Using Sitefinity's Amazon S3 Blob Storage Provider

# Table of Contents

Table of Contents 1

Overview 2

Code and Notable Features 3

Using the Sitefinity Amazon S3 Blob Storage Provider in your Sitefinity project 4

A Helper JavaScript Bookmarklet 7

How to use the code: 7

What this bookmarklet can do for you when you click it: 8

The script 11

# Overview

Sitefinity's Amazon S3 Blob Storage Provider is an implementation of a cloud blob storage provider, which stores the binary blob data of Sitefinity's library items on Amazon's Simple Storage Service (S3). This document describes how to use the provider.

Note that only the binary data of the items in a library are stored on the remote blob storage. Sitefinity still manages its logical items by library with their regular meta-data properties (title, description etc) in its own database.

For additional documentation about Sitefinity's Blob storage and Blob storage providers, please refer to Sitefinity's documentation at: <http://www.sitefinity.com/documentation/documentationarticles/developers-guide/deep-dive/blob-storage-providers>

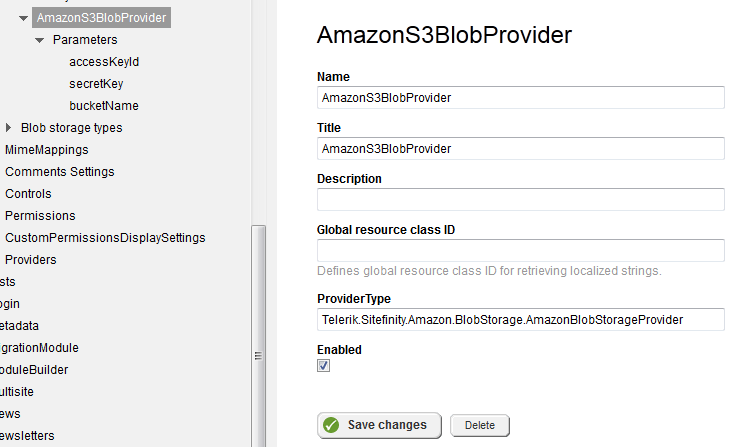
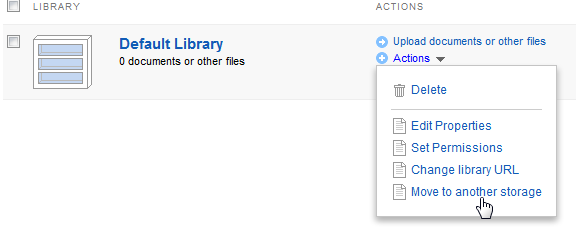
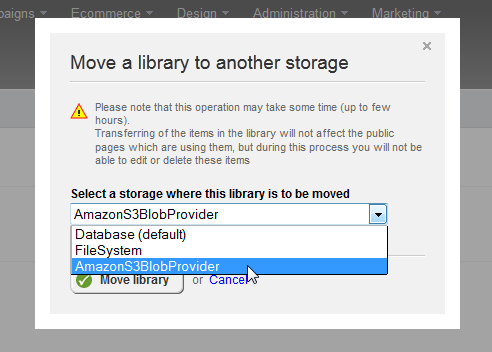
For additional information about Amazon's S3 and to obtain the Amazon S3 SDK, please refer to: <http://aws.amazon.com/s3/>

For debug (and other) purposes, you may also interact with your Amazon S3 with external clients, such as CloudBerry's Amazon Explorer for Amazon S3 Free tool: <http://www.cloudberrylab.com/free-amazon-s3-explorer-cloudfront-IAM.aspx>

# Code and Notable Features

* AmazonBlobStorageProvider inherits Sitefinity's CloudBlobStorageProvider class.
* Once the overridden InitializeStorage method is called, the provider uses values from its configured parameters (see how to set those below) in order to establish the secured connection to the right storage bucket.
* Headers of items which are uploaded by the provider are set to include "x-amz-acl" with its value set to "public-read". The result of this setting is that the items in the storage are publicly accessible (for reading) by everyone who holds the item's URL.
* Sitefinity's Amazon S3 provider points to a specific storage bucket (its name is set on the provider's parameters). Items which are uploaded to Sitefinity's library are stored in that bucket, under a folder which matches the name of the library.

# Using the Sitefinity Amazon S3 Blob Storage Provider in your Sitefinity project

1. Build the assembly:
   1. Build the Telerik.Sitefinity.Amazon project.
   2. Add the built binary Telerik.Sitefinity.Amazon.dll as a reference to your SitefinityWebApp project.
2. Register the provider in Sitefinity's configuration:  
   
   1. On Sitefinity's Backend's main menu go to the advanced Settings:  
      Administration 🡪 Settings and click Advanced.
   2. On the tree navigate to Libraries 🡪 Blob storage providers and click Create new.
   3. Fill in the Name and Title for your provider.  
      In the ProviderType field, enter the assembly qualified name of your provider's class.  
      Click Save changes.
   4. On the tree navigate to your newly created provider 🡪 Parameters and click Create new.  
      In the Key field enter "accessKeyId", in the Value field enter your Amazon S3 Access Key.  
      Click Save changes.
   5. On the tree navigate again to your newly created provider 🡪 Parameters and again click Create new.  
      In the Key field enter "secretKey", in the Value field enter your Amazon S3 Secret Key.  
      Click Save changes.
   6. On the tree navigate again to your newly created provider 🡪 Parameters and again click Create new.  
      In the Key field enter "bucketName", in the Value field enter the name of the Amazon S3 bucket which you wish to associate with this provider.  
      Click Save changes.
3. Associate a library with Amazon S3 Blob storage:
   1. On Sitefinity's Backend's main menu go to one of the library modules:  
      Content 🡪 Images / Videos / Documents & Files.
   2. Click Manage libraries.
   3. On the Libraries grid, choose the library you wish to associate a library with Amazon S3 Blob storage. Expand the Actions menu and select Move to another storage.  
      
   4. In the drop-down list, on the dialog which pops up, select your Amazon S3 Blob Provider and click Move library.  
      

Now the binary data of items in your selected library will be persisted in Amazon's S3 storage instead of Sitefinity's database.

# A Helper JavaScript Bookmarklet

The following bookmarklet code may help setting up the provider in Sitefinity's backend, guiding the administrator where to navigate to, what to click, and setting the right names of the variables.  
It's been tested to work with Google Chrome v23, Firefox v17.0 (not on Internet Explorer).

## How to use the code:

The easiest way to use a bookmarklet is to create a bookmark and set its URL to contain the script.

**In Firefox:**

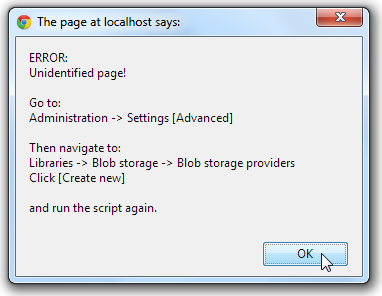
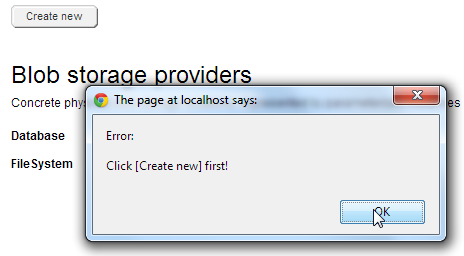
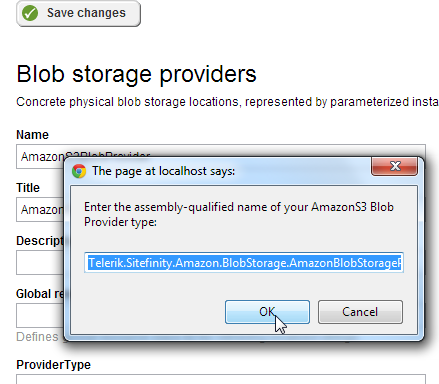
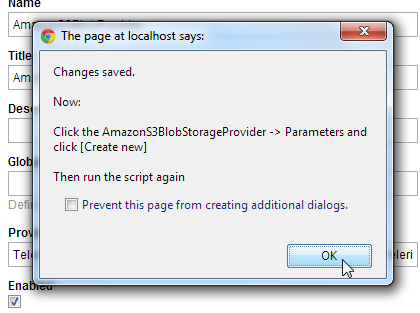
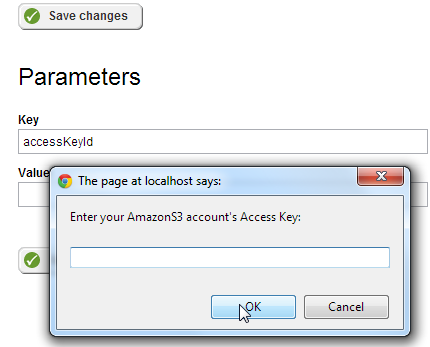
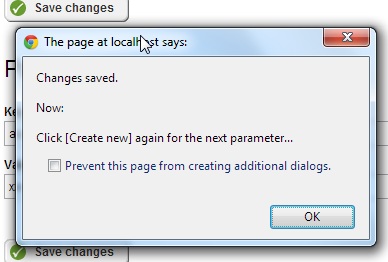
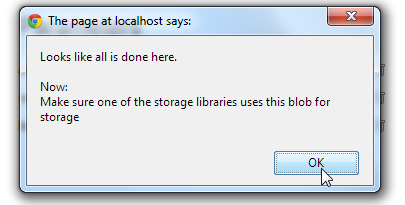
1. Create an arbitrary bookmark (no matter to which page).
2. Right-click the created bookmark and in the context menu select Properties.
3. In the Name field enter a title for the bookmarklet. E.g. "Create AmazonS3 Blob Storage Provider".
4. In the Location field paste the whole code listed below.
5. Click Save.
6. To use the bookmarklet, simply click it.

**In Google Chrome:**

1. Create an arbitrary bookmark (no matter to which page).
2. Right-click the created bookmark and in the context menu select Edit...
3. In the Name field enter a title for the bookmarklet. E.g. "Create AmazonS3 Blob Storage Provider".
4. In the URL field paste the whole code listed below.
5. Click Save.
6. To use the bookmarklet, simply click it.

Once you have the bookmarklet in place, click it anytime to get further instructions. After every step (creating the provider, navigating to the parameters, creating each parameter) you can click it again and you'll be prompted what to do next.

## What this bookmarklet can do for you when you click it:

* Guide you where to start…  
  
* Instruct you on your next step  
  
* Help you fill data  
  
* And tell you what to do next  
  
* Help you create parameters with the right names  
  
* Guide you on  
  
* Analyze when you when you're finished. And advise you on the next step  
  

## The script

|  |
| --- |
| javascript: (function () { function getElementByPartialNameOrId(partialid, tagname) { var re = new RegExp(partialid, 'g'); if (tagname == '' || tagname == null) tagname = '\*'; var el = document.getElementsByTagName(tagname); for (var i = 0; i < el.length; i++) { if ((el[i].id && el[i].id.match(re)) || (el[i].name && el[i].name.match(re))) { /\*alert(el[i].id);\*/ return (el[i]); } } return null; } function GetElementByPartialTextContent(textContent, tagname) { var retVal = null; var el = document.getElementsByTagName(tagname); for (var i = 0; i < el.length; i++) { if (el[i].textContent.toUpperCase().indexOf(textContent.toUpperCase()) >= 0) { retVal = el[i]; break; } } return retVal; } function trySaveChanges(messageAfter) { var saveButton = getElementByPartialNameOrId("btnSave", "a"); if (!saveButton) { alert("Couldn't find the [Save changes] button!\nFind it and click it yourself! Then:\n\n" + messageAfter); } else { SaveChanges(); alert("Changes saved.\n\nNow:\n\n" + messageAfter); } } var msgMakeSureStorageIsUsed = "Make sure one of the storage libraries uses this blob for storage"; var msgErrPressCreateNew = "Error:\n\nClick [Create new] first!"; var msgErrUnidentifiedPageUnknownStatus = "ERROR:\nUnidentified page!\n\nGo to:\nAdministration -> Settings [Advanced]\n\nThen navigate to:\nLibraries -> Blob storage -> Blob storage providers\nClick [Create new]\n\nand run the script again."; var msgErrUnidentifiedPageProviderVisible = "ERROR:\nUnidentified page (although I see you've already created the provider...)!\n\nGo to:\n\nLibraries -> Blob storage -> Blob storage providers -> AmazonS3BlobProvider -> Parameters\nClick [Create new]\n\nand run the script again."; var msgAllDone = "Looks like all is done here.\n\nNow:\n"; var msgClickNewForAnotherParam = "Click [Create new] again for the next parameter..."; var msgActionCancelled = "Action cancelled"; var partialCreateNewButtonId = "linkCreateItem"; var blobStorageHeader = null; var parametersHeader = null; var headers = document.getElementsByTagName("h2"); if (headers) { for (var i = 0; i < headers.length; i++) { if (headers[i].textContent == "Blob storage providers") { blobStorageHeader = headers[i]; } else if (headers[i].textContent == "Parameters") { parametersHeader = headers[i]; } } } var createButton = getElementByPartialNameOrId(partialCreateNewButtonId, "a"); var providerAlreadyCreated = GetElementByPartialTextContent("AmazonS3BlobProvider", "span"); var accessKeyIdParamCreated = GetElementByPartialTextContent("accessKeyId", "span"); var secretKeyParamCreated = GetElementByPartialTextContent("secretKey", "span"); var bucketNameParamCreated = GetElementByPartialTextContent("bucketName", "span"); if (!createButton || createButton.style.display != "none") { if (!providerAlreadyCreated) { if (!blobStorageHeader) { alert(msgErrUnidentifiedPageUnknownStatus); } else { alert(msgErrPressCreateNew); } } else { if (!parametersHeader) { alert(msgErrUnidentifiedPageProviderVisible); } else { if (providerAlreadyCreated && accessKeyIdParamCreated && secretKeyParamCreated && bucketNameParamCreated) { alert(msgAllDone + msgMakeSureStorageIsUsed); return; } else { alert(msgErrPressCreateNew); } } } } else { if (blobStorageHeader) { if (!providerAlreadyCreated) { var providerName = document.getElementById("Value0"); if (providerName) { providerName.value = "AmazonS3BlobProvider"; } var providerTitle = document.getElementById("Value1"); if (providerTitle) { providerTitle.value = "AmazonS3BlobProvider"; } var providerType = document.getElementById("Value4"); if (providerType) { var assemblyProviderNameProviderType = prompt("Enter the assembly-qualified name of your AmazonS3 Blob Provider type:\n", "Telerik.Sitefinity.Amazon.BlobStorage.AmazonBlobStorageProvider, Telerik.Sitefinity.Amazon, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null"); if ((assemblyProviderNameProviderType == null) || (assemblyProviderNameProviderType == "")) { alert(msgActionCancelled); return; } else { providerType.value = assemblyProviderNameProviderType; trySaveChanges("Click the AmazonS3BlobStorageProvider -> Parameters and click [Create new]\n\nThen run the script again"); return; } } } } if (parametersHeader) { if (!accessKeyIdParamCreated) { var accessKeyIdParamName = document.getElementById("Value0"); if (accessKeyIdParamName) { accessKeyIdParamName.value = "accessKeyId"; } var accessKeyIdParamValue = document.getElementById("Value1"); if (accessKeyIdParamValue) { var accesskeyval = prompt("Enter your AmazonS3 account's Access Key:\n", ""); if ((accesskeyval == null) || (accesskeyval == "")) { alert(msgActionCancelled); return; } else { accessKeyIdParamValue.value = accesskeyval; trySaveChanges(msgClickNewForAnotherParam); return; } } } if (!secretKeyParamCreated) { var secretKeyParamName = document.getElementById("Value0"); if (secretKeyParamName) { secretKeyParamName.value = "secretKey"; } var secretKeyParamValue = document.getElementById("Value1"); if (secretKeyParamValue) { var secretkeyval = prompt("Enter your AmazonS3 account's Secret Key:\n", ""); if ((secretkeyval == null) || (secretkeyval == "")) { alert(msgActionCancelled); return; } else { secretKeyParamValue.value = secretkeyval; trySaveChanges(msgClickNewForAnotherParam); return; } } } if (!bucketNameParamCreated) { var bucketNameParamName = document.getElementById("Value0"); if (bucketNameParamName) { bucketNameParamName.value = "bucketName"; } var bucketNameParamValue = document.getElementById("Value1"); if (bucketNameParamValue) { var bucketnameval = prompt("Enter your AmazonS3 account's Bucket Name:\n", ""); if ((bucketnameval == null) || (bucketnameval == "")) { alert(msgActionCancelled); } else { bucketNameParamValue.value = bucketnameval; trySaveChanges(msgMakeSureStorageIsUsed); return; } } } else alert(msgAllDone + msgMakeSureStorageIsUsed); } else { if (providerAlreadyCreated && accessKeyIdParamCreated && secretKeyParamCreated && bucketNameParamCreated) { alert(msgAllDone + msgMakeSureStorageIsUsed); } else { alert(msgErrUnidentifiedPageUnknownStatus); } } } })() |