
    </handlers>
    <validation validateIntegratedModeConfiguration="false" />
  </system.webServer>
```

## 7.4. Application Settings

Settings controlling the application. Should only be changed if the application or site is moved or if additional plugins to the Meridium link editor should be added.

```
<se.meridium>
  <MeridiumLinkEditor>
    <plugins>
      <add typeName="ProcessMap.SharePoint, Version=1.0.0.0,
        Culture=neutral, PublicKeyToken=01f4840270d48493" />
    </plugins>
  </MeridiumLinkEditor>
  <ProcessMap>
    <appSettings>
      <add key="DataServiceFactory"
        value="ProcessMap.SharePoint.Core.DataServiceFactory,
        ProcessMap.SharePoint, Version=1.0.0.0, Culture=neutral,
        PublicKeyToken=01f4840270d48493" />
    </appSettings>
  </ProcessMap>
</se.meridium>
```

## 8. Create new symbols

It is possible to modify existing symbols or creating new ones. Symbols are stored as xml-files on the server in the folder “<webroot>/ProcessMapSymbols/”. New symbols must have a unique id; avoid using id:s between 200 and 299. To make a new symbol appear in the Editor add the id to the “SelectedTemplates” node in the configuration file. See section 4 for more information about changing the configuration.

### 8.1. Symbol xml definition

#### 8.1.1. “ProcessMapItemTemplate” node

This is the parent node that contains all the information that is needed to draw the symbol. It has the following attributes.

- **templateId** (integer) - A unique id for the symbol
- **zLayer** (integer) - The z-index of the symbol. Symbols with high z-index are placed in front of symbols with lower indexes.
- **resizable** (boolean) - Can the symbol be resized?
- **rotate** (boolean) - Can the symbol be rotated?
- **rotateContent** (boolean) - Should the text in the symbol rotate when the symbol is rotated?
- **linkDisabled** (boolean) - Should the possibility to add links to the symbol be disabled?
- **affectRouting** (boolean) – Should the symbol affect routing of lines, i.e. make lines be routed round it. If not it will be possible to draw lines across the symbol.
- **defaultRotation** (integer) - How many degrees the symbol should be rotated by default (clockwise).
- **minimumWidth** (integer) - Minimum width in pixels.
- **minimumHeight** (integer) - Minimum height in pixels.
- **imageAlign** – the behavior of a shapes image. Only needed if the image node is used in a shape. The value can be stretch, fit, center, tile, topLeft, topCenter, topRight, middleLeft, middleRight, bottomLeft, bottomCenter or bottomRight

```
<ProcessMapItemTemplate templateId="206" zLayer="10"
resizable="true" rotate="false" rotateContent="true"
linksDisabled="false" affectRouting="true" defaultRotation="0">
<!-- Other nodes for the symbol's definition are placed here -->
</ProcessMapItemTemplate>
```

#### 8.1.2. “NameDictionary” node

This node contains the symbols names for different languages. The key is a language code and the value is the name.

```
<NameDictionary>
  <item>
    <key><string>en</string></key>
```

```

    <value><string>Database</string></value>
  </item>
  <item>
    <key><string>sv</string></key>
    <value><string>Databas</string></value>
  </item>
</NameDictionary>

```

### 8.1.3. “ShapePlaceholder” node

This node contains the shape of the symbol. The shape is split up in three child nodes. The “outline” node that specifies the outer shape of the symbol, the “decoration” node that specifies the lines within the symbol and the “textarea” node which defines the area where text can be written. The shapes are defined by using the following elements.

- **line** - a single line from one point to another.
- **round-rectangle** - draws a circle from a starting point, width, height and radius.
- **bezier** - draws a curved line through the specified coordinates.
- **arc** - draws a part of a circle from a starting point, a starting angel and a sweep angel.

This is an example of how to create a rectangle with a text area. The fill-mode attribute can be *Winding* or *Alternate* and determines how adjacent areas in the symbol are filled. It will not have any affect in this example since this symbol only has one area.

```

<ShapePlaceholder>
  <Shape id="" fill-mode="Winding">
    <outline>
      <line dash-style="Custom" width="-1">
        <point x="0" y="100" />
        <point x="100" y="100" />
      </line>
      <line dash-style="Custom" width="-1">
        <point x="100" y="100" />
        <point x="100" y="0" />
      </line>
      <line dash-style="Custom" width="-1">
        <point x="100" y="0" />
        <point x="0" y="0" />
      </line>
      <line dash-style="Custom" width="-1">
        <point x="0" y="0" />
        <point x="0" y="100" />
      </line>
    </outline>
    <textarea>
      <line dash-style="Custom" width="-1">
        <point x="1" y="1" />
        <point x="1" y="99" />
      </line>
      <line dash-style="Custom" width="-1">
        <point x="1" y="99" />
        <point x="99" y="99" />
      </line>
      <line dash-style="Custom" width="-1">

```

```

    <point x="99" y="99" />
    <point x="99" y="1" />
  </line>
  <line dash-style="Custom" width="-1">
    <point x="99" y="1" />
    <point x="1" y="1" />
  </line>
</textarea>
<enable-outline>True</enable-outline>
</Shape>
</ShapePlaceholder>

```

It is also possible to add a background image to the symbol. The value of the node should be a base64 encoded image.

```

<image x="0" y="0" width="100" height="100">
  iVBORw0KGgoAAAANSU.....
</image>

```

#### 8.1.4. “Connectors” node

This node contains connection points for the symbol. A connection point is the point where a line can be attached. The connection point is defined as a “ProcessMapItemConnector” node and has an x and y coordinate. ConnectorId and connectorType are not needed.

```

<Connectors>
  <ProcessMapItemConnector connectorId="1" x="0" y="50"
connectorType="Up">
    <Documents />
  </ProcessMapItemConnector>
</Connectors>

```

#### 8.1.5. “Brush” node

This node is used to set a background color on a node.

```

<Brush d2p1:type="SolidBrushTemplate"
xmlns:d2p1="http://www.w3.org/2001/XMLSchema-instance">
  <ColorString>ARGB(255,255,255,255)</ColorString>
</Brush>

```

#### 8.1.6. “Shadow” node

This node specifies if the symbol should have a shadow. It is set to true as default.

```

<Shadow>>false</Shadow>

```

#### 8.1.7. “Aspect Ratio” node

This node is used for giving the symbol a certain aspect ratio.

```
<AspectRatio>0.65</AspectRatio>
```

## 9. Property Rules

It's possible to create rules that can affect the appearance and behavior of the processmap. A rule consists of name, condition and effect. When the processmap is rendered the rules is applied to all symbols and if the conditions are met the effect is applied.

Examples:

```
<PropertyRule name="Links">
  <Condition>Symbol.Links > 0</Condition>
  <Effects>
    <ArrayOfPropertyEffect>
      <PropertyEffect xsi:type="ShadowPropertyEffect" enabled="true" />
    </ArrayOfPropertyEffect>
  </Effects>
</PropertyRule>
```

Any symbol with one or more links will be show with a shadow.

```
<PropertyRule name="HideLinks">
  <Condition>Symbol.Links.IsPropertySet ( Hidden )</Condition>
  <Effects>
    <ArrayOfPropertyEffect>
      <PropertyEffect xsi:type="HideLinkEffect" enabled="true" />
    </ArrayOfPropertyEffect>
  </Effects>
</PropertyRule>
```

Any symbol with the property “**Hidden**” will not be shown in the popup menu when the symbol is clicked.

```
<PropertyRule name="Test">
  <Condition>symbol.links.isPropertySet (Important)</Condition>
  <Effects>
    <ArrayOfPropertyEffect>
      <PropertyEffect xsi:type="ColorPropertyEffect"
        location="SymbolBorder" color="RGB(0,0,0)" />
    </ArrayOfPropertyEffect>
  </Effects>
</PropertyRule>
```

Any symbol with the property “**Important**” will be shown with a black border regardless of what color the symbol normally have.

Location can be either SymbolBackground or SymbolBorder.

## 9.1. Conditions

The condition is specified using a query language that should return a true or false statement. The syntax is described below.

Symbol.ID == 1001  
Is true if the symbol has id 1001

Symbol.Links > 5 AND Symbol.Links.IsPropertySet('My property')  
Is true if the symbol has more than 4 links and any of the links has the property "My property".

```
BNF for PropertyCondition language
http://cui.unige.ch/db-research/Enseignement/analyseinfo/AboutBNF.html

<question> ::= <questionStatement> [ <questionStatementSeparator>
<questionStatement>]

<questionStatement> ::= <propertyCondition> | <propertyConditionNode>

<questionStatementSeparator> ::= <andSeparator>|<orSeparator>

<propertyCondition> ::= <propertyConditionMethod> |
<propertyConditionPredicate>

<propertyConditionNode> ::= ( <questionStatement>
<questionStatementSeparator> <questionStatement> )

<andSeparator> ::= AND | <ampersand> | <ampersand><ampersand>

<orSeparator> ::= OR | <vertical bar> | <vertical bar><vertical bar>

<propertyConditionMethod> ::= Symbol.Links.IsPropertySet(<propertyName> |
<quotedPropertyName>)

<propertyConditionPredicate> ::= <propertyValue> <propertyPredicate>
<propertyValue>

<ampersand> ::= &

<vertical bar> ::= |

<propertyName> ::= string

<quotedPropertyName> ::= <quote><propertyName><quote>

<propertyValue> ::= <numericPropertyValue> | <propertyVariable>

<propertyPredicate> ::= <propertyPredicateEqual> |
<propertyPredicateNotEqual> | <propertyPredicateGreater> |
<propertyPredicateLess> | <propertyPredicateGreaterOrEqual> |
<propertyPredicateLessOrEqual>

<quote> ::= '

<numericPropertyValue> ::= numeric value
```

```
<propertyVariable> ::= <symbol Id variable> | <number of links on symbol>
<propertyPredicateEqual> ::= = | ==
<propertyPredicateNotEqual> ::= != | <>
<propertyPredicateGreater> ::= >
<propertyPredicateLess> ::= <
<propertyPredicateGreaterOrEqual> ::= >=
<propertyPredicateLessOrEqual> ::= <=
<symbol Id variable> ::= Symbol.ID
<number of links on symbol variable> ::= <number of links on symbol
variable alt1> | <number of links on symbol variable alt2>
<number of links on symbol variable alt1> ::= Symbol.Links
<number of links on symbol variable alt2> ::= Symbol.Links.Count
```

## 10. Web Part API

The source code for the web parts shipped with ProcessMap is available for download and can be used as examples of what can be built using the client side API.

See SDK for more information regarding this.

## 11. Logging

Logging is built in to the server parts of ProcessMap. The logging is implemented using log4j and can be activated by modifying the web.config file.

```
<configuration>
  <configSections>
    <section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler,
      log4net" />
  </configSections>
  <log4net>
    <appender name="RollingFileAppender"
      type="log4net.Appender.RollingFileAppender">
      <file value="logs\log4net.txt" />
      <appendToFile value="true" />
      <rollingStyle value="Date" />
      <datePattern value="yyyyMMdd" />
      <maxSizeRollBackups value="10" />
      <layout type="log4net.Layout.PatternLayout">
        <conversionPattern value="%date [%thread] %-5level %logger [%property{NDC}]
          - %message%newline" />
      </layout>
    </appender>
    <root>
      <level value="ALL" />
      <appender-ref ref="RollingFileAppender" />
    </root>
  </log4net>
</configuration>
```