

Coveo Enterprise Search 6.0

Sitecore Connector

Coveo's *Sitecore* connector allows users to index *Sitecore CMS* 5.3 and 6 installation content as well as *Sitecore's* Web and Master content. All the site items, the referenced media library items and security elements can be indexed using any *Sitecore* account.

Features

The following details the features available in the *Sitecore* connector:

- Site item metadata indexing; thus allowing the user to customize CES based on that metadata (Custom Fields, Sort, Refine, Group by, etc.).
- Media Library item indexing.
- **Security:** CES can index security from *Sitecore's* permission model.
- **Live indexing:** The connector periodically queries *Sitecore* for the latest edits, keeping the index content up to date.

Requirements

The *Sitecore* connector supports *Sitecore CMS* 5.3. and 6 installations.

Installation

▶ How to Install Sitecore's Web Service

In order to retrieve information from *Sitecore CMS*, import a special package in the *Sitecore* installation. These steps are required when installing CES for the first time and might need to be repeated when upgrading from an existing installation. The *Sitecore* connector will not work if the wrong version of the package is installed on *Sitecore*. Following the installation and upgrade of CES, consult the CES Console for any error messages.

To import the special package, perform the following procedure:

1. Start the *Sitecore Installation Wizard*.



2. Depending on the *Sitecore* version used, upload *Sitecore5.3CoveoWebService.zip* or *Sitecore6.0CoveoWebService.zip*. The package is located in the <Drive_Letter>:\Program Files\Coveo Enterprise Search 6\bin folder.
3. Install the package by following the instructions of the *Installation Wizard*.
4. Restart both *Sitecore* client and server at the end of the installation process.

Note: If an **Access Denied** error is displayed on the *bin_install* folder while installing the package, verify the security permissions for *ASP.NET* in the *bin* folder of *Sitecore* or install the files manually.

Configuration

▶ How to Configure a User Identity Based (Optional)

By default, the connector uses the *extranet\Anonymous* account to retrieve content from *Sitecore*. Depending on the security configuration, this user might not have access to all *Sitecore* items. You can configure a CES user identity to use another *Sitecore* account that has more access privileges. Only *Read* access is required in order to retrieve a content item.

Note: The user does not need to be part of the *Sitecore* administrators. When specifying a user, it is recommended, but not mandatory, to specify a user part of the same domain than the source crawled.

1. In the Administration Tool, access the **Administrators** page (Configuration > Security).
2. In the left navigation pane, click **User Identities**. The **User Identities** page is displayed.
3. Click . The **Modify User Identity** page is displayed.
4. In the **Name**, **User** and **Password** fields, enter the credentials of the account including the Sitecore domain name.

5. Click  Save .

Security - User...

Name Full read access user

User extranet\PowerReadUser

Password

Option Support basic authentication

Client certificate Use client certificate

Store Personal


Certificate [Dropdown]

Used By

▶ **How to Configure the Security Provider (Optional)**

Note: This procedure is required to index the security permissions and use the Search Interface integration within a *Sitecore* Web site.

Note: If your *Sitecore* security is handled by the Active Directory module (*Sitecore* 6 only), refer to the [Working with the Active Directory Module \(Sitecore 6 only\)](#) section.

1. Access the **Administrators** page (Configuration > Security).
2. In the left navigation pane, click **Security Providers**. The **Security Providers** page is displayed.
3. Click  Add to create a new security provider.
4. Enter the following information in the appropriate fields:

Field	Description
Name	Sitecore Security Provider (or any significant name)
DLL Path	<Drive_Letter>:\Program Files\Coveo Enterprise Search 6\Bin\Coveo.CES.CustomCrawlersSecurityProvider.dll
User Identity	Optional. By default the <i>extranet\Anonymous</i> user has access to the <i>Roles</i> and <i>Users</i> definitions.
Parameters	<i>AssemblyPath="<Drive_Letter>:\Program Files\Coveo Enterprise Search 6\Bin\Coveo.CES.CustomCrawlers.Sitecore.dll";WebServiceUrl="http://SitecoreWebSite"*</i>
Options	Select the following checkboxes: Support access list and Support expand group .

* This is the URL of the *Sitecore* installation.

← Security - Security...

Name	Sitecore Security Provider
DLL Path	c:\Program Files\Coveo Enterprise Search 5\Bin\Cov
User Identity	(none) ? Add Edit Manage user identities
Parameters	AssemblyPath="c:\Program Files\Coveo Enterprise S
SAML Redirection URI	
SAML Artifact Resolver URI	
SAML Logout URI	
SAML Signature Certificate File Name	
SAML Signature Certificate Password	
SAML Artifact Argument Name	SAMLart
SAML Post Response Argument Name	SAMLResponse
SAML Back Target Argument Name	TARGET
Authorization Cache Timeout	3600
Authentication Cookie Expiration	1 ?
Option	<input type="checkbox"/> Do not block exceptions <input type="checkbox"/> Require authorization <input checked="" type="checkbox"/> Support access list <input checked="" type="checkbox"/> Support expand group <input type="checkbox"/> Support expand user
Used By	

▶ How to Index a Sitecore Source

A *Sitecore* source only targets one *Sitecore* Web site. It is recommended to have one source for each Web site to index. All the available Web sites of a single *Sitecore* installation are defined under the **<sites>** node in *Sitecore's web.config* file:

```
<sites>
  ...
  <site name="danish" hostName="da.printers" language="da-DK" virtualFolder="/"
  <site name="german" hostName="de.printers" language="de-DE" virtualFolder="/"
  <site name="english" hostName="en.printers" language="en" virtualFolder="/"
  <site name="british" hostName="gb.printers" language="en-GB" virtualFolder="/"
  <site name="website" virtualFolder="/" physicalFolder="/"
  ...
</sites>
```

1. In the Administration Tool, access the **Sources and Collections** page (Index > Sources and Collections).
2. In the **Sources** section, click **Add**. The **Add Source** page is displayed.
3. Enter the appropriate values of the selected *Sitecore* site.

Field	Description	Example
Name	Any descriptive name.	English Printers site
Source Type	The connector used by this source.	Sitecore
Addresses	Base address of the Sitecore installation.	Addresses are in the form of <code>http://SitecoreWebsite</code> . The connector supports both <code>http</code> and <code>https</code> .
Target Site*	The name of the site (from the <i>web.config</i> file) to index.	The default value is website .
Content Start Path*	The starting point of indexing in the <i>Sitecore</i> content tree.	The default value corresponds to the content start item defined for the selected site. In the <i>web.config</i> file, it is represented by the rootPath and startItem attributes of the selected site. It is possible to specify one or more starting path by separating multiple root nodes with the “;” character: <i>/sitecore/content/home/MyNewRootNode;/sitecore/content/Resources</i>
Database	The name of the database to index.	The default value corresponds to the database defined for the target site. For example, this value can be set to <i>master</i> in order to index the non-published content of the target site.
Content Admin*	The <i>Windows</i> user that can see all indexed documents. By default, it is impossible to see the indexed content of a document when a source is using a security provider for indexing the permissions. Use this parameter if the source has a security provider and you need to see the document’s indexed content in the Index Browser .	<i>NTDomainName\UserName</i>
Include Media Library	Indicates that all the content of the media library will be indexed.	Default is True . This has the same effect as adding the <i>/Sitecore/content/media library</i> to the <i>Content Start Path</i> value. If media items are referenced by content items that are indexed, these media items are also indexed even if the value of this parameter is False .
Mapping File*	Defines how the connector handles metadata. For more information, refer to the Mapping File section.	The path of a valid configuration file.

Body Format	Specifies how the html cached version of an indexed document is saved.	<p>Web Page (default) stores the html version of an item.</p> <p>Meta Information only stores <i>Sitecore</i> fields and values. The second option is useful if the target audience is content editors.</p>
Languages	Indicates how the connector handles languages.	<p>Blank (default) uses the Language attribute of the site definition in the <i>web.config</i> file. It is possible to specify the languages to crawl by enumerating the language codes: <i>en;fr-CA</i>. Use * (wildcard) to index all languages. Items of the media library are always indexed with the site default language.</p>
Target Audience	Indicates to the connector what the targeted audience of the source is. This option affects how the items are opened when a user clicks a search result.	<p>Web (default) opens the results as a standard Web page.</p> <p>Content Editor opens the results for edition instead.</p>
Index if no Layout	Indicates to the connector to index items that have no defined layout.	<p>By default, items with no layout cannot be directly found from a Web browser.</p> <p>True indexes these items. Then you can change the clickable URL using a mapping file. This can be useful to index the content of the <i>blog posts</i> module.</p>
Login Page*	URL of the login page when forms authentication is used.	Refer to the Forms Authentication section.
User Control*	ID of the control used by the user to enter username when forms authentication is used.	Refer to the Forms Authentication section.
Password Control*	ID of the control used by the user to enter password when forms authentication is used.	Refer to the Forms Authentication section.
Login Command*	Login command sent by the page when forms authentication is used.	Refer to the Forms Authentication section.
Authentication User identity*	User used by the connector.	Use the previously created user identity.

* These parameters are optional.

4. Click  Save .

Name	<input type="text" value="Nicam"/> ?
Source Type	Additional Connector - Sitecore
Addresses	<input type="text" value="http://nicam"/> <small>Depends on the additional connector used. One entry per line.</small>
Rating	<input type="text" value="Normal"/> ?
Content Start Path	<input type="text"/> ?
Content Admin	<input type="text"/> ?
Languages	<input type="text"/> ?
Mapping File	<input type="text" value="D:\SitecoreMapping.config"/>
Login Command	<input type="text"/>
Username Control	<input type="text"/>
Password Control	<input type="text"/>
Login Page	<input type="text"/>
Number of Refresh Threads	<input type="text" value="4"/> ?
Index if No Layout	<input type="checkbox"/>
IndexAllVersions	<input type="checkbox"/>
Body Format	<input type="text" value="Web Page"/>
Include Media Library	<input checked="" type="checkbox"/>
Database	<input type="text" value="web"/>
Target Audience	<input type="text" value="Web"/>
Target Site	<input type="text" value="website"/> ?
Parameters	Add Parameter ?
Option	<input checked="" type="checkbox"/> Recursive ? <input type="checkbox"/> Index the document's metadata ? <input type="checkbox"/> Document's addresses are case-sensitive ? <input checked="" type="checkbox"/> Generate a cached HTML version of indexed documents ? <input type="checkbox"/> Open results with cached version ? <input type="checkbox"/> Disable document summarization ?
Title Selection Sequence	Use the title extracted by the converter Automatically detect the title of documents Use the filename
Title Metadata Name	<input type="text" value="Title"/> ?
Authentication	User Identity <input type="text" value="(none)"/> ? Add Edit Manage user identities

- To index the *Sitecore* security elements, click **Permissions**, select **Use a security provider** (a security provider must be defined to select that option).

Permissions

- Index security permissions
- Specifies the security permissions to index
- Index security permissions and specify additional security permissions to index
- Use a security provider

Security Provider

Note: If a security provider is not used and the **Index security permission** option is selected, only the items that are reached by the *extranet\Anonymous* user or the *Everyone* group can be searched for. However, it is not possible to use the CES Search Interface component in your Web site.

Note: Additional information on the way to index the security permissions is available in the [Working with the Active Directory Module \(Sitecore 6 only\)](#) section.

Important: *Sitecore* handles user access control differently than CES does. For more information, refer to the [Security](#) section.

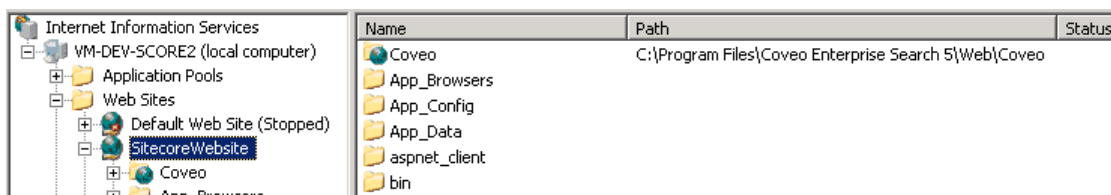
- Click  **Start**.

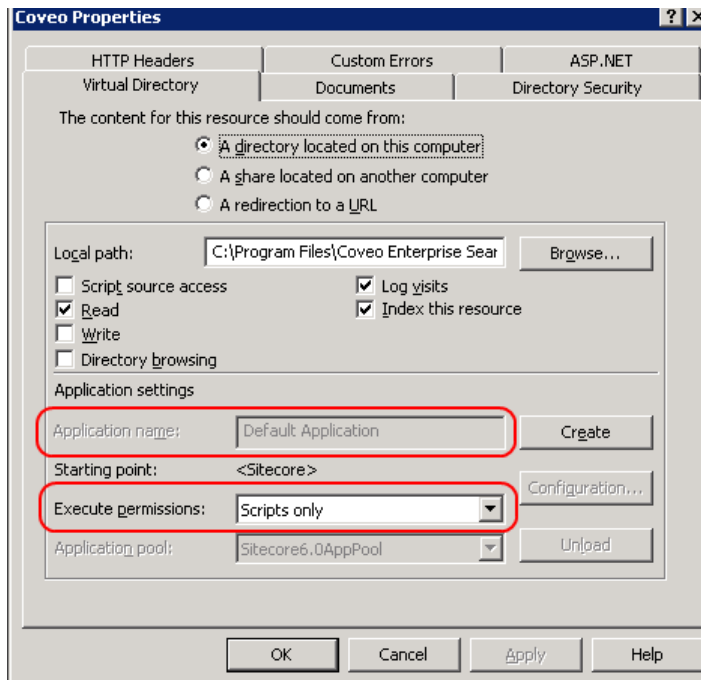
▶ **How to Integrate the CES Search Interface in a Sitecore Web Site**

The following details how to integrate the CES Search Interface component in your *Sitecore* Web site.

Important: The *Coveo* front-end (*Search Interface* installation) must be installed on the *Sitecore* server.

- Access the Internet Information Services (IIS) manager.
- Expand the root of your *Sitecore* Web application (e.g. *SitecoreWebSite*); create a virtual directory named *Coveo* that points to the `<Drive_Letter>:\Program Files\Coveo Enterprise Search 6\Web\Coveo` folder.
- Give at least the **Scripts only** permission. It is not necessary to create an application.




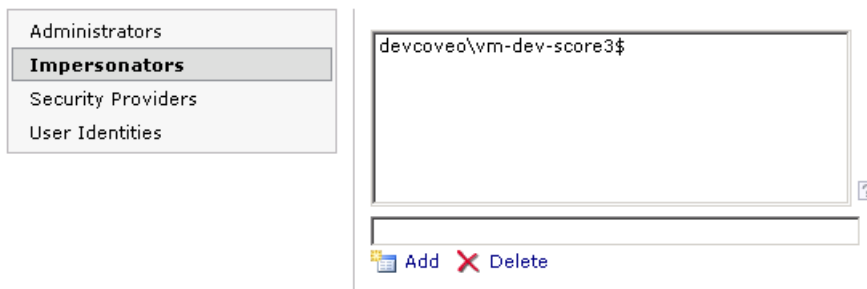


4. In the *Sitecore web.config* file, add the following entry at the bottom of the file:

```
</system.web>
<coveoEnterpriseSearch>
  <server hostname="CES_server_name" impersonate="False" mirrorName="default"
port="52800" />
</coveoEnterpriseSearch>
</configuration>
```

Note: The value of the *impersonate* attribute must be **True** if the CES indexer service is installed on another physical machine (front-end/back-end setup).

5. Access the **Administrators** page (Configuration > Security).
6. In the left navigation pane, click **Impersonator**. The **Impersonators** page is displayed.
7. Add the user running the *Sitecore* application pool* as an impersonator of the CES service. When the application pool is running under the *network service* identity, impersonator is *nt authority\network service* if CES is installed on the same server, *domain\machine_name\$* otherwise.
8. Click  **Add**.



* To find out what user is running the *Sitecore* application pool, open the IIS manager. Right-click and select the **Properties** menu of the **SitecoreAppPool** (or the application pool used by your *Sitecore* installation). The user is specified in the **Identiy** tab.

Adding the Search Interface is as simple as dropping the component in a *Sitecore* layout or sublayout.

```
<%@ Register TagPrefix="ces" Namespace="Coveo.CES.Web.Search.Controls"
Assembly="Coveo.CES.Web.Search, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=44110d16825221f2" %>
<%@ Assembly Name="Coveo.CNL, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=44110d16825221f2" %>
<%@ Assembly Name="Coveo.CNL.Web, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=44110d16825221f2" %>
<%@ Assembly Name="Coveo.CES.Interops, Version=6.0.0.0, Culture=neutral,
PublicKeyToken=44110d16825221f2" %>

<asp:Panel ID="pnResultsPanel" runat="server">
    <ces:SearchInterface id="c" runat="server"/>
</asp:Panel>
```

If the *Sitecore* security elements have been indexed (a security provider was previously created), it is possible to display the search results corresponding to the currently logged user. Using the *Sitecore* API, it is possible to get the currently logged user and pass that user to the CES Search Interface. The following code explains how to use the currently logged user with the CES Search Interface control dropped in a *Sitecore* layout or sublayout.

```
<script runat="server">
void c_OverrideUser(object p_Sender, Coveo.CES.Web.Search.Controls.OverrideUserEventArgs
p_Args)
{
    // The name of the security provider defined under the Admin Tool
    string securityProviderName = "Sitecore Security Provider";

    Coveo.CES.Interops.CESSearch.UserIDFactory factory = new
Coveo.CES.Interops.CESSearch.UserIDFactory();

    // Get the username from SitecoreAPI
    string userName = Sitecore.Context.GetUserName().ToLower();

    // Add to the collection of identities
    Coveo.CES.Web.Search.Providers.COM.COMUserIdentity user = new
Coveo.CES.Web.Search.Providers.COM.COMUserIdentity(factory.CreateExternalUserID(userName,
securityProviderName, false, null));
    p_Args.AdditionalIdentities.Add(user);
}
</script>
[...]
```

```
<ces:SearchInterface id="c" runat="server" OnOverrideUser="c_OverrideUser"/>
[...]
```

Note: CES security elements are case sensitive. The *Sitecore* connector always index security elements in lower-case letters. Be sure to provide a lower-case username to CES for the security to be resolved correctly.

Important: OnOverrideUser event is not available for all CES search controls (e.g. QuickSearch, SearchBox controls).

Finally, the Search Interface can target an explicit index source. The following explains how to do so:

```
<script runat="server">
override void OnInit(EventArgs p_Args)
{
    SearchBinding.MainSearchObject.SetupSearchBuilder += this.Search_SetupSearchBuilder;
    base.OnInit(p_Args);
}

void Search_SetupSearchBuilder(object p_Sender, SetupSearchBuilderEventArgs p_Args)
{
    p_Args.Builder.AddConstantExpression( "@Source=MySourceName" );
}
</script>
```

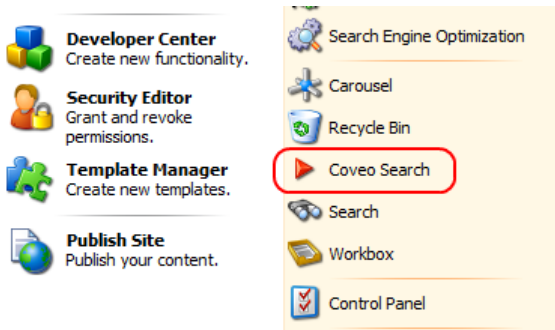
► **How to Integrate the CES Search Interface in the Sitecore Content Editor (Sitecore 6 only)**

The following details how to integrate the CES Search Interface component in the *Sitecore Content Editor*.

Important: The Coveo front-end (Search Interface installation) must be installed on the *Sitecore* server.

Follow the procedure from the [How to Integrate the CES Search Interface in a Sitecore Web Site](#) section (if not already done).

By installing the *Sitecore6.0CoveoWebService.zip* package, the CES search interface is automatically integrated with the *Sitecore* Content Editor. The application is available from the *Sitecore* application menu.



In order to search from the Content Editor, certain customizations must be performed:

1. Create the three following custom fields in CES: **SCItemID** (meta = `_CESSCID`), **SCLanguageCode** (meta = `_CESSCLanguageName`) and **SCVersion** (meta = `_CESSCVersion`).

Field Set : Default Scheme - Custom Fields Help

List of user defined fields indicating which ones are available for querying and/or merged in the keyword index.

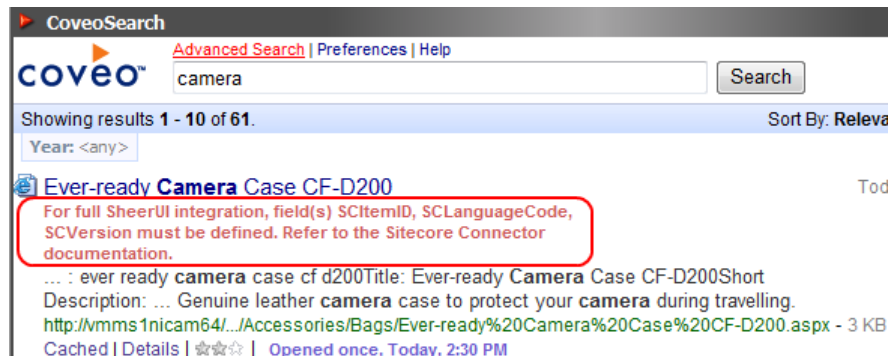
Name	Type	Default Value	Field Queries	Free Text Queries	Group By	Sort by
<input type="checkbox"/> SCItemID	String		No	No	No	No
<input type="checkbox"/> SCLanguageCode	String		No	No	No	No
<input type="checkbox"/> SCVersion	String		No	No	No	No

Select: All, None

2. Create a source in CES that will index the editors' content. The source needs to use a security provider (for more information, refer to http://www.coveo.com/en/support/online-help/admin/What_Are_Security_Providers.html).
3. In the *Sitecore* installation, edit the *CoveoSearch.aspx* file (located in the `WebSite\sitecore\shell\Applications\Coveo` folder), Uncomment and modify the source code of the control to match your security provider settings and source definition.
4. To make the results open directly in the *Sitecore* Content Editor, open the *ResultTemplate.ascx* file (located in the `Program Files\Coveo Enterprise Search 6\Web\Coveo\Skins\Default` folder). Replace the **ResultOpenLink** control by **SitecoreResultSheerUIOpenLink**. It is also possible to copy the *Default* skin to a skin dedicated to the Content Editor search:

```
<span class="<%=SearchStyles.ResultTitle%>">
  <ces:SitecoreResultSheerUIOpenLink id="opl" runat="server">
    <ces:ResultTitle id="ttl" Length="90" runat="server" />
  </ces:SitecoreResultSheerUIOpenLink>
</span>
```

Note: The **SitecoreResultSheerUIOpenLink** control will display an error message if there is a problem with the custom fields definition. To hide this error message, simply set the **ShowIntegrationErrors** control property to *false*.



Appendix

► How to Resolve Parameter Order

The following details the order CES prevails to resolve the different source parameters that can be set:

- **Target Site:** If set, use the specified site. If not, use **Website**.
- **Database:** If set, use the specified database. If not, use the default database of the target site resolved at **Target Site**.
- **Content Start Path:** If set, use the specified value. If not, use the default root path of the target site resolved at **Target Site**.
- **Include Media Library:** If **True**, append the media library root item to the content start path value resolved at **Content Start Path**.
- **Include All Languages:** Create a document for each language provided (for all languages if value is a wildcard - *). If parameter is blank, create a document for the default language of the target site resolved at **TargetSite**. If there is no language set on the site, **English** will be used.

▶ Metadata

Except for a few system fields, most of the content in *Sitecore* is a question of custom fields and values. If metadata is indexed, it can be searched; however, it is possible to display, sort, filter, group by, etc. on some of those fields. The following table details the metadata elements that are always attached to any *Sitecore* item. Basically, it is the *Sitecore* item property field preceded by the “_CESSC” prefix.

_CESSCServerBaseUrl	_CESSCChildrenIDs	_CESSCLocker
_CESSCContentPath	_CESSCParentID	_CESSCName
_CESSCFullPath	_CESSCParentPath	_CESSCUri
_CESSCID	_CESSCPath (sysuri)	_CESSCVersion
_CESSCIsContentItem	_CESSCTemplateID	_CESSCWorkflowstate
_CESSCIsFullyQualified	_CESSCTemplateName	_CESSCCreated (sysaddeddate)
_CESSCIsMasterPart	_CESSCDisplayName	_CESSCCreatedBy
_CESSCIsMediaItem	_CESSCFriendlyUrl	_CESSCRevision
_CESSCKey	_CESSCHasLayout	_CESSCUpdated (sysdate)
_CESSCLongID	_CESSCLanguageName	_CESSCUpdatedBy (sysauthorloginname)
_CESSCMediaPath	_CESSCLanguages	
_CESSCMediaUrl	_CESSCLanguageTitle	

Note: (<fieldname>) = Metadata mapped automatically to CES system fields

For any other item fields, the metadata is formed of the field name and value. For some field types, the value is stored as a reference to another *Sitecore* item. By default, the *Sitecore* connector sets the **Key** value of the referenced item as metadata value. If there is more than one referenced item for a single field, all the **Key** values are put in a single value separated by the “;” character.

▶ Mapping File

Sitecore metadata can be organized in various ways. For example, a single *Sitecore* item can be composed of multiple fields as well as multiple references to other *Sitecore* items. Those referenced items can also have many fields and references, etc. Depending on how the metadata is organized, it might be difficult for the *Sitecore* connector to retrieve all the fields that compose a *Sitecore* item.

This section explains how to write the XML mapping file, which can be used to map metadata fields to CES fields. Moreover, this file can be used to modify the way referenced *Sitecore* field values are resolved.

Basic XML structure of the configuration file

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <CommonMappings>
    <Fields>
      <FieldName>mapping value</FieldName>
    </Fields>
  </CommonMappings>
  <Mapping template="{00000000-0000-0000-0000-000000000000}">
    <Fields>
      <FieldName>mapping value</FieldName>
    </Fields>
  </Mapping>
  <Mapping item="{00000000-0000-0000-0000-000000000000}">
```

```

<Fields>
  <FieldName>mapping value</FieldName>
</Fields>
</Mapping>
<Ignore template="{00000000-0000-0000-0000-000000000000}">
<Ignore item="{00000000-0000-0000-0000-000000000000}">

```

<CommonMappings> section

This section defines field mappings that apply to all *Sitecore* items.

<Mapping> section

This section defines field mappings that apply to all *Sitecore* items that use the templates specified in the **template** attribute. Field mapping can also be applied directly on items by specifying the **item** attribute. Templates and items are identified by their *Sitecore* ID (guid) and separated by the pipe (|) character. If both attributes are set, only the **template** attribute will be considered. It is a better approach to set mappings on templates rather than items.

<Ignore> section

This section specifies templates and items that should be ignored by the connector. Templates and items are identified by their *Sitecore* ID (guid). Each entry accepts only one template or item. If both attributes are set, only the **template** attribute will be considered.

<FieldName> entry

This entry defines the field targeted by the mapping. The name of the entry corresponds to the name of CES system or custom field defined in the field set used by the source.

Mapping Value

This value is the mapping itself. It is composed of free text and metadata tags. The connector supports different metadata tags: %<[<metadata>], %<{<metadata>:<mapping value>} or @[<fully_qualified_class_name>].

A complete sample of the *Sitecore* connector configuration file is provided in the [Examples](#) section.

```

<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <CommonMappings>
    <Fields>
      <Title>%[_CESSCDisplayName]</Title>
    </Fields>
  </CommonMappings>
</Sitecore>

```

This mapping tells the connector to set the **Title** field of a document with the value of the **_CESSCDisplayName** metadata. This applies to all *Sitecore* documents. It is possible to browse the item hierarchy using the referenced metadata fields.

For example, a *Sitecore* item is based on the **Book** template. This template defines a field **Contributor** which is a reference to a **Person** item somewhere else in the content tree. The **Person** item contains several fields, including the *Firstname* and the *Lastname* fields. Here are the different possibilities:

- By default, the metadata **Contributor** is ignored.

- If the **Index the document's metadata** option is selected on the source, the **Contributor** metadata is indexed, but its value is the **Key** of the contributor item.
- It is possible to create a new custom field named **Contributor** (or any relevant name) and set its value to the complete name of the contributor metadata.

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <Mapping template="{AF0AD32B-8188-45E0-B354-5226F41D801B}"> <!--BookTemplate-->
    <Fields>
      <Contributor>%[Contributor.Lastname], %[Contributor.Firstname]</Contributor>
    </Fields>
  </Mapping>
</Sitecore>
```

If the field **Contributor** is a multiple value reference (i.e., the book might have more than one contributor), the user can tell the connector to concatenate the values by using the following mapping annotation: **%{<metadata>:<mapping value>}**

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <Mapping template="{AF0AD32B-8188-45E0-B354-5226F41D801B}"> <!--BookTemplate-->
    <Fields>
      <Contributors>%{Contributor:%[Lastname], %[Firstname]}</Contributors>
    </Fields>
  </Mapping>
</Sitecore>
```

Multiple value references can be organized by template. For example, in the *Nicam* sample site, a camera contains a field called **Accessories**. These accessories are built from different templates: Lenses, Flash, Other Accessories. To organize the value references by template, use the following annotation: **%{<metadata|template ID1|template ID2|...>:<mapping value>}**. Template IDs must be used without the “{, }” characters:

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <Mapping template="{B072B7C7-6F3F-4316-B8D7-010629AEBEF1}"> <!--SLR-->
    <Fields>
      <Accessories>%[Accessories.Title]</Accessories>
      <Lenses>%{Accessories|8FAC8E12-7459-43F8-97E8-1BC6840B9226:%[Title],%[Focal
length]}</Lenses >
      <Flash>%{Accessories|95681CF6-3635-49EC-A09A-CC548FA62389:%[Title],%[Guide
number]}</Flash>
      <Misc>%{Accessories|A93FA2C4-3AE4-45C2-8C3F-EFA7E129537E:%[Title]}</Misc>
    </Fields>
  </Mapping>
</Sitecore>
```

Finally, the mapping can be handled programmatically by using the **@[<fully_qualified_class_name>]** annotation. Hence, the connector can use a custom class to resolve a field value if none of the out-of-the-box mapping solutions match your needs. Although this can be a slower approach (the connector has no control on how the code is executed inside the class), it is a very powerful way to resolve mapping when complicated rules are applied. For example, a *Sitecore* items has the following fields: long description, short description, title and name. Only the name is required in *Sitecore*, but you want to use the long as well as the short description if one of them is present. If not, you will use the title before the name.

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <Mapping template="{AF0AD32B-8188-45E0-B354-5226F41D801B}">
```

```
<Fields>
  <Description>@[MyNamespace.MyDescResolver, MyAssembly]</Description >
</Fields>
</Mapping>
</Sitecore>
```

This is the code of the class contained in *MyAssembly.dll* used to resolve the description field:

```
using System;
using System.Collections;
using System.Collections.Generic;
using Coveo.CES.CustomCrawlers.Sitecore.ContentService;
using Coveo.CES.CustomCrawlers.Sitecore.Interfaces;

namespace MyNamespace {

public class MyDescResolver: ISitecoreMappingResolver
{
    [CLSCompliant(false)]
    public string ResolveMapping(ContentItemMeta p_Meta,
                                Hashtable p_ProcessedMeta,
                                List<string> p_MediaReferences)
    {
        string desc = null;
        string shortdesc = null;
        string title = null;

        foreach (ContentItemField field in p_Meta.Fields) {
            if (field.Name == "long description") {
                desc = field.Value.StringValue;
            } else if (field.Name == "short description") {
                shortdesc = field.Value.StringValue;
            } else if (field.Name == "title") {
                title = field.Value.StringValue;
            }
        }

        return desc ?? shortdesc ?? title ?? p_ProcessedMeta["_CESName"].ToString();
    }
}
} //namespace
```

The class must implement the **ISitecoreMappingResolver** interface. Your assembly will have to reference *Coveo.CES.CustomCrawlers.Sitecore.dll*, after which you only need to implement the **ResolveMapping** method. This method has the following parameters:

- **ContentItemMeta**: The *Sitecore* metadata of the item.
- **Hashtable**: The already processed system metadata (refer to the [Metadata](#) section).
- **List<string>**: The list of media items referenced by the items. This list can be modified if required.

There are additional examples in the [Examples](#) section.

Note: The mapping values are case-sensitive. If the *Sitecore* field is named “Short Description”, the following annotation will not work:

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <Mapping template="{AF0AD32B-8188-45E0-B354-5226F41D801B}">
    <Fields>
```

```

    <ShortDesc>%[short description]</ShortDesc > <!--No field will match-->
  </Fields>
</Mapping>
</Sitecore>

```

▶ Forms Authentication

The connector supports forms authentication when a Web site containing secured sections is indexed. This section describes how to set the different forms authentication parameters when the source **Body Format** option is set to **Web Page** and **Generate a cached HTML version of indexed documents** is selected.

Note: The connector must access the Web page to store a cached version of it and needs to be authenticated to do so.

Here are the four parameters to set:

- Login Page

This is the URL of the page where users log on.

- User Control and Password Control

These are the IDs of the controls where the users enter their credentials. One way to get these IDs is by exploring the source of your Web page (e.g. *View Source* menu of Internet Explorer).

For example: `ctl14$ctl00$login_username`, `ctl14$ctl00$login_password`

- Login Command

This is the command passed by the Web page to the server to invoke the login procedure.

For example: `ctl06$ctl01$LoginButton=Login`

Some Web sites use a lot of dynamic content (Ajax) and the page source might not be enough to get the forms authentication parameters. You can use an external Web debugger (like [Fiddler](#)) to find what are the values passed to the server when the login command is invoked.

▶ Security

Sitecore handles security access rights slightly different than CES. In CES, **deny** always overrules **allow**. That is, whenever a user (or one of its roles) is denied access on a document, CES denies the document for this user. In *Sitecore*, the same rule applies, but there are exceptions no matter if the access rule is explicitly set on an item or not, inherited from another item, if the user is an administrator, etc.

CES overcomes this problem by implementing heuristic detecting cases where a user might have the right to see a document; even if that user is part of the role that does not have the right to see the same document.

Important: The counterpart of that solution is that security changes on the *Sitecore* side **will not** be considered by CES in a live indexing run. You **must** have a refresh schedule set on your sources if such security is present in your site. Fortunately, CES logs a warning during the source indexing if such security is encountered. If your security model works like the one expected by CES, the connector operates normally.

▶ About Live Indexing

To enable live indexing, *Sitecore's* **Engines.HistoryEngine.Storage** class must be defined in your *Sitecore's* *web.config* file for the database that is targeted by the connector's source. For example, your source is targeting the site named **website**. The **web** database is serving that site. Make sure that the **Engines.HistoryEngine.Storage** entry is defined on that database in the *web.config* file.

Sitecore 5.3

```
<database id="web" singleInstance="true" type="Sitecore.Data.Database,
...
<Engines.HistoryEngine.Storage>
  <obj type="Sitecore.Data.$(database).$(database)HistoryStorage, Sitecore.$(database)">
    <param desc="connection" ref="connections/$(id)"></param>
    <EntryLifeTime>30.00:00:00</EntryLifeTime>
  </obj>
</Engines.HistoryEngine.Storage>
...
</database>
```

Sitecore 6

```
<database id="web" singleInstance="true" type="Sitecore.Data.Database,
...
<Engines.HistoryEngine.Storage>
  <obj type="Sitecore.Data.$(database).$(database)HistoryStorage, Sitecore.Kernel">
    <param connectionString="$(id)" />
    <EntryLifeTime>30.00:00:00</EntryLifeTime>
  </obj>
</Engines.HistoryEngine.Storage>
...
</database>
```

Note: Every modification to the site's database will be considered for indexing on a live indexing run. You might experience a slow live indexing operation in some special design of your *Sitecore* Web site. For example, if your Web site is counting the number of times a page has been visited by modifying an item metadata field, this will mark the item as modified. Then, the item will be re-indexed on the next live indexing run. When CES crawls a *Sitecore* site, it might load every page to retrieve its html content (depending of your source options). In this case, the page visited counter will be updated for each single page, causing each page to be marked as modified and re-indexed on the next live indexing run and so on. If you must rely on metadata item fields that have to be modified on-the-fly, you can workaround that limitation by checking the *http request user agent* before updating the database. When the *Sitecore* connector loads a page from the Web site, it uses the **Coveo Sitecore Crawler** user agent.

▶ Working with the Active Directory Module (*Sitecore 6 only*)

The *Sitecore Active Directory* module is used to integrate the Active Directory (AD) domain users and groups with the *Sitecore* CMS. CES natively handles security permissions using the AD domain. The *Sitecore* connector has different options to set AD security permissions on indexed documents.

Using a Security Provider

It is possible to configure a security provider as described in the [How to Configure the Security Provider \(Optional\)](#) section. At this point, the AD domain is seen as a standard *Sitecore* domain. However, additional options on the security provider can be used in order for CES to map these users and groups to AD users and groups:

- Add the **ADDomain** parameter to the security provider parameter value.
For example, `AssemblyPath="<Drive_Letter>:\Program Files\Coveo Enterprise Search 6\Bin\Coveo.CES.CustomCrawlers.Sitecore.dll";WebServiceUrl="http://SitecoreWebSite";ADDomain=myDomainName`
- Select the **Support expand user** checkbox. The security provider now support both *Sitecore* and AD security models.

Note: AD group expansion is limited to the LDAP root folder specified in the domain *connectionString* entry (part of the *Sitecore* AD module configuration).

For example, Search root: `LDAP://corp.domain.com/OU=Users,DC=CORP,DC=DOMAIN,DC=COM`. Only the users under the *Organizational Unit* (OU) *Users* will be considered by CES. If the group *Everyone* is allowed to see a document, only the users and groups under the OU *Users* will have access to the document.

Without Using a Security Provider

In the **Permissions** tab of the source, select the **Index security permissions** option (it is possible to use a security provider if the source has to manage AD and *Sitecore* permissions). In the **General** tab, enter the domain name that matches an AD domain in the **ADDomains** parameter (refer to the [Hidden Parameters](#) section). If you have multiple AD domains, enter all the names separated by semicolons (;).

For example, `myDomainA;myDomainB`

CES will directly use the AD for group expansion. This option can be used in collaboration with a standard security provider that will handle the *Sitecore* internal domains.

Note: Group expansion will not follow any LDAP search root filtering that might have been specified in the *Sitecore* Active Directory module configuration. However, this option is much faster than when using a security provider.

Note: Depending of your configuration, you might have to change the impersonation parameters of your Web site and CES when using the AD security model (with or without using a security provider).

▶ Hidden Parameters

Here are parameters that can be added on a *Sitecore* source as custom parameters.

Field	Description	Example
WebServiceTimeout	Modifies the time the connector waits for a content Web service request to complete.	Any value in seconds. Default is 100 seconds.
LiveIndexingDelay	Modifies the delay between live indexing runs.	Any value in seconds. Default is 300 seconds.
AdminThreshold	Specifies the maximum number of users a security group/role can contain. To handle larger groups, increase this number and consider increainsing the WebServiceTimeout parameter.	Default is 250 users.

ADDomains	Specifies to the connector what <i>Sitecore</i> security domains come from the Active Directory module (refer to the Working with the Active Directory Module (Sitecore 6 only) section).	A list of domain names separated by semicolons (;).
ErrorDetails	Changes the error messages verbosity level.	Possible values are: Normal , Detailed . Default is Normal .
UseCache	Specifies to the connector to use its caching capability in order to improve crawling performances.	Default is true . It is not recommended to set this parameter to false . However, caching can be disabled if you are experiencing serious memory problems on your Sitecore installation while crawling.
LoopDelayRefresh LoopDelayLive LoopDelayItemsBatch	Modifies the time a refresh (or live indexing) thread waits between calls to retrieve <i>Sitecore</i> items. This is useful to slow down the crawling if a refresh (or live indexing) operation puts too much load on the <i>Sitecore</i> server.	Any value in milliseconds. Default is 0 .
UseItemsBatch	Specifies to the connector to make multiple calls in order to get the list of all <i>Sitecore</i> items to index.	Default is true .
ItemsBatchSize	Specifies the maximum number of items to get at once when the UseItemsBatch parameter is set.	Default is 300 .
LiveIndexingIncludeSecurity	Specifies to the connector to handle the security changes during live indexing. This will slow down the live indexing process because of the hierarchical <i>Sitecore</i> security model.	Default is false .

▶ Examples

Mapping file for indexing blog posts when using the *Sitecore*'s blog module in the **Printers** sample site:

Index if No Layout option must be selected on the source.

```
<?xml version="1.0" encoding="utf-8" ?>
<Sitecore>
  <CommonMappings>
    <Fields>
      <Title>%[_CESSCDisplayName]</Title>
    </Fields>
  </CommonMappings>
  <Mapping template="{5CF2ED9B-6C32-4FA3-9549-2AB77085B131}"> <!--UserBlog-->
```

```

    <Fields>
      <ClickableUri> %[_CESSCServerBaseUrl]/Company/Blogs.aspx?blog=%[Blog
Title]</ClickableUri>
      <PrintableUri> %[_CESSCServerBaseUrl]/Company/Blogs.aspx?blog=%[Blog
Title]</PrintableUri>
    </Fields>
  </Mapping>
  <Mapping template="{1FBDD65D-5029-46F1-8D75-AF3E68810B25}"> <!--Article-->
    <Fields>
  <ClickableUri>%[_CESSCServerBaseUrl]/Company/Blogs.aspx?post=%[Title]&blog=%[_CESSCParent
ID.Blog Title]</ClickableUri>
  <PrintableUri>%[_CESSCServerBaseUrl]/Company/Blogs.aspx?post=%[Title]&blog=%[_CESSCParent
ID.Blog Title]</PrintableUri>
    </Fields>
  </Mapping>
  <Mapping template="{FB71F255-31D5-417A-BD5C-12D458EB8FDB}"> <!--Comment-->
    <Fields>
  <ClickableUri>%[_CESSCServerBaseUrl]/Company/Blogs.aspx?post=%[_CESSCParentID.Title]&blog
=%[_CESSCParentID._CESSCParentID.Blog Title]</ClickableUri>
  <PrintableUri>%[_CESSCServerBaseUrl]/Company/Blogs.aspx?post=%[_CESSCParentID.Title]&blog
=%[_CESSCParentID._CESSCParentID.Blog Title]</PrintableUri>
    </Fields>
  </Mapping>
</Sitecore>

```