

## SiteSearchASP.NET 5.1

### Enterprise Edition Implementation Guide

---

Congratulations on choosing SiteSearchASP.NET - a fast and extensible solution for adding search functionality to your website or intranet. Integrating over 7 years of research and development, it's the technology used by of hundreds of satisfied customers.

This guide provides information key to implementing the enterprise edition. By following the 5-Step install process you'll have it up and running within minutes.

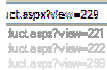
Support and information on other versions can be found at <http://www.sitesearchasp.net/>

#### Enterprise Excellence



##### Scalability and Performance

The indexer can be ran via a console application, greatly increasing scalability over the in-process indexer. The storage repository can also be configured to use a Microsoft SQL Server database, supporting sites with up to 500,000 pages.



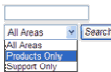
##### Indexer with Dynamic Site Support

The inbuilt indexing engine runs automatically, with support to index dynamic ASP.NET sites. (Including URLs with querystrings - commonly used for database-driven content).



##### Document Indexing Support

Various document types including HTML, PDF, and Microsoft Word / Excel can all be indexed.



##### Area Based Searching

Allow your visitors to search specific areas of your site that you 'tag'. The areas are totally customizable.



##### Feature Rich

Supports Authentication, Intelligent results ranking, Stemming, Robots.txt standard, Adjustable re-indexing schedule & more.

#### The SiteSearchASP.NET Edge

- Proven. We are a pioneer of ASP.NET search technologies.
- Fast to configure & deploy.
- ISP friendly - no server-side app required (depending on your installation choice). Just upload the search page & dll and they intelligently hook onto the ASP.NET engine.
- PDF, DOC, XLS, HTML, and ASP.NET indexing.
- Indexes dynamic ASP.NET sites.
- Totally customizable interface.
- VisualStudio.NET Design Time Support.
- High performance & self maintaining.
- Cross compatible with .NET 2.0, 3.0, 3.5 & 4.

#### Server Requirements

- Microsoft Windows 2000, 2003 or 2008 Server.
- Internet Information Server 5, 6 or 7 (Included with above servers).
- ASP.NET version 2.0, 3.0, 3.5 or 4.
- The Enterprise edition optionally requires SQL Server 2005 or newer (with full text index support).

## Contents

Key Implementation Decisions for the Enterprise Edition .....	3
Integration Process .....	3
Step 1: Obtain current integration package.....	3
Step 2: Configure license .....	4
Step 3. Configure storage repository .....	4
Step 4: Configure indexing mode.....	5
Step 5: Testing the Search .....	6
Advanced Features .....	7
Indexing Multiple domains: .....	7
Filtering by PageArea (or any other key column) .....	7
Associating custom MetaData with Pages and their results .....	8
Enabling support for sites using forms based authentication .....	8
Displaying a search box on each page of your site .....	10
Excluding pages / content from being indexed. ....	10
Support .....	12
Online support resources.....	12
Email support .....	12
Appendix 1: Configuration Options .....	13
Appendix 2: Other Implementation Options .....	17
Express Integrator .....	17
API .....	17
Appendix 3: Document Compatibility.....	20
Appendix 4: Server Permission Requirements.....	21
Appendix 5: Server Trust Requirements .....	22

## Key implementation decisions for the Enterprise Edition

Before beginning with the Enterprise edition, it is important to review and decide on which configuration mode you wish the following options to operate in:

### Storage repository (XML or SQL?)

- XML works great for smaller sites (up to 500 pages/small documents), as it is fast to set up, and the index is cached in the server's memory for fast queries.
- SQL mode moves both the index and job of querying to Microsoft SQL Server. This provides a much higher level of scalability, however is more time intensive / complex to set-up and creates a dependency on SQL Server.

### Indexing mode (In-process or Console application?)

- In Process mode spawns another thread in the w3svc worker process (web server) to handle the reindexing of the site. It's best suited to smaller sites. Also, be aware that if the application pool of your site has restrictions (memory/cpu limits or scheduled recycling enabled) the reindex may never complete.
- The Console based indexing application can be used to improve indexing reliability on larger or document rich sites. The app is configured via a standard .config file and is scheduled as desired using the Windows Task Scheduler.

## Installation Process

### Step 1: Obtain current integration package

- Obtain the latest version of SiteSearchASP.NET from <http://www.sitesearchasp.net/DownloadLatest/>
- Copy the dlls to the bin folder of your site.
- Create a folder named SiteSearchASP.NET in the root of your site. This folder is used for storage of documents whilst the text-extraction engine is ran, and is additionally used to store the data file when ran in XML mode. Ensure the user for which the w3svc process is running under has read/write (full) permission on this folder.

An example search page is included in the integration package, to be used as a starting point for your implementation.

- Copy the example page your site, or extract all search controls from it to your existing search page.

This is our recommended implementation approach as it's easy to set up and totally customizable to match the design of your site. You can also use our express integration option or API (Please see detailed in Appendix 2: Other Implementation Options).

## Step 2: Configure license

No license is required for localhost development (<http://localhost/>), and all features are enabled in such a scenario.

A (currently free) lite license allowing deployment to your site can be requested from our site at <http://www.SiteSearchASP.NET/license/>. The license is immediately emailed to you upon completing the form.

This lite license provides you with all features of the Enterprise URL edition for seven days (including PDF/Office document indexing), after which will revert to the standard lite license functionality.

You can purchase a license for your website by visiting <http://www.SiteSearchASP.NET/purchase/>

- Request a license code from SiteSearchASP.NET.
- Insert license code into either the web.config under an AppSetting name of SiteSearchASP.NET.LicenseCode (note American spelling of license rather than British) or into the SiteSearchASP.NET control's LicenseCode property. A setting on the control will take precedence over the web.config, so please delete the LicenseCode property on the control if you're using the web.config rather than leaving it an empty string.

## Step 3. Configure storage repository

The default storage repository is XML.

Alternatively, to use Microsoft SQL Server as the repository:

- Create a new database, or choose an existing database associated with your site.
- Open the SQL setup scripts from the integration package, and customise the '4.create full text indexes.sql' file, replacing **\*\*\*REPLACE\_WITH\_DATABASE\_NAME\*\*\*** to match your database name. Configure the prefix (e.g. SiteSearch) if you're not using a database dedicated to SiteSearchASP.NET.
- Disable noisewords to ensure accurate search results:
  - If you are using SQL Server 2008 (or newer), please run the following SQL command on your database:

```
ALTER FULLTEXT INDEX ON ***INSERT_DATABASE_NAME***.dbo.Repository
SET STOPLIST OFF;
GO
```

- If you are using SQL Server 2005 server you will need to open the noiseNeu.txt file found in 'Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\FTData' and delete all text.

Then restart the 'SQL Server FullText Search' service.

- Configure the following web.config appSettings OR configure as properties on the search control.
  - SiteSearchASP.NET.DataRepository (with a value of Sql)
  - SiteSearchASP.NET.SqlConnectionString
  - SiteSearchASP.NET.SqlSprocPrefix (optional)
  - SiteSearchASP.NET.SqlTablePrefix (optional)

#### Step 4: Configure indexing mode

The default indexing mode is 'In Process', which requires no additional setup though the recycling settings of the application pool should be verified to ensure they won't kill the indexing thread mid-index. Err on the side of caution and ensure memory, cpu and time based recycling is disabled.

Alternatively - to configure the indexing to be handled by the console app:

- Disable 'In process' indexing by adding an attribute of IndexAutomatically=false to the <Search:Settings control on the search page, or adding it to the web.config as an appSetting named SiteSearchASP.NET.IndexAutomatically with a value of false.
- Copy the 'Console Based Indexer' folder from the integration package to your server.
- Open the application's xml configuration file (SiteSearchASP.NET.IndexerService.exe.config) in a text editor for customization. It is clearly documented with comments, though please ensure close attention is paid to the following nodes:
  - SiteSearchASP.NET.ServerDomainToIndex
  - SiteSearchASP.NET.ServerPathTranslated
  - SiteSearchASP.NET.LicenseCode
  - SiteSearchASP.NET.DataRepository

If you are using the SQL repository, the following properties should also be taken into account:

- SiteSearchASP.NET.SqlConnectionString
- SiteSearchASP.NET.SqlSprocPrefix (Optional)
- SiteSearchASP.NET.SqlTablePrefix (Optional)
- Use the Windows Task Scheduler to configure the indexing frequency of the console based exe. Ensure a user with appropriate permissions is set in the scheduled task's settings.

## Step 5: Testing the Search

You're now ready to run the search. If you are using the console based indexer, we recommend running initially using a dos based prompt (Click Start / All Programs / Accessories / Command Prompt. Right click and select Run as Administrator if required). This will allow you to review the indexing process and confirm it's functioning without a configuration exception.

If you're using in-process indexing simply view the search page in your browser. This will initiate the indexing process and provide feedback should a configuration issue exist.

Generally the implementation process is without any dramas, though different environments occasionally uncover a configuration or technical issue. Our team is experienced at resolving quickly but before contacting support, please enable debug mode and view the indexing process yourself. Usually the issue is quite clear.

To enable debug mode, please add (or set, if pre-existing) an attribute of Debug=True to the <Search:Settings control or if you're using the console based indexer ensure the SiteSearchASP.NET.Debug key is set in the SiteSearchASP.NET.IndexerService.exe.config file.

Should you need to contact our support department, please include a copy of the debug log. This can be saved by copying and pasting from your browser with the in-process indexer or alternatively by piping the output to a text file when using the console based indexer:  
E.g. SiteSearchASP.NET.IndexerService.exe > c:\siteSearchDebug.txt

Basic debugging and status information can also be found in the Event Log of your computer.

Finally - for general permissions issues, please ensure your security is configured according to Appendix 4 of this guide.

**Note: If you are using Windows 7 and encounter an error similar to:**

Security Exception (System.Security.SecurityException: Request for the permission of type 'System.Web.AspNetHostingPermission, System, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089' failed.)

**then please open IIS, and right click your application pool, selecting 'Advanced Settings...'. Then under the 'Process Model' section change the 'Load User Profile' value to True. We are working to find an easier solution to this occasional issue.**

## Advanced Features

### Indexing Multiple domains:

The ServerDomainToIndex property accepts a comma separated list of urls the indexer will start at. It's pretty flexible, allowing you to optionally specify the protocol and port.

The format for the property supports a wider range of urls, e.g. www.example.com, www.example.com/mystartpage.aspx, http://www.example.com, http://www.example.com:82/

Just make sure you have purchased a license compatible across ALL the domains you're using or it will revert to Lite mode (with a limit of 25 pages).

You can use the ResultsFilter property on the control (specifying the PageHost value) to have results limited to particular domains. Further information on the syntax is found below in 'Filtering by PageArea (or any other key column)'.

Please be aware of the following when indexing multiple domains:

- If the indexing is ran 'In-process' rather than via the console application, one site will need to be defined as the primary 'indexer'. Other sites would have the IndexAutomatically property set false, and use the ServerPathTranslated property to link to the primary data file if using the XML repository, or alternatively use the SqlConnectionString for the SQL implementation.
- All existing control properties/web.config settings (including authentication, indexPDF setting, etc) will apply globally to all domains.
- Area based searching will not be specific to each domain.
- EnableLegacyIndexing and any other non-relevant settings will not be supported.

### Filtering by PageArea (or any other key column)

In addition to the existing PageArea functionality (with the drop-down selector), you can hard-code a filter on a <Search:Settings control.

This provides great flexibility to group results as you can add multiple <Search:Settings controls to your page, with each instance can be filtered for specific results. E.g PDFs, Press releases, etc,

The syntax is:

```
ResultsFilter="PageArea='about'"
```

Be aware this is directly injected into the SQL where clause, thus don't use user-passed values unless you've vetted them.

If using the XML data repository you can also filter based on any custom MetaData (see next section).

## Associating custom MetaData with Pages and their results

You can define custom meta data on each page, which can be rendered with the search results.

The syntax for the tag required on each page is:

```
<meta name="sitesearchasp.net.metadata:{fieldname}" content="{fieldvalue}">
```

E.g. <meta name="sitesearchasp.net.metadata:Price" content="\$200">

This can be accessed within your datagrid or repeater's itemtemplate with the following syntax:

```
<%# Search.Render(Container,"metadata:{fieldname}") %>
```

E.g. Price: <%# Search.Render(Container,"metadata:Price") %>

## Enabling support for sites using forms based authentication

Forms based authentication is supported though your site's global.asax must be configured to allow the indexer to be authenticated so it can access your pages.

The following code is provided as an example and may need to be modified according to your specific requirements. A more in-depth second example is also provided below.

Basic code required in Global.asax file:

```
<script language="C#" runat="server">
    void Application_BeginRequest(Object sender, EventArgs E) {

        // Allow SiteSearchASP.NET to bypass authentication,
        // using the FormsAuthentication.SetAuthCookie
        // method. The security check confirms both that the
        // request is originating from the server's IP address
        // and the SiteSearchASP.NET user agent is correct.

        if ((Request.ServerVariables["REMOTE_ADDR"] ==
            Request.ServerVariables["LOCAL_ADDR"]) &&
            (Request.ServerVariables["HTTP_USER_AGENT"] ==
```

```

        "SiteSearchASP.NET"))
    {
        FormsAuthentication.SetAuthCookie("", false);
    }
}
</script>

```

You may need to adapt the above code to your specific login routine. E.G:

```

<script language="C#" runat="server">
    void Application_BeginRequest(Object sender, EventArgs E) {

        // Allow SiteSearchASP.NET to bypass authentication,
        // using the FormsAuthentication.SetAuthCookie
        // method. The security check confirms both that the
        // request is originating from the server's IP address
        // and the SiteSearchASP.NET user agent is correct.

        if (
            (Request.ServerVariables["REMOTE_ADDR"] ==
             Request.ServerVariables["LOCAL_ADDR"]) &&
            (Request.ServerVariables["HTTP_USER_AGENT"] == "SiteSearchASP.NET"))
        {
            string email = "myuser@mydomain.com";
            FormsAuthentication.SetAuthCookie(email, false);

            string[] userData =
                {
                    "2712",
                    "user name",
                    email
                };

            FormsAuthenticationTicket ticket =
                new FormsAuthenticationTicket(1, email, DateTime.Now,
                DateTime.Now.AddHours(1), false, string.Join(",", userData));

            FormsIdentity formsIdentity = new FormsIdentity(ticket);
            HttpContext.Current.User = new

                System.Security.Principal.GenericPrincipal(formsIdentity, null);

        }
    }
}
</script>

```

## Displaying a search box on each page of your site

You can define a search box on each page of your site using the following code.

```
<form action="search.aspx" method="GET">
  <input type="text" name="SearchTerm">
  <input type="submit" value="Search">
</form>
```

If you need to add this to a page with a pre-existing form, please use the following code instead:

```
<script language="JavaScript" >

function searchRedirect(e) {
  var keyPressed;
  if (e != null)
    keyPressed = (window.event) ? window.event.keyCode : e.which;
  if((e == null) || (keyPressed == 13)) {
    searchUrl = "/search.aspx?SearchTerm=" +
      escape(document.forms[0].SearchTerm.value) + "&SearchType=All";
    document.forms[0].disabled=true;
    self.location = searchUrl
  }
}

</script>

<input type="text" name="SearchTerm" onKeyPress="searchRedirect(event)">
<input type="submit" value="Search" onClick="searchRedirect(null);
return false">
```

## Excluding pages / content from being indexed.

Specific pages or page fragments of your site can be excluded from the search results using either of the three following ways:

### 1. Individual Pages

Individual pages can be excluded from being indexed by adding the following tag to the desired page.

```
<meta name="robots" content="NONE|NOINDEX|NOFOLLOW">
```

The supported content attribute values are either:

NONE: The page will not be indexed, and the links within won't be followed or indexed.

NOINDEX: The page will not be indexed, however the links within will be followed and indexed.

NOFOLLOW: The page will be indexed, however the links within will be not be followed or indexed.

## **2. Page Fragments (with links still being followed)**

Page fragments wrapped in <NOINDEX></NOINDEX> tags will be excluded from being indexed. This is typically of benefit hiding repeated items such as navigation menus from the search results.

Please however note that links within the <NOINDEX></NOINDEX> tags will still be followed and indexed.

## **3. Page Fragments (with links not being followed)**

Page fragments wrapped in <SiteSearchASP.NET.NOINDEX></SiteSearchASP.NET.NOINDEX> tags will be excluded from being indexed. Unlike the NOINDEX tags, this includes all content, including titles, meta tags, and links.

## **4. Robots.txt file**

By using the industry standard robots.txt file placed in the root of the website. You can define custom metadata on each page, which can be rendered with the search results.

An example robots.txt file would be formatted as follows:

```
User-agent: *      # All robot search engines
Disallow: /documentationFolder/
Disallow: /myOtherPage.aspx
```

## Support

### Online support resources

Please visit the support area of our site for tips, tricks and first level support:  
<http://www.sitesearchasp.net/support/>

### Email support

We're dedicated to ensuring our customers are satisfied with their purchase and integrate SiteSearchASP.NET successfully. If you have any implementation questions or would like to report an issue please use our online form at: <http://www.sitesearchasp.net/contact/>

To ensure a quick response, please provide as much information as possible, preferably including a link to your site.

## Appendix 1: Configuration Options

These configuration options can be set directly on the <Search:Settings control on your search page OR alternatively they can be set in your Web.Config, prefixed with 'SiteSearchASP.NET.' in your appSettings container. If using the Express Integrator you will need to set each property in the web.config rather than on the control.

**If a property exists on the <Search:Settings control it will override the value set in the web.config. This includes empty properties. Please bare this in mind as you may need to delete unwanted properties from your search page.**

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="SiteSearchASP.NET.LicenseCode" value=""/>
  </appSettings>
</configuration>
```

Property	Description
bool <b>Debug</b>	Setting debug to true will return a verbose output of the indexing process to the browser. This will allow you to debug problems if content is not being indexed. Please note when using in-process indexing that if you close your browser or press stop prior to the indexing completing, your index will be saved with only partial data. The default of this value is false.
string <b>HighlightFoundTermPrefix</b>	This prefix will be prefixed to highlighted words within the search results. This can be any html tag. Eg: "<b>" or "<span style='background-color:#FFCC33'>"
string <b>HighlightFoundTermSuffix</b>	This suffix will be appended to highlighted words within the search results. This can be any html tag. Eg: "</b>" or "</span>"
bool <b>MatchWholeWordOnly</b>	Setting this property to true will return only matches where a complete word is found that exactly matches the search term(s) (as opposed to the partial matching of characters within words.) The default value is false.

Note: The MatchWholeWordOnly property is locked to True if EnableStemming is set to True. This is by design, as it was originally implemented to allow a search for 'fish'

to find 'fishing'. The stemming feature now provides this functionality, as well as the reverse ('fishing' when searching for 'fish' or 'fished').

- bool EnableStemming** For licenses of Standard or Professional URL, Server and Enterprise Editions this allows you to configure whether search terms are stemmed. This reduces the search terms to their stem, for example a search for 'computing' would be internally reduced to 'compute' and return results containing 'computers', 'computer', etc. This provides a better selection of results, though is only recommended for sites in the english language. The default of this value is true for purchased licenses, and false for Lite.
- bool EnableHttpsIndexing** For licenses of Standard, Professional URL, Server and Enterprise Editions this allows you to configure the site is indexed over HTTPS (port 443). The default value is false.
- bool IgnoreRobotsTxtFile** Configures whether indexer ignores the rules in the robots.txt file. The default value is false.
- bool IndexAnchorTagText** Configures whether indexer indexes the text within anchor tags. The default value is false.
- bool IndexOfficeFiles** For licenses of the Professional URL/Server/Enterprise Editions this allows you to configure whether Doc/XLS/Powerpoint files should be indexed. The default of this value is true.
- bool IndexPDFs** For licenses of the Professional URL/Server/Enterprise Editions this allows you to configure whether PDF files should be indexed. The default of this value is true.
- bool EnablePdfTitleParsingFromMetadata** When using the optional advanced PDF parsing library this property allows you to configure whether the title within PDFs metadata is parsed/used. The default value is false.
- bool IndexTextFiles** For licenses of the Professional URL/Server/Enterprise Editions this allows you to configure whether Txt files should be indexed. The default of this value is true.
- int IndexAfterHours** For licenses of the Professional URL/Server/Enterprise Editions this allows you to specify the frequency that the indexing engine to run.  
 We recommend this is set between 1 day (24) and 7 days (168). For all other licenses, this re-indexing will be performed on a weekly basis. The default of this value is 7 days.  
 Note if you are using the console based indexer, this doesn't apply, as the indexer is scheduled using the

Windows Task Scheduler.

string **ResultsControl** Specifies the ID of the DataGrid, DataList, or Repeater of the control you wish to bind the search results to. This is a required property on your search page's <Search:Settings control.

string **SearchTermControl** Specifies the ID of the ASP.NET HtmlInputText control used to specify the search term entered by the user. This is a required property on your search page's <Search:Settings control.

string **SearchTypeControl** Specifies the ID of the ASP.NET HtmlSelect control used to specify the search type entered by the user. This is a required property on your search page's <Search:Settings control.

string **ServerDomainToIndex** Optionally you may specify the domain you wish SiteSearchASP.NET to index. Typically used with when your site needs to index a specific domain or port due to Load-Balancing/Firewall/NAT limitations. Eg: 127.0.0.1:802 or 10.0.0.2.

Multiple domains are supported providing they are comma separated.

The format for the property supports a wider range of urls, e.g. www.example.com, www.example.com/mystartpage.aspx, http://www.example.com, http://www.example.com:82/

Just make sure you have generated a license compatible across ALL the domains you're using or it will revert to Lite mode (with a limit of 25 pages).

bool **GenerateRanking** Our unique ranking algorithm ensures the most important pages appear at the top of the search results. Disabling ranking will provide a significant performance increase (300%). The default value is true.

bool **EnableBasicAuthenticationPassThrough** When using in-process indexing, setting this as true will automatically forward across the user details of the currently authenticated user (if Basic Authentication is utilised) It is important to note that enabling this could cause issues in a website that is personalized to each user (as the search results would reflect the site content personalized to another user). Support for forms based authentication is not provided at this point.

Alternatively the AuthenticationUser/Pass/Domain properties below can be used (Which will automatically

set this property to false).

```
AuthenticationUser="<string>"
AuthenticationPass="<string>"
AuthenticationDomain="<string>"
```

These specify the username and password that the indexing engine should use when requesting the site. These should be set if your site uses Basic or Integrated Windows Authentication.

The AuthenticationDomain property should only be specified when Integrated Windows Authentication is used. This is optional in such scenario.

**bool IndexDomainsIndependently** Certain deployments will be on sites that share the same codebase (On one IIS site) across various domains, but output different content depending on the which domain the site is being accessed on.

Configuring this as true allows each independent domain to have its own index of site content.

Configuring this as true is only compatible with the Professional Server and Enterprise Server Edition licenses. The default setting is false.

**bool EnableLegacyIndexing** When using in-process indexing in environments where the ASP.NET trust level has been set as Medium, (Please see the Server requirements page) you can run SiteSearchASP.NET in legacy indexing mode by adding an attribute of EnableLegacyIndexing="true" to the <Search:Settings tag. This will index the text of every aspx file in the same folder of your search page (Files within subfolders and files not ending with aspx are not indexed). Please bare in mind this is a very simple search that extracts the text from the physical ASP.NET page, hence no dynamic content or programmatic content is indexed. You will need to ensure your ASP.NET user account has the appropriate access to the search folder.

The default setting is false.

## Appendix 2: Other Implementation Options

### Express Integrator

This option allows you to add search by placing a control (2 lines of code) on your ASP.NET page. Search results are shown directly within the page using advanced AJAX like overlaying technology.

**This is only recommended for basic sites as the design is not customizable and CSS markup from your site can interfere with the user-interface of the results overlay. We do not support CSS issues.**

To implement, add the following line to the top of your ASP.NET page.

```
<%@ Register TagPrefix="Search" Namespace="SiteSearchASP.NET.Interface"
assembly="SiteSearchASP.NET" %>
```

Then add the following tag on your page(s) where you desire the search field to appear.

```
<Search:ExpressIntegrator runat=server />
```

The Express Integrator requires that you specify the configuration options via the web.config file rather than on the control. The options are listed in to Appendix 1: Configuration Options . Please don't forget to enter your LicenseCode.

Please ensure each property is prefixed with 'SiteSearchASP.NET.' in your appSettings container. E.g.:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="SiteSearchASP.NET.LicenseCode" value=""/>
  </appSettings>
</configuration>
```

### API

In most cases the API should only be used to supplement an implementation of a customized or express integration.

**We don't recommend a implementation purely using the API, as SiteSearchASP.NET has not been optimized for such scenario and you will not be able to set all the configuration properties you'll likely desire.**

Potential uses for the API:

1. Automatically forcing a re-index when the site content is modified by a CMS, by directly calling the indexer.
2. To allow other systems, eg webservices to access the search index.

If you are hoping to pre-process the results before binding to the page (for example to add or remove data) our recommendation is to use the main integration option, access the `SearchSettings.GetSearchResultsDataView()` method in the `Page_Load` event, modify, then bind to your page. This assumes that you assign an ID of `SearchSettings` to the `Search:Settings` control. In this scenario the `ResultsControl` property is ignored.

#### **Negative aspects of using the API:**

Not all SiteSearchASP.NET configuration options are available, hence you won't be able to configure authentication, specify what document types you wish to index, etc.

You will receive less assistance messages, for example information that your file system permissions are incorrect, that you don't have a default page, and many more. We recommend if you encounter any issues using the API, you try using the main integration option as that will generally identify what the issue may be.

Purely using the API will mean you'll need to programmatically call the `ReIndexSite` method when you wish the index to be updated.

Search terms will no longer be highlighted within the results.

#### **Integration Steps**

Use the following methods to interact with the indexer and querier.

##### **Indexer**

Constructor:

```
SiteSearchASP.NET.API.Indexer(string licenseCode)
```

Method:

```
ReIndexSite(bool indexInDebugMode, string dataFolderLocation, string urlToBeginIndexingAt)
```

An SiteSearchASP.NET.API.IndexInProgressException will be thrown if indexing is already in progress.

## Querier

Constructor:

```
SiteSearchASP.NET.API.Querier(string licenseCode, string dataFolderLocation)
```

Methods:

```
DataView Query(string query)

DataView Query(string query, string type, string area, bool
generateSmartSummary, bool generateRanking)

DataView Query(string query, string type, string area, bool searchPageTitles,
bool searchPageText, bool searchMetaKeywords, bool searchMetaDescription, bool
generateSmartSummary, bool generateRanking, bool matchWholeWordOnly)
```

## Notes

**dataFolderLocation** should be set to a path matching your environment, but in the format of: "C:\inetpub\wwwroot\mysite\sitesearchasp.net\".

**urlToBeginIndexingAt** should be set to a path matching your environment, but in the format of: "http://localhost/mysite/".

**type** may be set as either exact, all or any.

**area** should be left as an empty string for searching all areas.

## Appendix 3: Document Compatibility

PDF and Office (Word doc, PowerPoint ppt, and Excel xls) files are all indexable by the crawler.

If Office 2007 files aren't being indexed correctly, please install the latest Microsoft IFilters on your server. These are available free from Microsoft at:

<http://www.microsoft.com/downloads/details.aspx?familyid=60c92a37-719c-4077-b5c6-cac34f4227cc&displaylang=en#Overview>

Certain PDF files will not be indexed using the inbuilt indexer - due to their mark-up and size. These may appear in the index as either 'Document text unavailable. Too large for indexing.' or 'Document text unavailable. Unable to parse.'. They will still be searchable based on the text within the link pointing to them and the file name of the PDF.

Some PDF files will not be able to be indexed due to their security or encryption settings.

### **Please note**

PDF indexing can be improved using an add-on PDF indexer we have available. This is not distributed with SiteSearchASP.NET as it is 20MB in size. Please contact us with your license code to receive to this add-on.

## Appendix 4: Server Permission Requirements

The Windows ASPNET user must be granted read/write access to the SiteSearchASP.NET data folder. Use Windows Explorer to grant full access for the Windows ASPNET user to the 'sitesearchasp.net' folder. This account is typically named {MACHINE}\\ASPNET on IIS 5 or Network Service on IIS 6.

Your ISP or System Administrator may need to assist with this if you do not have such access to the server.

If you are developing on Windows XP, you will need to disable Simple File Sharing to make this change. This can be done by opening Windows Explorer, selecting Folder Options from the Tools Menu, clicking the View tab, and un-ticking the 'Use Simple File Sharing (Recommended)' check box at the bottom of the Advanced settings list. You will then be able to right click the SiteSearchASP.NET folder, select properties, click the security tab, and add the relevant user with 'Full Control' privileges.

## Appendix 5: Server Trust Requirements

For all features of SiteSearchASP.NET to be enabled, the ASP.NET trust level must remain at its default value - Full.

Certain hosting providers, particularly in shared environments, lower this setting to High or Medium. This prevents access to critical .NET framework components including the WebRequest object.

This can be overcome if your hosting provider changes their ASP.NET security policy file to allow WebPermission, or enters the url of your website in the policy file's <ConnectAccess> <URI uri attribute in the format of a regular expression (with a trailing .\*). For further information please view this Wrox article: <http://www.wrox.com/WileyCDA/Section/id-291738.html>

Most hosting providers are flexible enough to make this change.

If this change can't be made, you can run SiteSearchASP.NET in legacy indexing mode by adding an attribute of EnableLegacyIndexing="true" to the <Search:Settings tag. This will index the text of every aspx file in the same folder of your search page (Files within subfolders and files not ending with aspx are not indexed). Please bare in mind this is a very simple search that extracts the text from the physical ASP.NET page, hence no dynamic content or programmatic content is indexed. You will need to ensure your ASP.NET user account has the appropriate access to the search folder. Office document indexing, or using our add-on PDF indexer requires the trust level to be set as Full due to system interop calls.

Note: Changes to the machine.config generally require a restart of the W3CSVC (World Wide Web Publishing) service.