FatWire SOFTWARE

Community Server Blog Module

Version 1.0

System Operations Guide

For Developers, Administrators, and Content Providers

Document Revision Date: *Jun. 11, 2010*

FatWire Community Server Blog Module: System Operations Guide
Document Revision Date: Jun. 11, 2010
Product Version: Version 1.0

FatWire Technical Support
www.fatwire.com/Support

FatWire Headquarters
FatWire Corporation
330 Old Country Road
Suite 207
Mineola, NY 11501

www.fatwire.com

Table of

Contents

Part 1. Developers

Part 2. Administrators

Part 3. Content Providers

Introduction

This guide provides instructions for installing and configuring the FatWire Community Server Blog module on a content management site. It begins with an overview of the blog module, and moves on to describe the default blog components the installer adds to the site. This guide also provides information about reusing the default blog components to fit your site requirements. The last chapter in this guide provides you with instructions for creating the content that will be displayed on the blog pages of your website.

Who Should Use This Guide?

This guide is written for FatWire Content Server developers, administrators, and content providers. It is assumed that these users are familiar with the concept of blogs and the Content Server flex family structure. These users should also have prior knowledge of the FatWire Community Server application and the commenting functionality the application can add to their company's website.

Developers must have experience installing components on a FatWire Content Server system. They must also have a thorough understanding of their company's site design, and they must be able to develop templates and CSElements.

Chapter 1

Installing the FatWire Community Server Blog Module

This chapter provides an overview of the FatWire Community Server Blog module and instructions for installing the blog module on a content management site. It also provides information about the components that are added to the site once the blog module is installed.

This chapter contains the following sections:

Overview

Installing the FatWire Community Server Blog module on a content management site supplies content providers with the necessary components to create pages dedicated to displaying blogs on a website.

The blog module components are:

- The `Blog` flex family
- A default SiteEntry asset
- Default blog templates
- Default CSElements
- Sample blog page assets

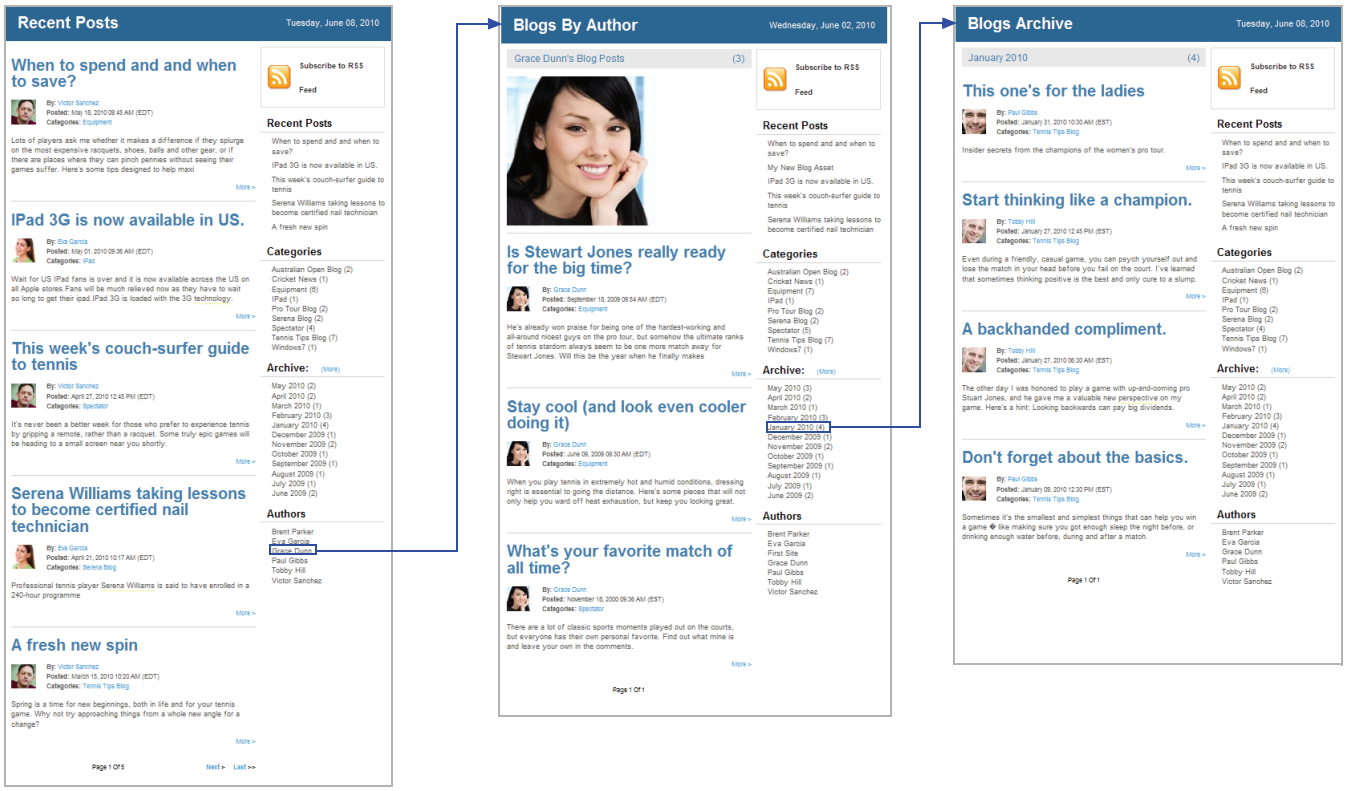
These components are pre-configured with default values and code which, out-of-the-box, render the sample blog Web pages shown in Figure 1.

Figure 1: Blog module's sample pages

The "Recent Posts" page is the sample blog home page. Clicking an author's name, in the "Authors" section of the right navigation panel, renders the "Blogs By Author" sample page.

The "Blogs By Author" sample page displays the blogs posted by a given author. Clicking a month, in the "Archive" section of the right navigation panel, renders the "Blogs Archive" page.

The "Blogs Archive" sample page displays the blogs posted for a given month.



To add blog pages to your website, you can reuse and reconfigure the blog components to fit your requirements.

If you integrate the blog module with Community Server, Community Server's comment fields are automatically inserted into the blog pages and displayed to site visitors. Commenting capability enables visitors to contribute feedback on the blogs, which in turn enables the content providers of your website to improve the content of the blogs they post.

Prerequisites

Before installing the blog module, ensure the following requirements are met:

- The blog module must be installed on a fully functional Content Server system running version 7.5 Patch 3, or higher. The Content Server installation must be either a content management or development system. The Content Server system must also have the Web Experience Management (WEM) Framework installed.
- If you wish to integrate Community Server functionality with the blog module, then you must know the host name and port number of Community Server's production system. For more information about Community Server functionality, see the *Community Server User's Guide*.
- Select or create a content management site on which to install the blog module. This site will be dedicated to containing the blog module components and sample assets. The purpose of this site is to help you review the sample assets, and to learn and experiment with the blog data model. **The same site must exist on the delivery system.**
- Run the installation on the content management or development system. You must also run the same installer on the delivery system.

Installing the Blog Module

This section provides instructions for using the graphical interface to install the Community Server Blog module on the content management site of your choice. You will run the installer on the content management or development system, and on the delivery system.

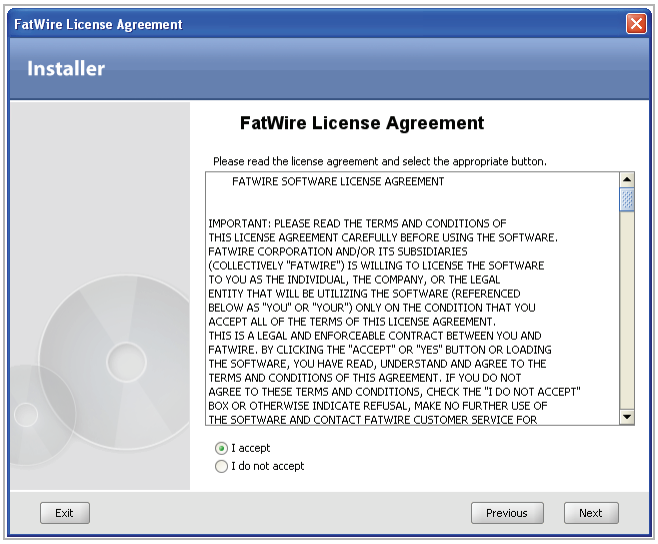
Note

Make sure your Content Server installation is not running when you begin installing the blog module. If Content Server is running, the installation will fail.

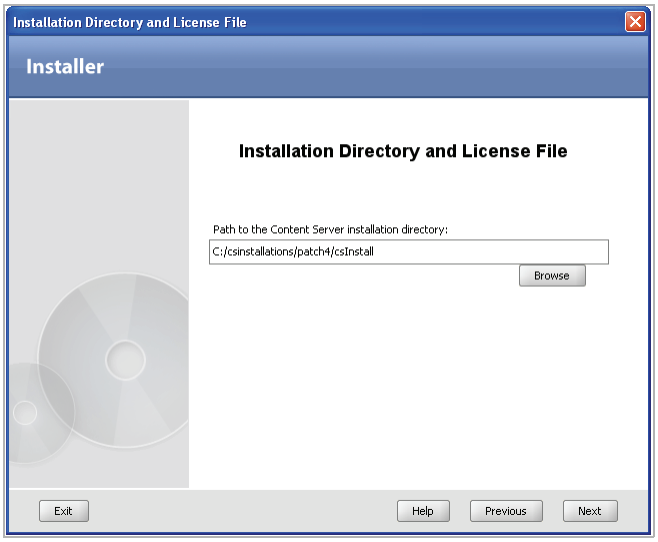
To install the blog module on your Content Server system

1. Extract `CommunityServerBlogs.zip` to a machine running Content Server 7.5 Patch 3 or higher.
2. Run the blog installer script:
   - Windows: `blogInstall.bat`
   - Unix: `blogInstall.sh`
3. In the "Welcome" screen, click **Next**.

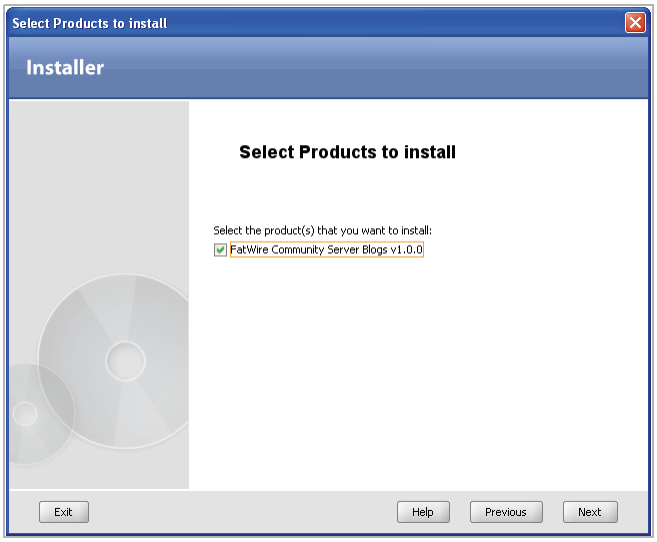
4. In the "FatWire License Agreement" screen, select **I accept**, and click **Next**.



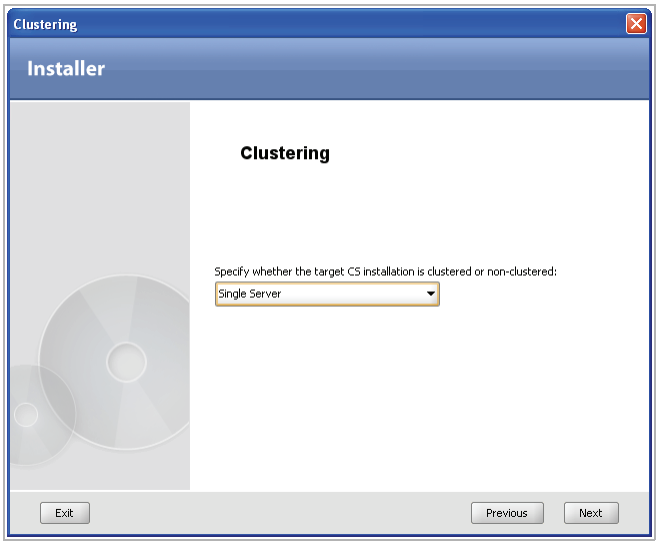
5. Enter the path to your Content Server installation directory.



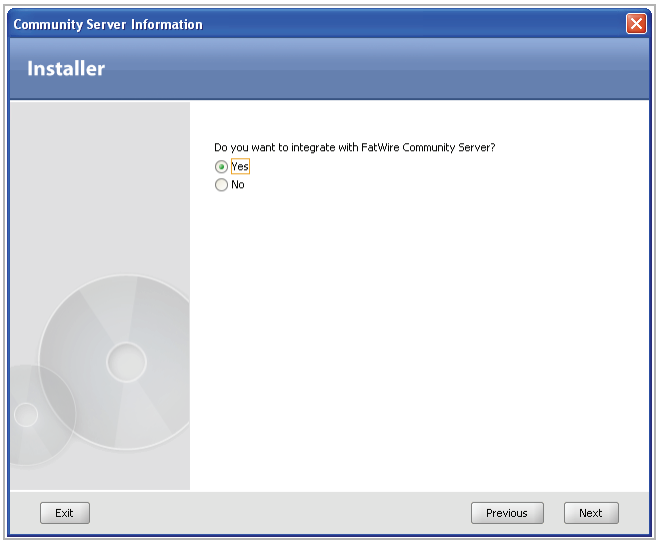
6. Select the product(s) you wish to install.



7. Specify whether you are installing the blog module on a single server or a cluster member.



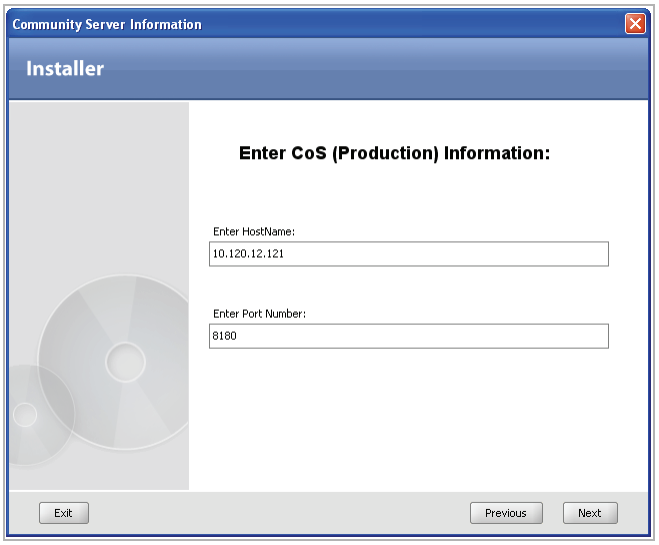
8. Select **Yes** if you wish to integrate the blog module with FatWire Community Server.



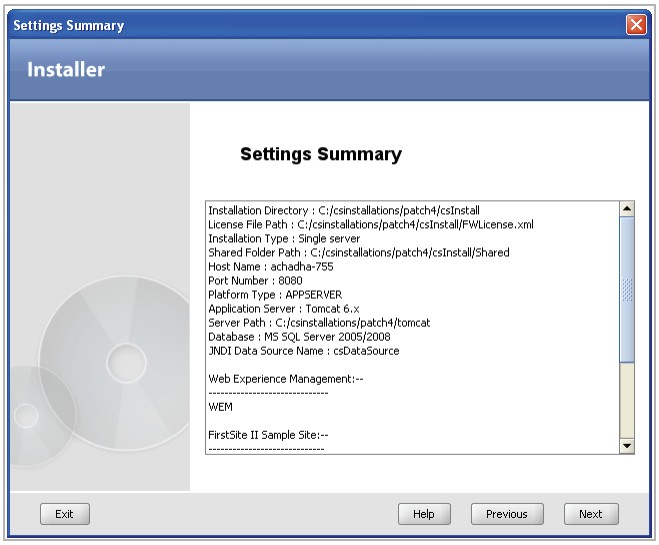
Note

Integrating the blog module with Community Server automatically adds the tags for the comments widget and comments link to the blog templates of your website.

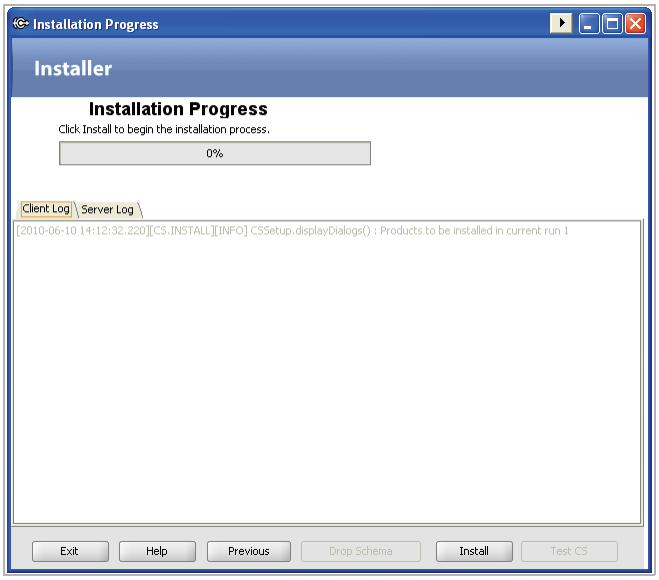
9. If you chose to integrate the blog module with Community Server, then you must enter the following information about Community Server's production system:
   - Host name or IP address of the machine running Community Server.
   - Port number on which Community Server is running.



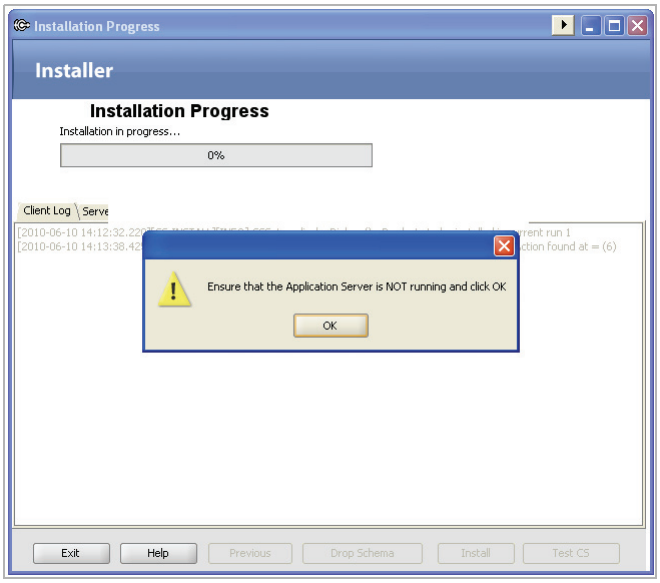
10. In the "Settings Summary" screen, review the environment on which you are installing the blog module. Click **Next**.



11. Click **Install**.



12. Ensure that your application server is not running. Click **OK**.

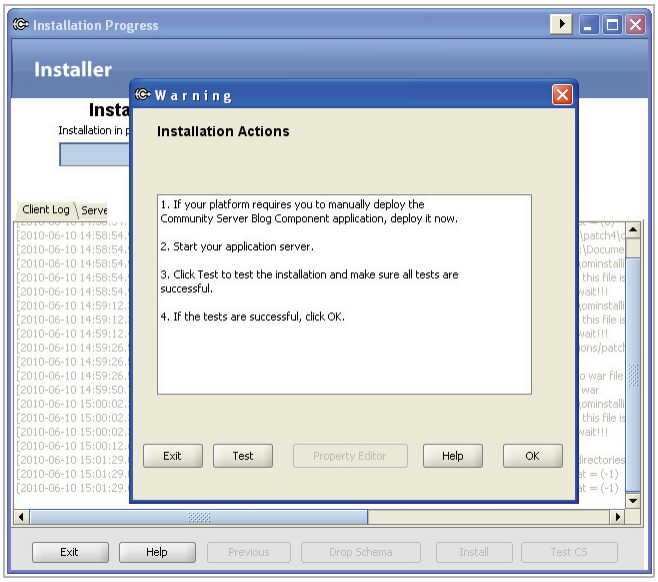


13. During the installation the "Installation Actions" dialog box opens:

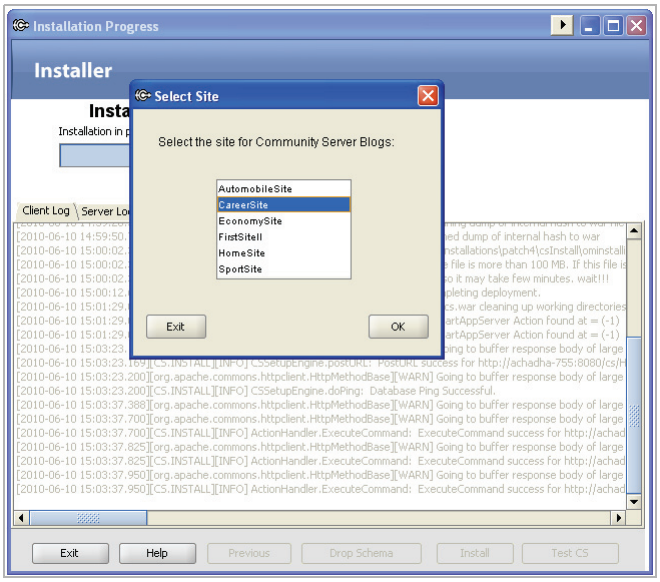
a. Start your Content Server system.

b. Click **Test** to test the database connections between the blog module and Content Server.

c. Click **OK** to continue with the installation.



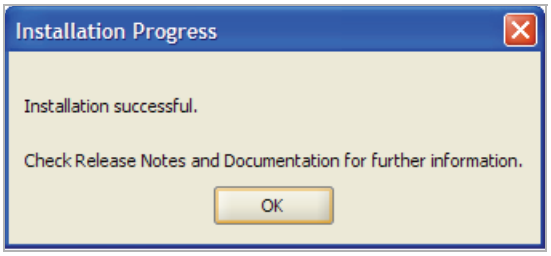
14. The "Choose Site" screen opens, listing the content management sites on your Content Server system. Select the site on which you wish to install the blog module. Click **OK**.



Note

If you run the blog installer more than once, the blog module will be automatically installed on the site that you selected the first time you ran the installer.

15. The installation is complete. Click **OK** to exit the installer. Stop, and then restart your Content Server system.



16. Verify that the installation was successful. See "Verifying the Installation," on page 17. If you need information about the blog components and sample assets that are added to your Content Server system, see "The Blog Module's Components," on page 17.

Verifying the Installation

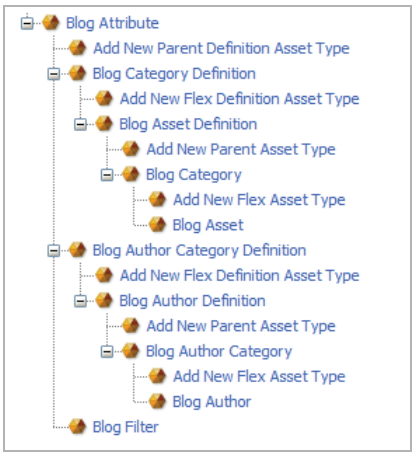
The installer added the `Blog` flex family to your content management site. This flex family provides the framework for creating blog Web pages.

To verify that the Blog flex family was added to the CM site

1. Log in to the Content Server Advanced interface with the credentials of a general administrator (`fwadmin`/`xceladmin`, by default).
2. Select the site on which you installed the blog module.
3. Navigate to the Content Server tree, and select the **Admin** tab.
   a. Expand the **Flex Family Maker** node.

      You should see the **Blog Attribute** asset type listed.

   b. Expand all of the nodes for the `Blog` flex family. The blog hierarchy looks as follows:



   If the `Blog` flex family hierarchy is rendered as shown above, then you have successfully installed the blog module. For more information about the blog components that are added to your Content Server installation, see "The Blog Module's Components."

The Blog Module's Components

During the installation process, the blog installer added the following components to your Content Server system:

- The `Blog` flex family was added to the CM site that you selected:
  - `FW_BlogAsset` (Blog Asset)
  - `FW_BlogParent` (Blog Category)
  - `FW_BlogAssetDef` (Blog Asset Definition)
  - `FW_BlogParentDef` (Blog Category Definition)
  - `FW_BlogAttribute` (Blog Attribute)
  - `FW_BlogFilter` (Blog Filter)
  - `FW_BlogAuthor` (Blog Author)

- `FW_BlogAuthorParent` (Blog Author Category)
- `FW_BlogAuthorDef` (Blog Author Definition)
- `FW_BlogAuthorParentDef` (Blog Author Category Definition)

- Start menu items for the `Blog` flex family are automatically enabled on the site.

Note

A tree tab for the blog module must be manually created and enabled for the site. This tree tab will supply content providers with a way of reviewing the hierarchy of the blog assets and categories. For instructions about creating a tree tab, see "Enabling Blog Asset Types," on page 54.

- The following blog related components and sample assets were added to your content management site:
  - Sample blog page assets
  - Default blog CSElements
  - Default blog templates
  - A default SiteEntry asset

  Out-of-the-box, you can publish the sample blog pages to a test website to view their default appearance (shown in Figure 1, on page 8). You can also reuse and reconfigure the default blog components to fit your requirements. For more information about the components that compose the sample blog pages, see Chapter 3, "Sample Blog Pages."
- If you integrated the blog module with FatWire Community Server, the following properties are added to the `futuretense_xcel.ini` file:

Property	Description
`fwblog.cos.enabled`	This property is set to `TRUE` if FatWire Community Server is integrated with the existing Content Server installation.
`cos.hostname`	This property points to the host name of your FatWire Community Server production system.
`cos.portnumber`	This property points to the port number of your FatWire Community Server production system.

Chapter 2

The Blog Module's Data Model

This chapter provides information about the blog module's flex family hierarchy, and the assets that are created for it during the blog module's installation process.

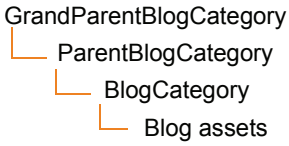
This chapter contains the following section:

Overview of the Blog Flex Family

The `Blog` flex family provides the data structure for storing blog assets and authors. The `Blog` flex family members, shown in the figure below, are organized as follows:

Blog assets are stored in a four-level hierarchy:

- `Blog Asset` is used to create the unit of content that will be displayed on the website.
- `Blog Category` is used to name the blog categories.
- `Blog Category Definition` is used to place blog categories in hierarchical order. By default the blog category definitions are: `GrandParentBlogCategory` (first level), `ParentBlogCategory` (second level), and `BlogCategory` (third level):



Admin tab **> Flex Family Maker**

- `Blog Asset Definition` is used to determine a blog asset's parent. By default, a blog asset's parent is a category at the `BlogCategory` level (third level). The `Blog Asset Definition` also specifies which blog attributes make up the Blog Asset form.
- `Blog Attribute` is used to create attributes, which you specify as fields for the Blog Asset and Blog Author forms.

Blog authors are stored in a flat structure, under a single category named `BlogAuthors`:

- `Blog Author` is used to create author assets. Authors' names (and optionally images) will be displayed with the blogs they create.
- `Blog Author Category` is used to name the categories of blog authors. By default, there is only one author category, named `BlogAuthors`, which is used to contain all the author assets.
- `Blog Author Category Definition` is used to place blog author categories in hierarchical order. By default, there is only one blog author category definition (`BlogAuthorCategory`), which creates a flat structure (all blog author categories at the same level).
- `Blog Author Definition` is used to place author assets under the parent named `BlogAuthors`. The `Blog Author Definition` also specifies which blog attributes make up the Blog Author form.

Blog filters are used for data transformation:

- `Blog Filter` is used to take the data from one blog attribute, transform or evaluate it in some way, and then store the results in another blog attribute when you save the blog asset. The resulting value from a blog filter action is called a derived attribute value.

Flex Family Members

The rest of this chapter provides detailed descriptions of the default `Blog` flex family members:

Default Blog Attributes

The `Blog Attribute` asset type contains the following default attributes:

- `Abstract` – Defines the "Abstract" field for the blog asset form.
- `AssetId` – Is used to get the parent asset ID of the blog asset. Since the blog asset can be assigned to multiple blog categories under the same parent blog category, this attribute is used to find all blog categories to which the current blog asset belongs.
- `Author` – Defines the "Author" field of the blog asset form.
- `BlogAuthorFullName` – Defines the "Full Name" field of the blog author form.
- `BlogAuthorImage` – Defines the "Author Profile Image" field of the blog author form.
- `BlogAuthorImageHeight` – Is used to identify the default height of an uploaded author image.
- `BlogAuthorImageMimeType` – Is used by the Blog Author filters to identify the media type (for example, `.jpeg` or `.gif`) of an uploaded author image.
- `BlogAuthorImageWidth` – Is used to identify the default width of an uploaded author image.
- `BlogAuthorLargeThumbnailImage` – Is used to render the large thumbnail of the author image in the blog author form.
- `BlogAuthorLThumbHeight` – Is used to specify the appropriate height for the large thumbnail of the uploaded author image.
- `BlogAuthorLThumbWidth` – Is used to specify the appropriate width for the large thumbnail of the uploaded author image.
- `BlogAuthorSmallThumbnailImage` – Is used to render the small thumbnail of the uploaded author image.
- `BlogAuthorSThumbHeight` – Is used to specify the appropriate height for the small thumbnail of the uploaded author image.
- `BlogAuhtorSThumbWidth` – Is used to specify the appropriate width for the small thumbnail of the uploaded author image.
- `Body` – Defines the "Blog Body" field for the blog asset form.
- `Category` – Defines the "Blog Category" field for the blog asset form.

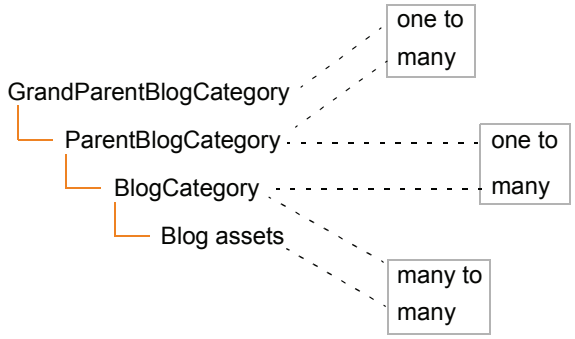
- `Date` – Defines the "Blog's Date and Time" field for the blog asset form.
- `GrandParentCategory` – Is used to retrieve the top most category to which the blog asset belongs. The `GrandParentCategory` encapsulates the parent and asset categories.
- `GrandParentDescription` – Contains the description for the grandparent category to which the blog asset belongs.
- `ParentCategory` – Is similar in concept to the `GrandParentCategory`, only it contains the category information about the parent.
- `ParentDescription` – Contains the description of the parent category to which the blog asset belongs.
- `Title` – Defines the "Blog Title" field for the blog asset form.
- `Username` – Is applied to the Blog Author Definition to retrieve the user name of the blog author for a given blog asset.

The default attributes can be used as a quick start for your own site's blog functionality. You can also create new attributes, and modify or delete the default attributes to fit your requirements. For more information, see "Customizing the Blog Asset Form," on page 40.

Default Blog Category Definitions and Blog Categories

The `Blog Category Definition` asset type contains the following definitions: `GrandParentBlogCategory`, `ParentBlogCategory`, and `BlogCategory`. Each definition specifies the level at which a category can be placed (Figure 2).

Figure 2: Category levels



`BlogCategory` defines the third level of categories. All blog assets belong to categories at the `BlogCategory` level. A given blog asset can belong to many blog categories under the same `ParentBlogCategory`.

The `Blog Category` asset type is used to name categories and place them in hierarchical order (by means of the blog category definitions described above). The `Blog Category` asset type contains the following grandparent categories: "Sports Blogs" and "Technology Blogs" (as shown in Figure 3). Each grandparent category can contain multiple parent categories. Each parent category can contain multiple blog categories, and each blog category can contain multiple blog assets. Figure 3, on page 23 displays a sample tree with the default categories and blog assets.

Note

Blog assets cannot have more than one distinct inherited value for a single-valued attribute. Therefore, if you are going to assign a blog asset to more than one blog category, make sure the blog categories are under the same parent blog category.

Figure 3: Sample blog category hierarchy

When creating blog categories for the blog assets of your website, you can reuse the hierarchy that is defined by the default blog category definitions. To display the hierarchy of blog categories and blog assets in Content Server's tree (as shown in Figure 3), you will need to create a blog tree tab. For instructions, see "Enabling Blog Asset Types," on page 54.

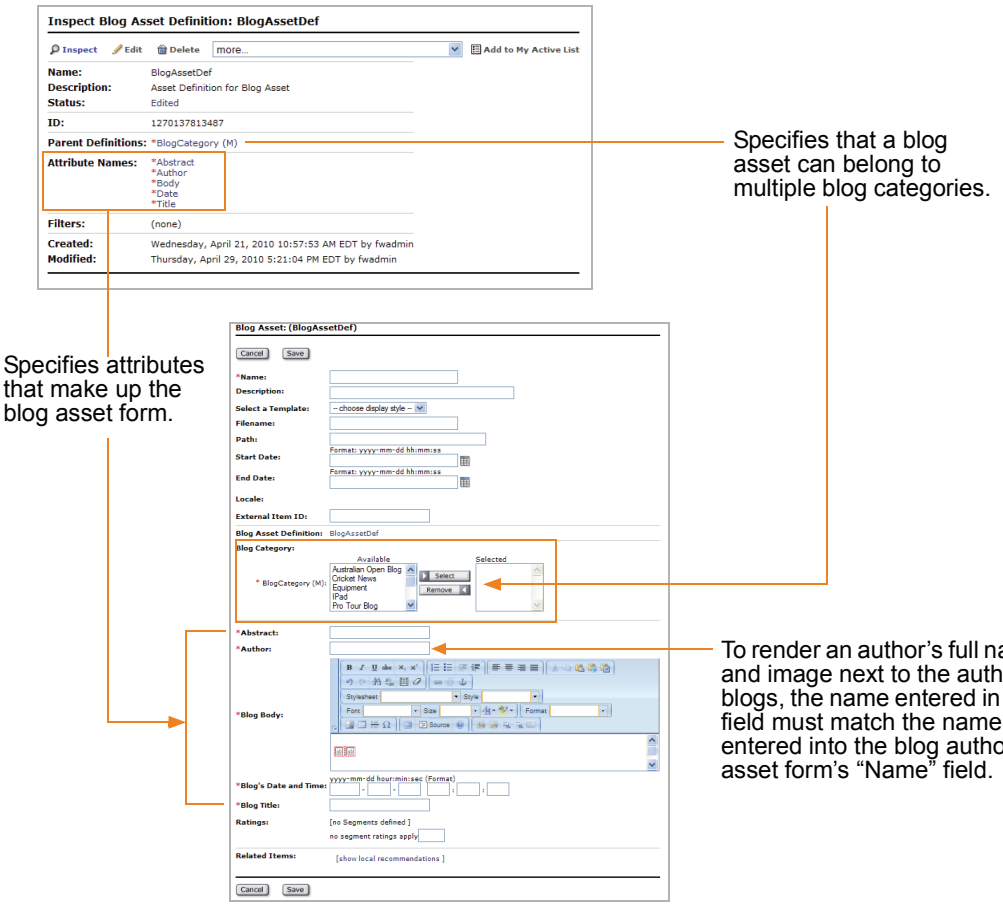
Default Blog Asset Definition and Blog Assets

The `Blog` flex family contains one blog asset definition, named `BlogAssetDef`, which specifies that a blog asset can belong to multiple blog categories (third-level categories) under the same parent blog category (second-level category). `BlogAssetDef` also specifies attributes that make up the blog asset's form:

- Abstract
- Author
- Body
- Date
- Title

The `Blog Asset` asset type supplies the form that content providers fill out to create blog assets. Figure 4 displays `BlogAssetDef` and shows how its parameters are used in the "Blog Asset" form.

Figure 4: Relationship between the blog asset definition and the blog asset form

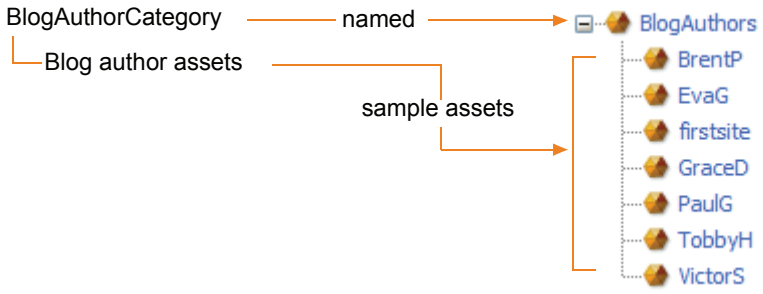


If you create your own attributes for the blog asset form, you will need to add those attributes to the default blog asset definition. For instructions on creating new attributes and adding them to the blog asset definition, see "Customizing the Blog Asset Form," on page 40.

Default Blog Author Category Definition and Blog Author Category

By default, the `Blog` flex family contains one blog author category definition, named `BlogAuthorCategory`, which defines a flat structure for storing blog authors. The `Blog Author Category` asset type is used to name the categories. Only one blog author category exists, named `BlogAuthors`, which is used to contain all the author assets created (or shared) on the site.

Figure 5: Sample blog author category hierarchy



Default Blog Author Definition and Blog Authors

The `Blog` flex family contains one blog author definition – `BlogAuthorDef` – which specifies that blog author assets can belong to only the "BlogAuthors" category. `BlogAuthorDef` also specifies attributes that make up the blog author form:

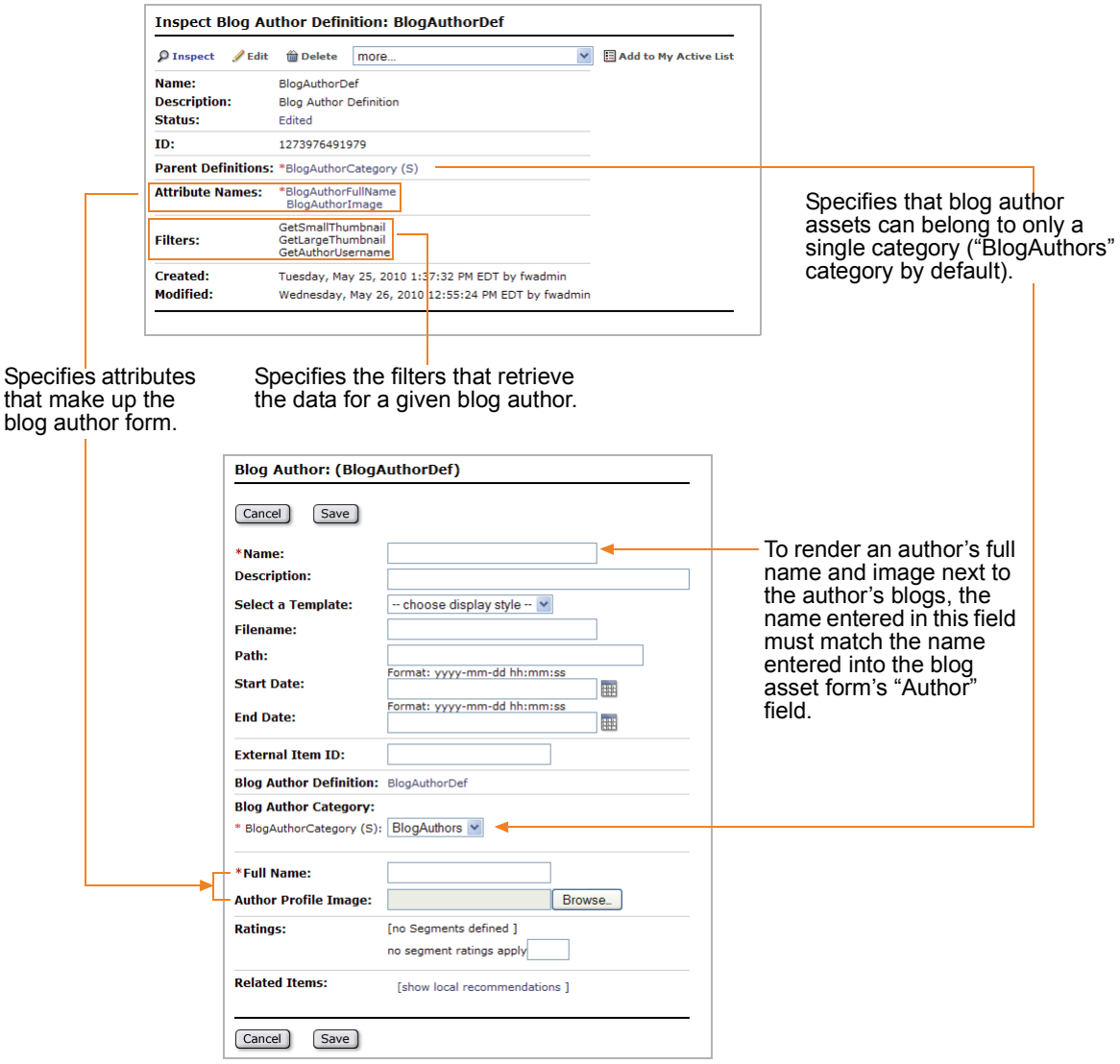
- Full Name
- Author Profile Image

In addition, `BlogAuthorDef` specifies filters that retrieve the author's full name, and large and small thumbnails of the image (optional).

The `Blog Author` asset type supplies the form that administrators fill out to create blog author assets for display next to published blogs. Blog authors can modify their names and images.

Figure 6, on page 26 displays `BlogAuthorDef` and shows how its parameters are used in the blog author form.

Figure 6: Relationship between the blog author definition and the blog author form



Default Blog Filters

The `Blog` flex family contains default filters, which use the field copier filter to copy the content of the system-defined attributes into user-defined flex attributes.

The default blog filters are:

- `GetAssetId` – Applied to the `Blog Category Definition` to retrieve the ID of the blog asset.
- `GetAuthorUsername` – Defined in the `Blog Author Definition` and applied to the `Blog Author` to dynamically generate the full name of a given blog's author.
- `GetCategory` – Applied to the `Blog Category Definition` to retrieve the appropriate category associated with the blog asset.
- `GetGrandParentCategory` – Applied to the `GrandParentBlogCategory` to retrieve the grandparent category associated with a given blog asset.
- `GetGrandParentDesc` – Applied to the `GrandParentBlogCategory` to retrieve the grandparent category's attributes.
- `GetLargeThumbnail` – Defined in the `Blog Author Definition` and applied to the Blog Author to dynamically generate the large thumbnail image of a given blog's author.
- `GetParentCategory` – Applied to the `ParentBlogCategory` to retrieve the parent category associated with a given blog asset.
- `GetParentDesc` – Applied to the `ParentBlogCategory` to retrieve the parent category's attributes.
- `GetSmallThumbnail` – Defined in the `Blog Author Definition` and applied to the `Blog Author` to dynamically generate the small thumbnail image of a given blog's author.

For instructions on creating additional blog filters, see the *Content Server Developer's Guide*.

Chapter 3

Sample Blog Pages

This chapter provides information about the default blog components that are supplied by the blog module. Out-of-the-box, these components render the sample blog pages.

This chapter contains the following sections:

Overview of the Sample Blog Pages

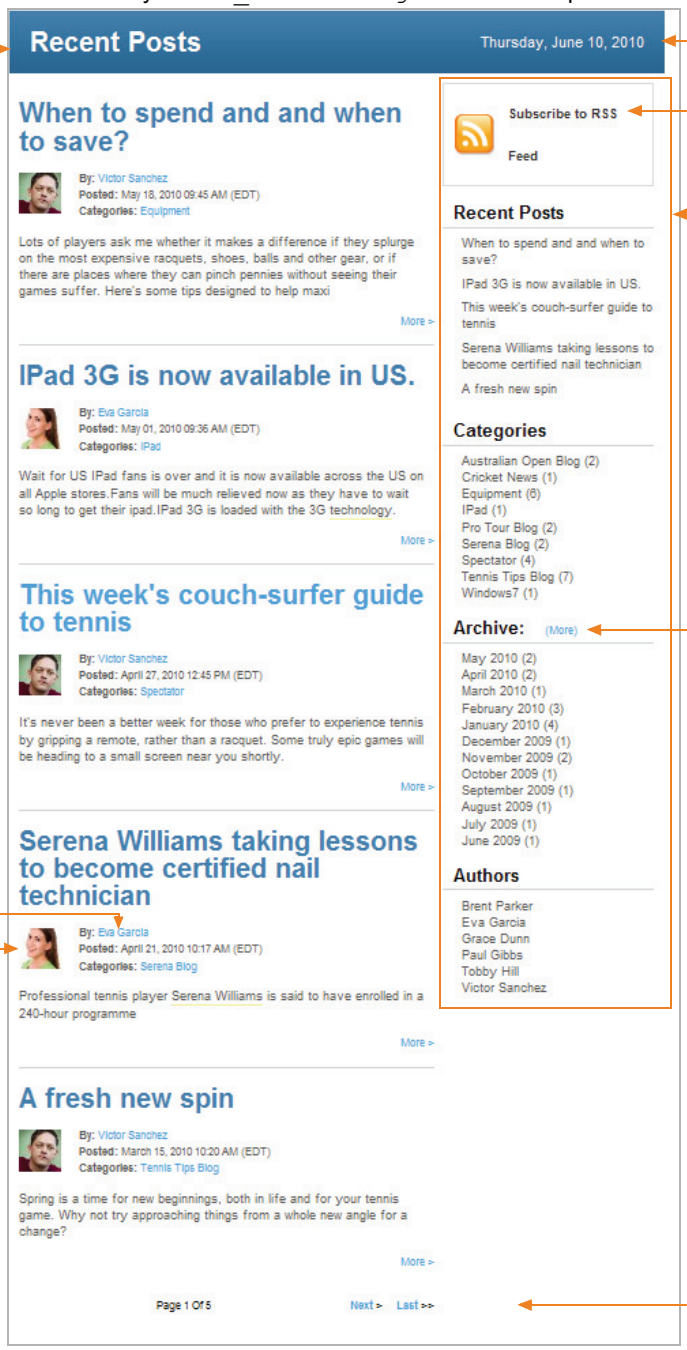
Blog pages are composed of blog assets, blog author assets, a right navigation panel, a header, and a footer. Blog pages are rendered by default blog templates, CSElements and a SiteEntry asset (as shown in Figure 7). By default, all blog pages are cached (except the wrapper page).

Figure 7: Sample home page

The `FW_RecentBlogs` page is the blog module's main page. It is called by the `FW_RecentBlogsDetail` template.

The wrapper element calls the sample layout template (`FW_BlogLayout`). The layout defines the overall appearance of the sample blog pages and calls the `FW_BlogContainer` template, which then calls the corresponding detail templates for the page, and for the asset types `Blog Category` and `Blog Asset`.

The wrapper's parameters (including the number of blogs that can appear on a page at one time) are defined by the default SiteEntry asset. The blog module's SiteEntry asset is rendered by the `FW_Blogwrapper` CSElement.



Blog header

RSS Feed rendered by the `FW_BlogRSS` template.

Right navigation panel renders the names of the blog pages and links to the blog assets on those pages.

The default blog pages are:

- Recent Posts
- Archive
- Authors

The **More** link is used to fetch another 12 months of content on top of what is displayed by default.

The full name of the blog's author is displayed next to her blog post(s).

Image of the blog's author displayed next to her blog post(s). This image is retrieved from the user's author asset. (The author asset is optional.)

Blog footer

Components of the Sample Blog Pages

This section provides information about the default blog templates, CSElements, and SiteEntry asset. These components are pre-configured with default parameters and code which render the sample blog pages.

Content Server renders the sample blog pages by executing the code associated with the sample page names. You can use the sample blog pages as a reference to create your own blog pages that will conform to the look and feel of your own website.

Note

Page assets cannot be shared between sites. If you wish to use the blog module on a different CM site, you must create new blog pages for those sites. All other components can be shared. If you wish to completely customize blog functionality on your sites, you can add the necessary blog code to your own site's components, and reconfigure them to fit your requirements. For instructions on creating pages, see "Creating Blog Pages," on page 43.

The sample blog pages can be found in Content Server's tree (**Site Plan** tab > **Unplaced Pages**). The sample pages are described below and shown in Figure 1, on page 8:

- `FW_RecentBlogs` – This page is the sample home page for the blog module and displays all recently published blogs.
- `FW_AuthorBlogs` – This page displays the blogs posted by a selected author. The blogs are rendered when you click the author's name in the "Authors" section of the right navigation panel.
- `FW_ArchiveBlogs` – This page displays all the blogs that have been posted in a given month. The blogs are rendered on this page when you click the month's name in the "Archive" section of the right navigation panel.

These pages are rendered by the `FW_BlogLayout` template. However, to navigate through the pages, start from the sample home page (`FW_RecentBlogs`), because this page initializes the parameters that are required for the other sample blog pages.

For information about the components of the sample blog pages, see the following topics:

- Default Blog CSElements
- Default Blog SiteEntry Asset
- Default Blog Templates
- Blog Archive 'More' Link Functionality

Default Blog CSElements

The blog module provides you with default CSElements, which store reusable code (in the form of XML or JSP and Java) that build the sample blog pages. For example, since the "Share" link is displayed on all three sample blog pages, a CSElement was created for it (`FW_Blogs/Utils/AddThisWidget`) which contains the full code for the link's functionality.

Table 1 lists and describes the blog module's default CSElements.

Table 1: Default blog CSElements

Name and ElementCatalog Entry	Description
`FW_BlogWrapper`	The wrapper for the blog module's sample pages.
`FW_Blogs/Utils/AddThisWidget`	Renders the "Share" link, which enables visitors to share blogs between websites.
`FW_Blogs/Utils/CreateArchiveLink`	Renders a link that calls the `FW_ArchiveBlogs` template.
`FW_Blogs/Utils/CreateAuthorLink`	Renders a link that calls the `FW_AuthorBlogsDetail` template.
`FW_Blogs/Utils/GetAssetData`	Retrieves the data for the blog asset. For example, this element can retrieve the attributes the blog asset inherits from its parent.
`FW_Blogs/Utils/GetAuthor`	Retrieves the full name and profile image of a given blog's author from that author's author asset (if any).
`FW_Blogs/Utils/GetBlogAuthors`	Retrieves the list of authors, and provides links to each author.
`FW_Blogs/Utils/GetBlogCategories`	Retrieves the list of categories, and provides links to each category.
`FW_Blogs/Utils/GetBlogsByMonth`	Retrieves the list of months, and provides links to each month.
`FW_Blogs/Utils/GetChildren`	Retrieves the children for a particular blog parent.
`FW_Blogs/Utils/GetExternalURL`	Creates the external URLs of the blog assets whose titles and summaries are listed in the RSS Feed. By default, this element creates the local Content Server URL. To modify the URL you must customize this element, see "Customizing URLs for the RSS Feed," on page 50.
`FW_Blogs/Utils/GetRecentBlogs`	Retrieves the list of recently published blogs, and provides links to their detail pages.

Default Blog SiteEntry Asset

The blog module's SiteEntry asset represents a pagelet and is associated with the `FW_Wrapper` CSElement, which is the root element that generates the sample pages' wrapper. The blog module's SiteEntry asset specifies the `blogsperpage` pagelet parameter, which defines the number of blogs that can be displayed on a page at one time.

Table 2 lists the default pagelet parameters for the `FW_BlogWrapper` SiteEntry asset.

Table 2: FW_BlogWrapper SiteEntry asset's pagelet parameters

Pagelet Parameter	Value
`blogsperpage`	`5`
`rendermode`	`live`
`seid`	*auto-generated*
`site`	*The name of the site on which you installed the blog module.*

The Number of Blogs Per Page

The number of blogs per page is passed as the parameter `blogsperpage` from the `FW_BlogWrapper` SiteEntry asset. If nothing is passed, the default value of `10` is used to specify the number of blogs per page. For instructions on specifying the number of blogs that can be displayed at one time on your own site's blog pages, see "Adding Blog Parameters to Your Site's SiteEntry Asset," on page 49.

Default Blog Templates

The blog module supplies you with default templates that render the sample blog pages. When a link is clicked, the corresponding template is called to render the content. For example, if you click the **More** link for a given blog post, the associated detail template renders the post's full content.

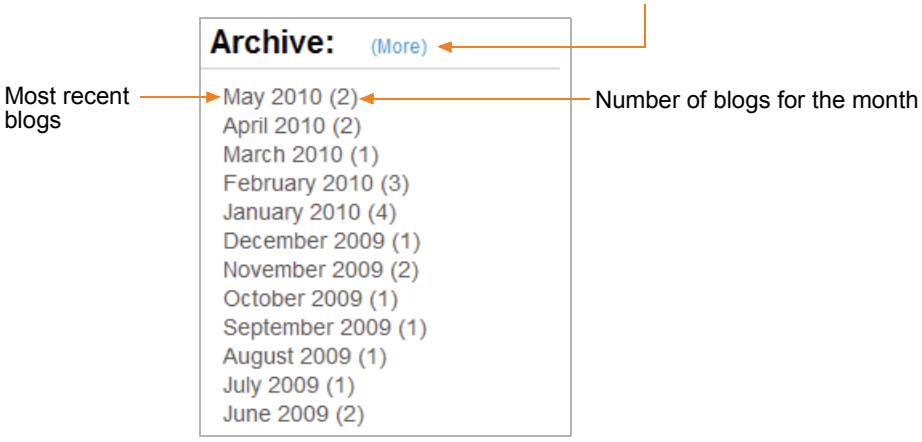
The page layout of your site is rendered by your own headers, footers, navigation, and content containers. To add blog pages to your website, you can add the necessary code specified in the blog module's default templates to your own site's templates. Table 3 lists and provides information about the blog module's default templates.

Table 3: Default blog templates

Template	Default Name	Asset Type	Description
Detail	FW_ArchiveBlogsDetail	Page	Used to render the blogs listed in the "Blogs Archive" page.
	FW_AuthorBlogsDetail	Page	Used to render the blogs listed under the "Blogs By Author" page.
	FW_BlogAssetDetail	FW_BlogAsset	Used to render the full content of a selected blog.
	FW_BlogCategoryDetail	FW_BlogParent	Used to render the blog categories listed under the "Categories" section of the right navigation panel.
	FW_RecentBlogsDetail	Page	Used to render the blog home page which lists all recently published blogs.
Container	FW_BlogContainer	Page	Used to call the detail templates.
	FW_BlogContainer	FW_BlogParent	
	FW_BlogContainer	FW_BlogAsset	
Blog Layout	FW_BlogLayout	N/A	Used to render the main layout of the sample blog pages.
Blog RSS	FW_BlogRSS	N/A	(Typeless) Used to render the FW_BlogSummaryRSS.
	FW_BlogSummaryRSS	FW_BlogAsset	(Typed) Renders the RSS feed on a Web page. Called by FW_BlogRSS.
Summary	FW_BlogSummary	FW_BlogAsset	Used to list the summaries of each blog asset. The summary of a blog asset includes the title, abstract, author's full name and profile image, date, and time.
Link	FW_Link	FW_BlogParent	Used to render the blog category links.
	FW_Link	FW_BlogAsset	Used to render the blog links.
Navigation	FW_Nav	FW_BlogParent	For each detail template there is a navigation link which internally calls all the elements (Recent Posts, Categories, Authors, and Archive).
	FW_Nav	FW_BlogAsset	
	FW_Nav	N/A	
	FW_Nav	Page	

Blog Archive ‘More’ Link Functionality

The **More** link, located next to the “Archive” section in the right navigation panel, is used to fetch additional data associated with the “Archive” page. By default, the “Archive” page lists the last 12 months of blogs. Clicking the **More** link multiple times retrieves another 12 months of blog data for every click.



The **More** link is coded in the blog module’s default layout template. For more information about the layout template’s code, see “Adding Blog Code,” on page 45.

Next Steps

So far this guide has provided information about the `Blog` flex family and the default components that make up the sample blog pages. The rest of this guide is procedural. For developers, Chapter 4, “Customizing Blog Components” contains procedures and code for adding blog functionality to custom pages. For administrators, Chapter 5, “Enabling Blog Components” contains instructions for enabling blog asset types for the content providers. Chapter 6, “Creating Blog Content” can be used for testing the data model and creating blog assets for publication.

Part 1

Developers

This part contains procedures for customizing the blog data model. It also covers template code and provides guidelines for developing blog functionality on different CM sites.

This part contains the following chapter:

- Chapter 4, “Customizing Blog Components

Chapter 4

Customizing Blog Components

This section provides general information and instructions for reconfiguring the default blog components to render custom pages.

This chapter contains the following sections:

Customizing the Blog Asset Form

The Blog flex family hierarchy can be modified to fit your requirements. This section provides instructions for creating a new blog attribute and adding that attribute to the blog asset definition to display the attribute as a field in the blog asset form.

Note

For your own reference, if you wish to see the hierarchical relationships between blog categories and blog assets, you can create a tree tab. For instructions on creating a tree tab, see "Enabling Blog Asset Types," on page 54.

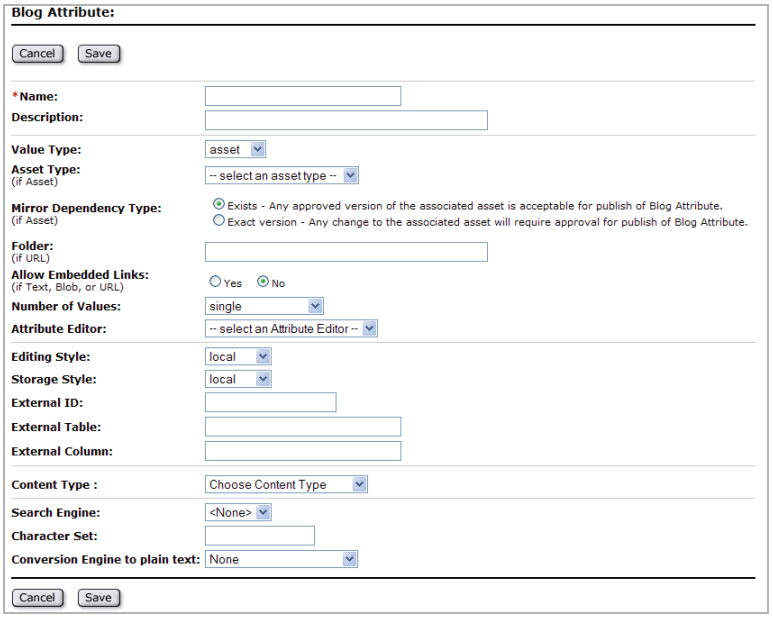
This section contains the following topics:

Creating a Blog Attribute

The blog attribute you create will be displayed as a field in the blog asset form once you add the attribute to the blog asset definition.

To create a blog attribute

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select the site on which the blog module is enabled.
3. In the button bar, click **New**.
4. Click **New Blog Attribute**.
5. In the "Blog Attribute" form, fill in the fields:



Note

The fields you need to fill in can differ significantly based on the data type that you select for your attribute. If you need more information about a given field, see the *Content Server Developer's Guide*.

- **Name** – Enter a name of up to 64 characters (the name cannot contain spaces).
- **Description** – Enter a short summary that describes the use or function of the attribute.
- **Value Type** – Select a data type for this attribute.
- **Asset Type** – If the attribute is of type `asset`, select an asset from the drop-down list.
- **Mirror Dependency Type** – If the attribute is of type `asset`, select a dependency type.
- **Folder** – (Optional) If the attribute is of type `blob`, enter a path to the directory that you want to store the attribute values in.
- **Allow Embedded Links** – If the attribute is of type `text`, `blob`, or `URL`, select whether links to other pages or websites can be embedded in the attribute's content field.
- **Number of Values** – Select either **single** or **multiple** from the drop-down list, depending on the data type selected for the **Value Type** field.
- **Attribute Editor** – (Optional) If you do not wish to use the default input type, click in the **Attribute Editor** field to select the appropriate attribute editor for the field.
- (Optional) **Character Set** – If you need to override the default ISO character set (ISO-8859-1), enter the character set you want to use for this attribute.

If you are creating a foreign attribute (keeping data in an external system) fill in the following fields:

- **Editing Style** – If you want this attribute to be available to users in its native table on the external system, select **external**.
- **Storage Style** – Select **external**. For more information, see the *Content Server Developer's Guide*.
- **External ID** – Specify the name of the column that serves as the primary key for the table that holds this foreign attribute (the column that uniquely identifies the attribute).
- **External Table** – Enter the name of the table that stores this attribute.
- **External Column** – Enter the name of the column in the table specified in the **External Table** field that holds the value of the attribute.

6. Click **Save**.

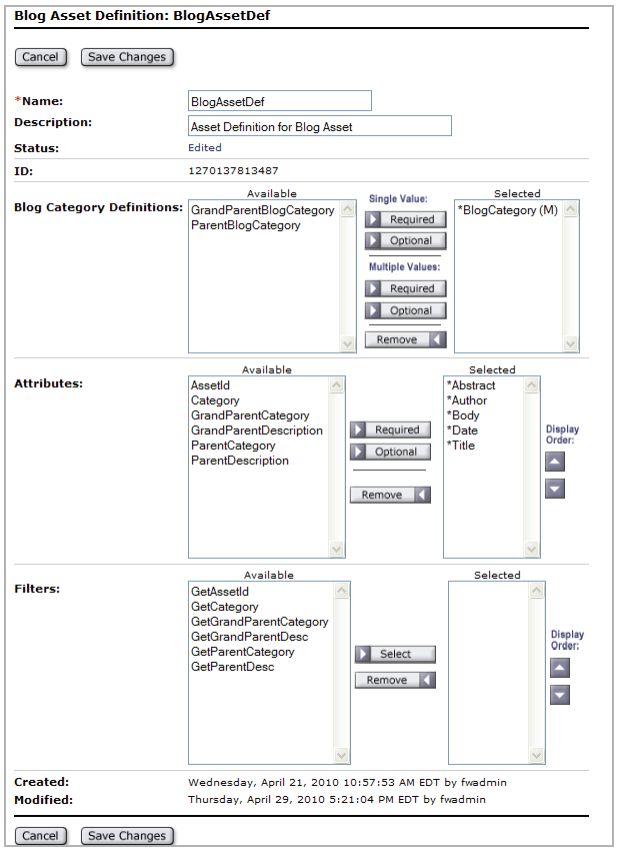
   Now that you have created the attribute, add the attribute to the blog asset definition. For instructions, see the next section.

Adding a Blog Attribute to the Blog Asset Definition

To add an attribute to the blog asset form you must add the attribute to the blog asset definition.

To add an attribute to the blog asset definition

1. Log in to the Content Server Advanced interface as a general administrator, and select the site on which the blog module is enabled.
2. Access the blog asset definition's "Inspect" form:
   a. In the button bar, click **Search**.
   b. In the "Search" form, click **Find Blog Asset Definition**, and click **Search**.
   c. Select **`BlogAssetDef`**.
3. In the "Inspect" form, click **Edit**.



4. In the "Attributes" field, select the attribute(s) from the "Available" list box and use the **Required** or **Optional** button to move the attribute(s) to the "Selected" list box. Which button you choose determines whether the attribute(s) will be required or optional in the blog asset form.
5. Click **Save**.

   The attributes you selected are now included as fields in the blog asset form. When a user creates a blog asset, the new attributes will be displayed as either required or optional fields.

Adding Blog Functionality to CM Sites

If you wish to use the blog module on a different CM site, you must create new blog pages for those sites. Adding blog functionality to your website entails the following steps:

1. Create pages on the content management site that will be used to render blog assets on the website.
2. Copy the blog code from the default blog templates and CSElements to your site's templates and CSElements. How your site is set up determines the modifications you must make to the default code once you insert it into your templates and CSElements.
3. Add the `blogsperpage` parameter that is specified in the blog module's SiteEntry asset to your own site's SiteEntry asset.

This section contains the following topics:

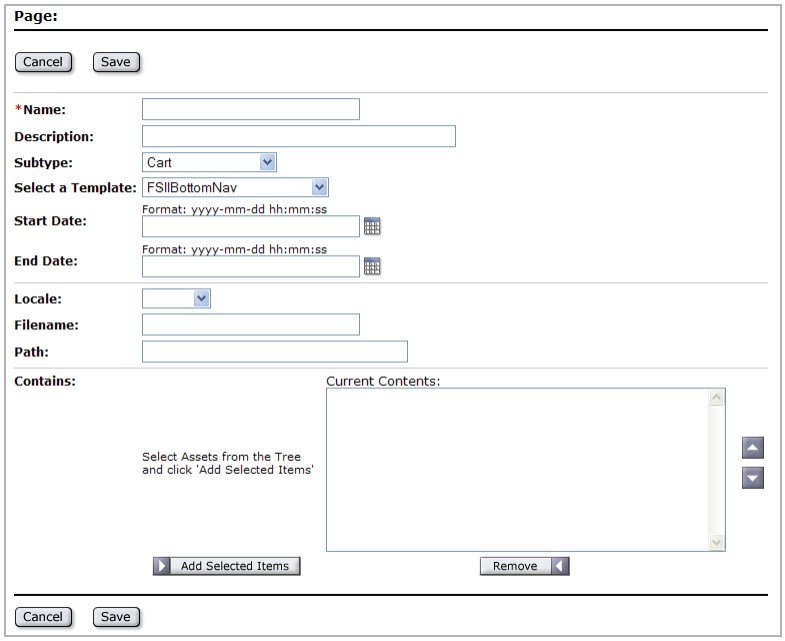
Creating Blog Pages

Before creating blog pages, map out their types: the main blog page (which is the `FW_RecentBlogs` page in the blog module), category pages, and so on. Also determine your site's graphical, navigational, and functional features in order to create blog pages that will conform to the layout of your website. For more information, see the *Content Server Developer's Guide*.

To create blog pages

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select the site on which you wish to create the pages for displaying blogs to website visitors.
3. In the button bar, click **New**.
4. Click **New Page**.

5. In the "Page" form, fill in the fields:



- **Name** – Enter a name of up to 64 characters.
- **Description** – Enter a short, descriptive summary about the page.
- **Select a Template** – Select the template that will render the page.
- **Start Date** and **End Date** – If the page must be available for only a certain period of time, set the date range in which the page will be available on the website.
- **Contains** – Add items from your active list to the "Current Contents" box.

6. Click **Save**.

Now that you have created a page to display blog assets, code your site's templates and CSElements to call the new page and render blog functionality on your website. For more information, see "Adding Blog Code," on page 45.

Adding Blog Code

The blog module is configured to render sample blog pages. Your own site's layout is likely to differ from the layout of the sample blog pages. For example, your site may call a left navigation, while the sample blog pages call a right navigation. Instead of coding your templates from scratch to incorporate blog functionality, you can reuse the sample code by inserting it into your own templates and CSElements, then reconfiguring the code as necessary.

Note

Your site's wrapper element renders the layout of your site. To ensure that your wrapper element renders blog pages in addition to existing pages, copy the relevant blog code from the default `FW_Wrapper` element to your site's wrapper element.

To add blog code to your site's layout template

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select the site on which the blog module was installed.
3. Access the `FW_BlogLayout` template:
   a. In the button bar, click **Search**.
   b. In the "Search" list, select **Find Template**.
   c. In the "Search" field, enter **`FW_BlogLayout`**, and then click **Search**.
   d. Click **FW_BlogLayout**.
4. In the layout template's "Inspect" form, click **Edit**.
5. Copy the necessary code from the blog module's layout template and insert it into your own layout template:

   The following lines retrieve the site description from the wrapper and load the site:

```
<ics:if
   condition='<%=ics.GetVar("tid")!=null%>'><ics:then><render:logdep
   cid='<%=ics.GetVar("tid")%>' c="Template"/></ics:then></ics:if>

<publication:load name='Publication' field="name"
   value='<%=ics.GetVar("site")%>'/>
<publication:get name='Publication' field="id" output="spubid"/>
<publication:get name='Publication' field="description"
   output="pubdesc"/>
```

   The following line retrieves the body of the page:

```
String sContainerTName = "FW_BlogContainer";
```

The following lines load the site, page, and asset descriptions:

```
String sTitle = "";
       if (Utilities.goodString( ics.GetVar("p") )) {
              %><asset:load name='t1' type='Page'
                 objectid='<%=ics.GetVar("p")%>' /><%
              %><asset:get name='t1' field='name' output='t1Name' /><%
              %><asset:get name='t1' field='description'
                 output='t1Desc' /><%
              sTitle += ":
                 "+(Utilities.goodString(ics.GetVar("t1Desc")) ?
                 ics.GetVar("t1Desc") : ics.GetVar("t1Name"));
           }
           if (!"Page".equals(ics.GetVar("c")))
           {
              %><asset:load name='t2' type='<%=ics.GetVar("c")%>'
                 objectid='<%=ics.GetVar("cid")%>' /><%
              %><asset:get name='t2' field='name' output='t2Name' /><%
              %><asset:get name='t2' field='description'
                 output='t2Desc' /><%
              sTitle += ":
                 "+(Utilities.goodString(ics.GetVar("t2Desc")) ?
                 ics.GetVar("t2Desc") : ics.GetVar("t2Name"));
           }
```

The following line calls the style sheet for the blog layout template. The style sheet defines the look and feel of the sample blog pages. Because your site already has its own style sheet, copy the parameters you need from the blog module's style sheet and insert them into your own site's style sheet. Make sure you resolve any conflicts between the blog module's style sheet and your own style sheet:

```
<render:callelement elementname="FW_Blogs/CSS/blogsCSS"/>
```

The following lines are the JavaScript method that retrieves the **More** link for the "Archive" blogs page:

```
<script type="text/javascript">
          function getMoreBlogs(url){
              var xhtReq = getXMLHttpRequest();
              xhtReq.open("GET", url, true);
              xhtReq.onreadystatechange = function(){
                 if(xhtReq.readyState==3){
                     document.getElementById('moreLink').innerHTML='
                    <img src="<%=ics.GetSSVar("baseurl")%>images/
                    wait_ax_tiny.gif"/>';
                 } else if(xhtReq.readyState==4){
                     document.getElementById('archiveDiv').innerHTML =
                        xhtReq.responseText;
                 }
              };
              xhtReq.send(null);
           }

           function getXMLHttpRequest() {
              try { return new XMLHttpRequest(); } catch(e) {}
              try { return new ActiveXObject("Msxml2.XMLHTTP"); }
                 catch (e) {}
              alert("XMLHttpRequest not supported");
```

```
            return null;
        }

        </script>
```

The following lines call the main blog page or the body of a given asset. (These lines also call the header and footer for the blog sample pages. Since your site has its own headers and footers, copy only the code you require for your site's blog functionality, and retain your own site's headers and footers):

```
<!--main start-->
      <div id="main">
         <!--header start-->
         <%-- <div id="header">
            <!--Call header from here-->
         </div> --%>
         <!--header end container start-->
         <div id="container">
            <!--Body  -->
            <%-- Call the container template for the current page
               subtype  --%>
               <render:calltemplate tname='<%=sContainerTName%>'
                  site='<%=sSite%>' tid='<%=ics.GetVar("tid")%>'
                  slotname="BlogBodyContainer"
                    c='<%=ics.GetVar("c")%>'
                       cid='<%=ics.GetVar("cid")%>' ttype="Template">
                    <render:argument name="p"
                       value='<%=ics.GetVar("p")%>' />
                    <render:argument name="locale"
                       value='<%=ics.GetVar("locale")%>'/>
                    <render:argument name="packedargs"
                       value='<%=ics.GetVar("packedargs")%>'/>
                   <render:argument name="site"
                       value='<%=ics.GetVar("site")%>'/>
                   <render:argument name="spubid"
                       value='<%=ics.GetVar("spubid")%>'/>
                   <render:argument name="blogsperpage"
                       value='<%=ics.GetVar("blogsperpage")%>'/>
               </render:calltemplate>
         </div><!-- End of container -->
      </div><!-- End of main -->
      <!-- Footer -->
      <%-- <div id="footer">
         <!--Call footer from here-->
      </div> --%>
</body>
```

6. Reconfigure the blog code you inserted into your site's layout template to fit your requirements.
7. Inspect the code of the other default blog templates and CSElements and copy the relevant sections into your own templates and CSElements.

Adding Community Server Functionality

If the blog module was integrated with Community Server, the tags for the comments widget and comments link were added to the `FW_BlogSummary` and `FW_BlogAssetDetail` templates. To add Community Server functionality to custom pages, copy the widget tags from those templates to your custom templates.

To add a comments widget to a page, copy the following lines from the default blog templates to the custom templates:

```
<%
boolean cosEnabled =
   Boolean.parseBoolean(ics.GetProperty("fwblogs.cos.enabled",
   "futuretense_xcel.ini", true));
if(cosEnabled){
String cosHost = ics.GetProperty("cos.hostname", "futuretense_xcel.ini",
   true);
String cosPort = ics.GetProperty("cos.portnumber",
   "futuretense_xcel.ini", true);
%>
<div id="comments_container"></div>
<script type="text/javascript">
if(typeof cos == "undefined")
{
    var cos = {};
    cos.pageWidgets = new Array();
}else if(typeof cos.pageWidgets == "undefined")
{
    cos.pageWidgets = new Array();
}
cos.pageWidgets.push({name:"comments", version:"0.1", attributes:
   {"site_id":'<%=ics.GetVar("site")%>',"resource_id":'<%=ics.GetVar("cid
   ")%>'}, element:document.getElementById("comments_container")});
</script>
<script type="text/javascript" src="http://<%=cosHost%>:<%=cosPort%>/cos-
   service/widgets/<%=ics.GetVar("site")%>/comments/1.0.js"></script>
<%}%>
```

To add the comments link widget to a page, copy the following lines from the default blog templates to the custom templates:

```
boolean cosEnabled =
   Boolean.parseBoolean(ics.GetProperty("fwblogs.cos.enabled",
   "futuretense_xcel.ini", true));
if(cosEnabled){
String cosHost = ics.GetProperty("cos.hostname", "futuretense_xcel.ini",
   true);
String cosPort = ics.GetProperty("cos.portnumber",
   "futuretense_xcel.ini", true);%>
<script type="text/javascript">
    if(typeof cos == "undefined")
    {
        var cos = {};
    }
    if(typeof cos.pageWidgets == "undefined")
    {
        cos.pageWidgets = new Array();
    }
```

```
    cos.pageWidgets.push({name: "comments-link",
        version: "0.1",
        elementID: "comments_link_div",
        attributes:
           {"site_id":'<%=ics.GetVar("site")%>',"resource_id":'<%=ics.Get
           Var("cid")%>',"count_template":"Comments (#count)"}});
</script>
<script type="text/javascript" src="http://<%=cosHost%>:<%=cosPort%>/cos-
   service/widgets/<%=ics.GetVar("site")%>/comments_link/0.1.js"></
   script>
```

Adding Blog Parameters to Your Site's SiteEntry Asset

The blog module's SiteEntry asset specifies the `blogsperpage` parameter, which enables you to specify the number of blog assets that can be displayed on a Web page at one time. You can specify this parameter in your own site's SiteEntry asset.

Note

If you do not specify the `blogsperpage` parameter in your site's SiteEntry asset, then the default number of blogs per page, which is 10, will be used. For information about the `blogsperpage` parameter, see "Default Blog SiteEntry Asset," on page 33.

To add the `blogsperpage` parameter to your custom SiteEntry asset

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select the site to which you are adding blog functionality.
3. In the button bar, click **Search** to find your site's SiteEntry asset.
   a. In the "Search" form, click **Find SiteEntry**.
   b. Click **Search**.
   c. Select your site's SiteEntry asset.
4. In the SiteEntry asset's "Inspect" form, click **Edit**.
5. In the "Pagelet Parameters" field, add the following:
   - **Name** – Enter `blogsperpage`.
   - **Value** – Enter the number of blogs that can be displayed on a page at one time.
6. Click **Save**.

Customizing URLs for the RSS Feed

The right navigation panel of the sample blog pages contain an RSS Feed link. When a visitor clicks the RSS Feed link, it renders an up-to-date listing of the titles and summaries of the blog assets that are published to the sample blog pages. When a visitor clicks the title of a blog, the entire content of the selected blog is rendered. External URLs for the blog assets included in the RSS Feed are created by the `GetExternalURL` CSElement. By default this element creates the local Content Server URL:

```
http://<host name>:<port number>/<application context>/
   <path to file>
```

where `<host name>` is the host name of the Content Server installation from which the assets of the RSS Feed are accessible, `<port number>` is the port number of the Content Server application, and `<application context>` is the context of the Content Server application on which the blog module is running.

If you wish to use the RSS Feed with your own site's external URLs, you must modify the `GetExternalURL` element's URL string, and add the parameters listed in Table 1 to the `futuretense_xcel.ini` file with the values for the host information and context of your site. The `GetExternalURL` element reads these parameters from the `futuretense_xcel.ini` file to create the external URLs for the blog assets listed in your site's RSS Feed.

Table 1: Parameters read by `GetExternalURL` element to create external URLs

Parameter	Description
`fwblogs.hostscheme`	Specifies the top level of the URL naming structure. For example, `http`.
`fwblogs.hostname`	Specifies the host name of the Content Server installation from which the assets of the RSS Feed are accessible.
`fwblogs.portnumber`	Specifies the port number of the Content Server application.
`fwblogs.contextinfo`	Specifies the context of the Content Server installation on which the blog module is running.

Next Steps

This chapter provides developers with guidelines and procedures for adding blog functionality to custom pages. The next chapter, Chapter 5, "Enabling Blog Components" provides administrators with instructions for enabling the data model for content providers. Instructions in Chapter 6, "Creating Blog Content" can be used to test the blog data model, to create blog pages for publication, and to work with Community Server functionality on blog pages.

Part 2

Administrators

This part provides administrators with instructions for enabling content providers to use the blog module to create blogs and modify their author profiles.

This part contains the following chapter:

- Chapter 5, "Enabling Blog Components"

Chapter 5

Enabling Blog Components

This chapter provides instructions for enabling the blog components your content providers will need to access in order to create blog content. It also provides instructions for creating author assets.

This chapter contains the following sections:

Enabling Blog Asset Types

If you wish to use the blog module on different CM sites you must enable the blog asset types on those sites. When you enable the blog asset types you can also enable their start menu items. Start menu items allow content providers to create and search for assets of the types that you enable.

To enable blog asset types

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select a site.
3. In the tree, select the **Admin** tab, expand the **Sites** node, expand the node of the site for which you wish to enable the blog asset types, and expand the **Asset Types** node.
4. Double-click **Enable**.
5. In the "Enable Asset Types" form, select the check boxes of the blog asset types. For descriptions of the blog asset types, refer to Chapter 2, "The Blog Module's Data Model."
6. Click **Enable Asset Types**.
7. In the start menu selection form, click **Enable Asset Types**.
8. Authorize users to access the start menu items. For instructions, see the *Content Server Administrator's Guide*.

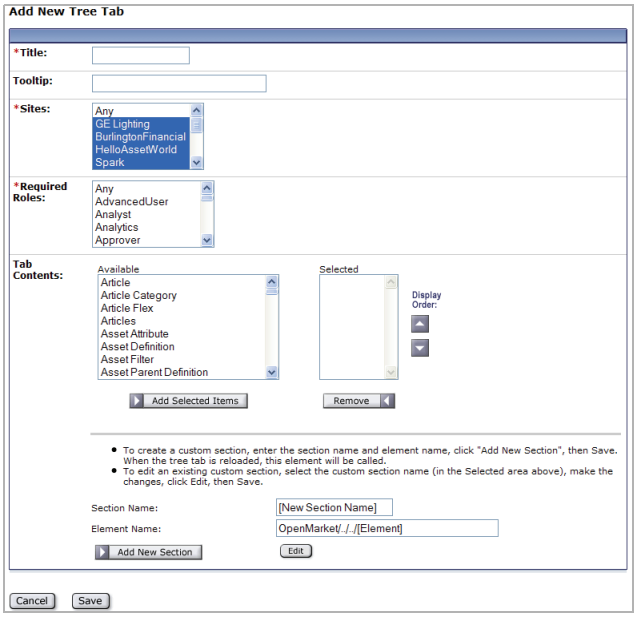
Creating a Blog Tree Tab

A tree tab can be configured to display the hierarchy of blog categories and their assets.

To create a blog tree tab

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select a site.
3. In the **Admin** tab, double click the **Tree** node.
4. Click **Add New Tree Tab**.

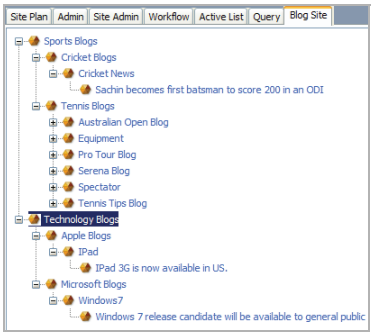
5. In the "Add New Tree Tab" form, fill in the fields:



- **Title** – Enter a short, descriptive name, of up to 64 characters.
- **Tooltip** – Enter a short, informative description of the tab, up to 255 characters. This description is displayed when a user mouses over the tab.
- **Site** – Select the sites that will display this tab. (The blog asset types must be enabled on these sites. For instructions, see "Enabling Blog Asset Types," on page 54.)
- **Required Roles** – Select the roles a user must be assigned to access this tab.
- **Tab Contents** – Select the asset types that will be displayed on the tab and then click **Add Selected Items**. Select **Blog Category** to display your blog categories and assets in hierarchical order (as shown in Figure 3, on page 23).
- (Optional) If you want to add custom functionality to this tab, use the **Section Name** and **Element Name** fields. For more information, see the *Content Server Developer's Guide*.

6. Click **Save**.

   The new blog tab is displayed in the tree (refresh your browser, if necessary).



Sharing Blog Assets

You can share blog flex assets with other sites. All types of blog assets must be shared.

Note

The blog asset definition is required to define the attributes that make up the blog asset form.

To share blog assets

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select a site.
3. Locate the asset you wish to share:
   a. In the button bar, click **Search**.
   b. In the "Search" form, click the asset type for which you wish search.
   c. Click **Search**.
4. In the search results list, click the asset you wish to share.
5. In the blog asset's "Inspect" form, click the "more..." drop-down menu, and select the option to share the asset with another site.
6. In the "Share to:" field, select the site(s) with which you wish to share the blog asset you selected.
7. Click **Save Changes**.

   The blog asset you selected to share is now available on the site(s) with which you shared it.

For detailed information about sharing assets, see the *Content Server Advanced Interface User's Guide*.

Creating Workflows for Blog Assets

Workflows can be used if you wish to manage the work on a blog asset. Before a blog asset can be published, it must be approved for publishing. The workflow feature routes blog assets through the steps that you determine as necessary, from creation to approval.

For example, if a blog asset must be reviewed by an editor before it can be approved for publishing, Content Server's workflow feature can route that blog asset to the appropriate user at the appropriate time. For more information about workflow processes, see the *Content Server Administrator's Guide*.

Working with Blog Authors

On the sample blog pages, the author's full name and image are displayed next to the author's blog posts. This section provides instructions for creating blog author assets, which contain author information. This section also provides instructions for granting users access to their own blog author assets.

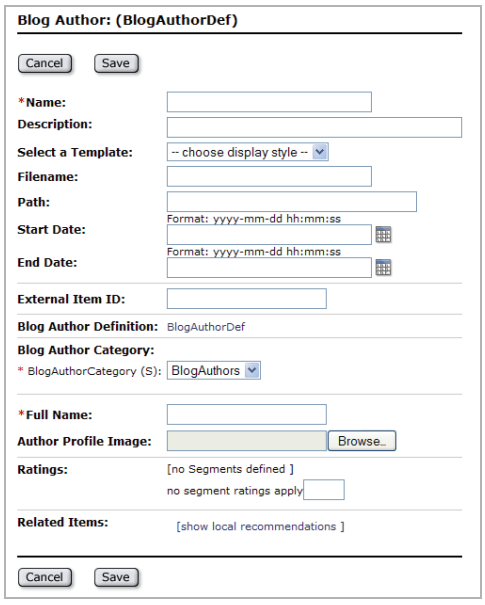
This section contains the following topics:

Creating Author Assets

To display a blog author's full name and image next to the author's blog posts, you must create an author asset in which you specify that user's full name and optionally, an image.

To create an author asset

1. Log in to the Content Server Advanced interface as a general administrator.
2. Select the site on which the blog asset types are enabled.
3. In the button bar, click **New**.
4. Click **New Blog Author**.
5. In the "Blog Author" form, fill in the fields:



- **Name** – Enter a name for the blog author asset.

> **Note**
>
> When the author creates a blog asset, she must enter the same name into the blog asset's "Author" field. Otherwise, the author asset you have created will not be published to its allocated space.

- **Description** – Enter a short, descriptive summary about the author asset (for example, the author's full name).
- **Blog Author Category** – By default, this field is single-valued. The only available value for this field is `BlogAuthors`.
- **Full Name** – Enter the blog author's full name. This name will be displayed on the blogs that this author creates.
- **Author Profile Image** – Click **Browse**, and select an image for the content provider.

6. Click **Save**.

Setting Access Permissions to a Blog Author Asset

A blog author asset contains the profile of the content provider who creates blog content for the website. Setting access permissions for a blog author asset entails the following steps:

1. Create a unique role for the blog author.
2. Assign the new role to the blog author.
3. Grant the blog author access to the blog author asset.

The same steps are presented in detail below.

To create a unique role for the blog author

1. Log in to the Content Server Advanced interface as a general administrator, and select a site.
2. In the tree, select the **Admin** tab, expand the **Roles** node, and double-click **Add New**.

   The "Add New Role" form opens:

   - **Name** – Enter a unique name for the role.
   - **Description** – Enter a short, descriptive sentence explaining what the role will do.

3. Click **Save**.

   Now you must assign the role to the content provider who requires access permissions to the blog author asset.

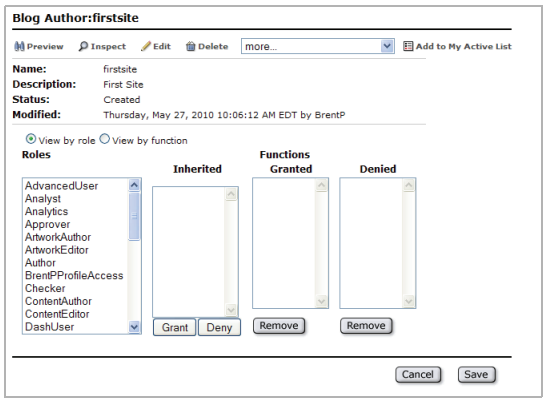
To assign the role to the blog author

4. Select the **Admin** tab, expand the **Sites** node, and then expand the node of the author's site. Double-click **Users**.
5. In the "Username" field, enter the name of the user to which you wish to assign the role, and click **Select**.
6. Click the edit (pencil) icon.
7. In the "Edit Roles for User" form, select the role you created in steps 2 - 3.

8. Click **Save**.

To set access permissions to a blog author asset

9. Access the blog author asset to which you want to set permissions:
   a. In the button bar, click **Search**.
   b. Click **Find Blog Author**.
   c. Click **Search**.
10. Select the blog author asset.
11. In the blog author's "Inspect" form, select the "more..." drop-down menu, and click **Access Permissions**.



12. Select the **View by role** radio button.
13. In the "Roles" list box, select the role you created in steps 2 - 3 on page 58.
14. In the "Inherited" list box, select all the functions (**Shift-click** the first and last functions), and click **Grant**.

    All the functions in the "Inherited" list box move to the "Granted" list box.
15. Click **Save**.

The only user who can work with this blog author asset, besides the administrator, is the blog author who was assigned the selected role.

Publishing Blog Assets

If you are planning to publish a blog asset, you must pre-approve its author asset. This is to ensure that the author's profile is published with the blog asset.

Next Steps

This chapter provides administrators with instructions for enabling content providers to create and publish blog assets. The next part in this guide contains Chapter 6, "Creating Blog Content," which can be used to test and further develop the blog data model, to create blog pages for publication, and to work with Community Server functionality on blog pages.

Part 3

Content Providers

This part contains instructions for creating blog content, modifying your profile, and working with Community Server, assuming it is integrated with the blog module.

This part contains the following chapter:

- Chapter 6, "Creating Blog Content"

Chapter 6

Creating Blog Content

This chapter provides instructions for creating blog assets and categories, and modifying blog author assets. It also provides instructions for working with the comment fields that are added to your blog pages if Community Server was integrated with the blog module.

This chapter contains the following sections:

Creating Blog Assets

Blog assets are the units of content that are displayed on the website. You can create blog assets using the Content Server Advanced or Dash interface.

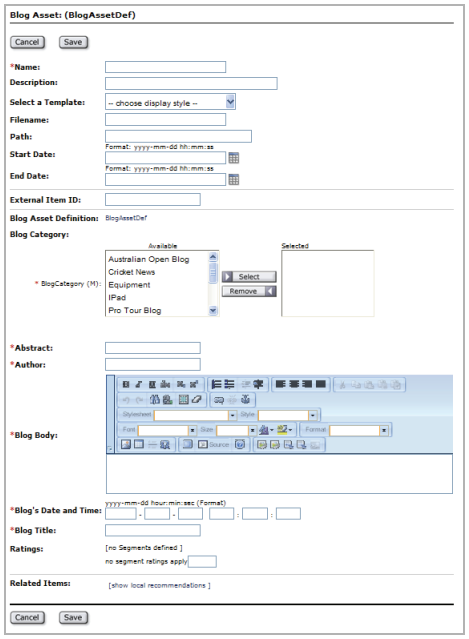
This section contains the following topics:

Creating Blog Assets in the Advanced Interface

The Advanced interface is used mainly for site management. If you have access to the Advanced interface, then you can create blog assets using this interface. Otherwise, you must use the Content Server Dash interface.

To create a blog asset in the Advanced interface

1. Log in to the Content Server Advanced interface with your assigned credentials.
2. Select the site on which your blog start menu items are enabled.
3. In the button bar, click **New**.
4. In the "New" list, click **New Blog Asset**.



- **Name** – Enter a name for the blog asset (up to 64 characters).
- **Description** – Enter a short descriptive sentence about the blog asset.
- (Optional) **Start Date** and **End Date** – If the blog asset must be available for only a certain period of time, set the period by specifying the start and end dates.
- **Blog Category** – Select the category (or categories) for the blog asset.
- **Abstract** – Enter a summary for the blog asset. This summary will be displayed on the website.

- **Author** – Do one of the following:
  - If a blog author asset was created for you, enter the name of the asset (to access the asset, follow the steps in "Editing Your Blog Author Asset from the Advanced Interface," on page 69). When your blog asset is published with your blog author asset, your full name and image (both contained in the blog author asset) will be displayed next to your blog post.
  - If a blog author asset was not created for you, enter the name you want to be displayed next to your blog post.
- **Blog Body** – Enter the content of your blog.
- **Blog's Date and Time** – Enter the date and time the blog asset was created. This date and time will be displayed under the blog asset to website visitors.
- **Blog Title** – Enter the title of the blog. This title will be displayed on the website to visitors.

5. Click **Save**.

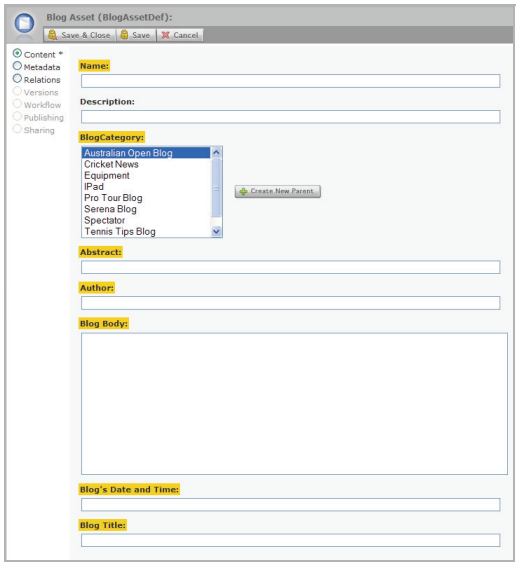
   This asset must be approved and published in order for its content to be displayed to website visitors. If a blog author asset exists, make sure that asset, as well, is published. For more information about approval and publishing, see the *Content Server Advanced Interface User's Guide*.

Creating Blog Assets in the Dash Interface

The Dash interface is Content Server's editorial interface. You can create blog assets in the Dash interface if you are an authorized user.

To create a blog asset in the Dash interface

1. Log in to the Content Server Dash interface with your assigned credentials.
2. Select the site on which the blog start menus are enabled.
3. In the "Create New..." drop-down menu, select **New Blog Asset**.
4. In the "Blog Asset" form, fill in the fields:



- **Name** – Enter a name for the blog asset (up to 64 characters).
- **Description** – Enter a short descriptive sentence about the blog asset.
- **Blog Category** – Select one or more categories for the blog asset.
- **Abstract** – Enter a summary for the blog asset. This summary will be displayed on the website.
- **Author** – Do one of the following:
  - If a blog author asset was created for you, enter the name of the asset (to access the asset, follow the steps in "Editing Your Blog Author Asset from the Dash Interface," on page 69). When your blog asset is published with your blog author asset, your full name and image (both contained in the blog author asset) will be displayed next to your blog post.
  - If a blog author asset was not created for you, enter the name you want to be displayed next to your blog post.
- **Blog Body** – Enter the content of your blog.
- **Blog's Date and Time** – Enter the date and time the blog asset was created. This date and time will be displayed under the blog asset to website visitors.
- **Blog Title** – Enter the title of the blog. This title will be displayed on the website to visitors.

5. (Optional) Select the blog asset's **Metadata** option. If the blog asset must be displayed for a certain period of time, use the **Start Date** and **End Date** fields to set the date range.
6. Click **Save & Close**.

   The asset must be approved and published to the website in order for its content to be displayed to website visitors. If a blog author asset exists, make sure that asset, as well, is published. For more information about approval and publishing, see the *Content Server Dash Interface User's Guide*.

Creating a Blog Category

When creating blog assets, you may want to create categories that are relevant to the content of the new blogs. You can create new blog categories in the Content Server Advanced and Dash interfaces.

For example, if your site will contain blogs about new products your company is coming out with, you may want to create a "New Products" category. All blogs pertaining to a new product would be associated with this category.

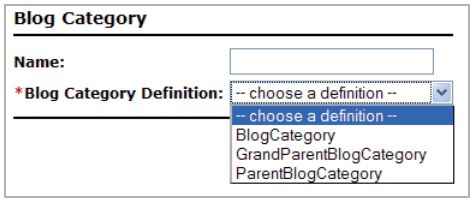
This section contains the following topics:

Creating a Blog Category in the Advanced Interface

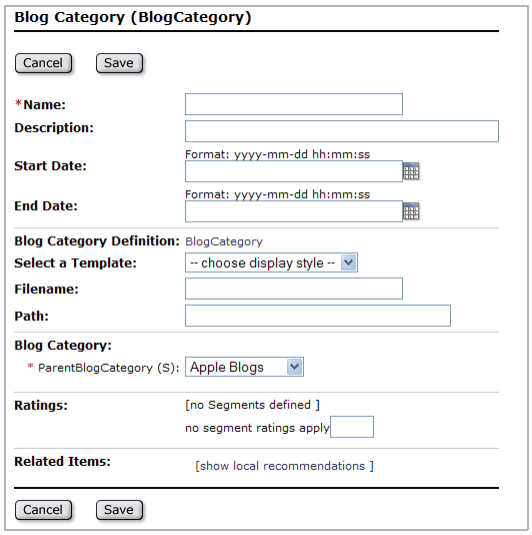
The Advanced interface is used mainly for site management. If you have access to the Advanced interface, then you can create blog category assets using this interface. Otherwise, you must use the Content Server Dash interface.

To create a blog category in the Advanced interface

1. Log in to the Content Server Advanced interface with your assigned credentials.
2. Select the site on which your blog start menu items are enabled.
3. In the button bar, click **New**.
4. In the menu items list, click **New Blog Category**.
5. In the "Blog Category" form, select the level of category you are creating:



   a. Fill in the fields:
      - **Name** – Enter a name for the new blog category (for example "New Products").
      - **Blog Category Definition** – Select a definition from the drop-down list, `BlogCategory` in this example (`BlogCategory` places the "New Products" category at the third level). For more information about the default options, see Chapter 2, "The Blog Module's Data Model."
   b. Click **Continue**.
6. In the next form, name the category and fill in the remaining fields:



   - **Description** – Enter a brief description about the category.

- **Start Date** and **End Date** – If the category must be available for only a certain period of time, set the period by specifying the start and end dates.
- **Select a Template** – Select the template that will render this category's content.
- **Blog Category** – Select the name of your category's parent (one level above the category you are creating. In our example, the parent you select in this field becomes the grandparent of blog assets that will be associated with the category.)

7. Click **Save**.

   The blog category asset must be approved and published to the website in order for it to be displayed to website visitors. For more information, see the *Content Server Advanced Interface User's Guide*.

Creating a Blog Category in the Dash Interface

The Dash interface is Content Server's editorial interface. You can create blog category assets in the Dash interface if you are an authorized user.

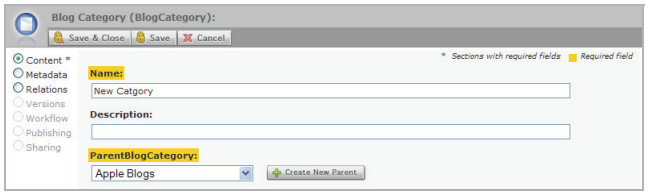
Creating a blog category in the Dash interface

1. Log in to the Content Server Dash interface with your assigned credentials.
2. Select the site on which the blog start menu items are enabled.
3. In the "Create New..." drop down list, select **New Blog Category**.
4. In the "Create New Blog Category" form, select the level of category you are creating:

   a. Fill in the fields:
      - **Name** – Enter a name for the category.
      - **Blog Category Definition** – Select a definition from the drop-down list, `BlogCategory` in this example (`BlogCategory` places the "New Products" category at the third level). For more information about the default options, see Chapter 2, "The Blog Module's Data Model."

   b. Click **Next**.

5. In the next form, name the category and set the remaining fields.



6. Click **Save & Close**.

   The blog category asset must be approved and published to the website in order for it to be displayed to website visitors. For more information, see the *Content Server Dash Interface User's Guide*.

Editing Your Blog Author Asset

An administrator of your system may have created an author asset for you, which is used to render your full name and image next to the blogs you post. This author asset is your blog profile; it contains your full name and profile image. You can modify the information in this author asset from the Advanced or Dash interface. Any changes you make are automatically generated on the Web pages that contain your blogs.

This section contains the following topics:

Editing Your Blog Author Asset from the Advanced Interface

The Advanced interface is used mainly for site management. If you have access to the Advanced interface, then you can modify your blog author asset using this interface. Otherwise, you must use the Content Server Dash interface.

To edit your blog author asset from the Advanced interface

1. Log in to the Content Server Advanced interface with your assigned credentials.
2. Select the site on which your blog start menu items are enabled.
3. Access your author asset:
   a. In the button bar, click **Search**.
   b. Click **Find Blog Author**.
   c. Click **Search**.
   d. Select your author asset.
4. In your blog author asset's "Inspect" form, click **Edit**.
5. Make the desired changes to the fields, and click **Save Changes**.

   If you changed the full name or profile image, then the changes will be automatically generated next to your blog posts.

Editing Your Blog Author Asset from the Dash Interface

The Dash interface is Content Server's editorial interface. You can modify your blog author asset in the Dash interface if you are an authorized user.

To edit your blog author asset from the Dash interface

1. Log in to the Content Server Dash interface with your assigned credentials.
2. Select the site on which your blog start menu items are enabled.
3. Access the blog author asset to which you have been granted permissions:
   a. In the "Search Criteria..." drop-down menu, select **Find Blog Author**.
   b. Click **search**.
   c. From the search results list, select your blog author asset.
4. In the author asset's "Inspect" form, click **Edit**.
5. Make the desired changes to the fields, and click **Save & Close**.

If you changed the full name or profile image, then the changes will be automatically generated next to your blog posts.

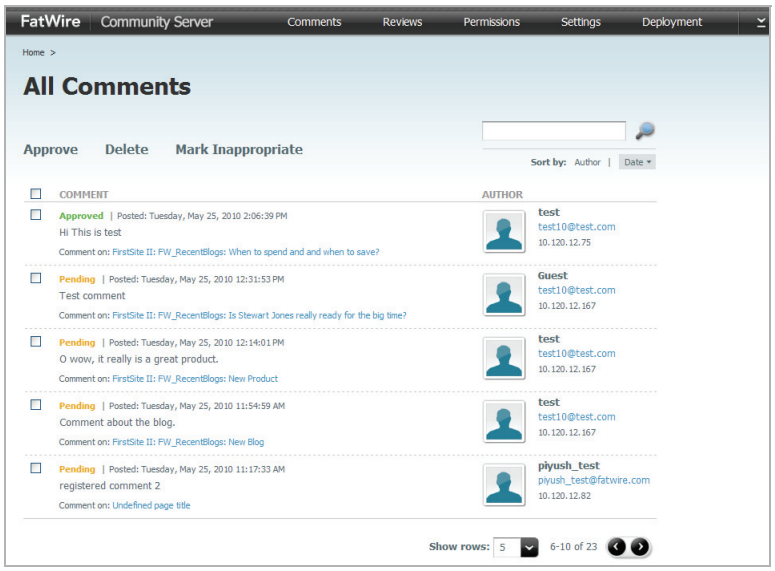
Reviewing Visitors' Comments on Blog Assets

If your system is integrated with Community Server and your blog pages are tagged with Community Server's comment tags, then the pages display comment fields, where visitors can contribute feedback on blog content. Comment links may also be included. If Community Server is configured for manual comment moderation, then all comments on blog assets will await approval from the moderators. If you are a moderator you can approve, delete, or edit comments, or mark them as inappropriate.

To review comments that visitors posted to blog assets

1. Log in to the WEM Admin interface with administrator or moderator credentials.
2. Select the site on which Community Server is enabled, and select the **Community Server** icon.

The Community Server Admin interface's "All Comments" screen opens:



3. (Optional) If you want to review comments posted about a specific blog asset, select **Comments > Comments by Topic**, and select the blog title whose comments you wish to review.

To approve a comment

1. Mouse over the comment you wish to approve, and click **Approve**.

   The comment is now displayed in the "All Comments" list with a status of "Approved." The comment is also displayed on the website under the blog to which it was posted.

To modify a comment before approving it

1. Mouse over the comment you wish to edit, and click **Edit**.
2. In the "Comment" field, modify the content of the comment. Click **Save**.

   The comment is now displayed in the "All Comments" list with the modifications you made.

To delete a comment

1. Mouse over the comment you wish to delete, and click **Delete**.
2. In the warning box, click **OK**.

   The comment is removed from the "All Comments" list and deleted from the system.

To mark a comment as inappropriate

1. Mouse over the comment that contains the inappropriate content, and click **Mark Inappropriate**.

   The comment is displayed in the "All Comments" screen with a status of **Inappropriate**.

For more information about working with comments in the Community Server Admin interface, see the *Community Server User's Guide*.