QueryStringModule for ACA Description

# Introduction

This document describes a set of ASP.NET customizations developed by Woolpert Inc. for the City of Torrance, CA for use with their Accela Citizen Access deployment. It gives a bit of background about the original issue, describes the basic operation of the solution, and describes system dependencies and deployment. Finally, a brief section on troubleshooting concludes the document.

# Background

The City of Torrance, CA is using Accela Citizen Access. One of the features of Accela Citizen Access is that it provides a rule-based HTTP handler named “urlRouting.ashx”. When the right URL parameters with appropriate values are passed to this HTTP handler, it allows the creation of “deep links” that can be distributed, for example in an email.

When the deep link is clicked, the user is navigated directly to a record details page. This is a great convenience, and simplifies navigation for the user.

However, the HTTP handler makes certain assumptions about the system that is generating these deep links. It assumes that the following values are known when the deep link is generated:

* capId1
* capId2
* capId3
* module

Unfortunately, not all systems generating deep links have access to this information. A system creating a link may only have access to a records “AltID” field, for instance.

This mismatch in the parameters known to the system generating the deep links during runtime, and the parameters required by the HTTP handler makes it possible to create deep links that are not recognized by the HTTP handler.

# Basic Operation

The solution that was developed to handle the mismatch in parameters available to systems generating deep links, and the receiving HTTP handler, is a new ASP.NET HTTP module.

An ASP.NET HTTP module is a \*.dll that can be registered to participate in a site’s ASP.NET HTTP Request/Response pipeline through configuration. That is, the \*.dll file for the module, along with any dependent libraries, is deployed to an ASP.NET website’s bin directory. It’s registered by making appropriate changes in the web site’s web.config file, and from that point onwards will inspect either incoming requests, outgoing responses, or both, depending on the definition of the module. In this case, the HTTP module being described simply listens to incoming requests.

The file provided to Torrance that contains this HTTP module is named “Woolpert.ACAModules.dll”. This library contains a single HTTP module named “QueryStringModule”.

When QueryStringModule is called, it is passed the context of the incoming request as an event handler argument. When the event handler is called, it inspects the request’s “QueryString” object, which represents the query string parameters contained in the incoming request’s URL.

Ifit sees that the request contains a URL parameter named “altID”, it will process the request. If the request does not contain this parameter, it will ignore the request.

When “altID” is present, the QueryStringModule will make a database call, using functions contained in an accompanying assembly called “Woolpert.BusinessObjects.Accela.dll”, which was also built for this effort. It will search for the first Accela B1PERMIT row that matches the incoming “altID” parameter. If such a row is found, it will read that row’s “B1\_PER\_ID1”, “B1\_PER\_ID2”, “B1\_PER\_ID3”, and “B1\_MODULE\_NAME” fields.

As these fields are read, they are added to the incoming request’s “QueryString” object as “capId1”, “capId2”, “capId3”, and “module”.

The result is that the eventual recipient of the request, in this case “urlRouting.ashx” is supplied with the expected parameters, and will behave as if they had been passed in along with the original request. So, even though the original request may have only had access to a supplied “altID”, the ultimate recipient of the request still receives all QueryString parameters that it’s looking for. This is a type of “URL Rewriting”.

# Dependencies

The QueryString module makes a minimal few assumptions about the environment in which it runs. These are shown below.

* Target Platform: Any
* Target Framework: .NET Framework 4
* Entity Framework version 5.0 (see accompanying file “EntityFramework.dll”)
* Log4Net version 1.2.10.0 (see accompanying file “log4net.dll”)

# Deployment

Deployment of QueryStringModule is non-invasive and reversible, provided precautionary measures are taken before deployment begins.

Before deployment begins, a backup of the target ACS site’s “web.config” file should be made and stored in a safe place.

To deploy QueryStringModule, perform the steps listed below.

## Deploy Assemblies

Copy the following assemblies to the target ACA site’s bin directory:

* EntityFramework.dll
* Woolpert.ACAModules.dll
* Woolpert.BusinessObjects.Accela.dll

## Update Web.config

After deploying the assemblies to the bin folder make the changes to Web.config listed below.

### Add the “entityFramework” config section registration.

The following <section/> element should be added to Web.config’s <configSections/> parent element:

<section name="entityFramework" type="System.Data.Entity.Internal.ConfigFile.EntityFrameworkSection, EntityFramework, Version=4.4.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false" />

### Add the “entityFramework” config section definition

The following “<entityFramework/>” element should be added to the root “<configuration/>” element:

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.SqlConnectionFactory, EntityFramework" />

</entityFramework>

### Add the app settings

The following <add/> elements should be added to the root <appSettings/> element:

## <add key="QueryStringModule:TargetUrl" value="urlRouting.ashx"/>

## <add key="QueryStringModule:AltIdParameterName" value="altId"/>

## <add key="QueryStringModule:Id1ParameterName" value="capId1"/>

## <add key="QueryStringModule:Id2ParameterName" value="capId2"/>

## <add key="QueryStringModule:Id3ParameterName" value="capId3"/>

## <add key="QueryStringModule:ModuleParameterName" value="Module"/>

### Add the connection string

The <connectionStrings/> parent element should exist as a child element of the root <config/> element, and should contain the single <add/> element as shown below:

<connectionStrings>

<add name="AccelaEntities" connectionString="metadata=res://\*/AccelaModel.csdl|res://\*/AccelaModel.ssdl|res://\*/AccelaModel.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=AADBMAINT;initial catalog=Accela;persist security info=True;user id=aareporting;password=password;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

</connectionStrings>

### Update the connection string

The connection string shown above is based on Torrance’s test ACA site. It will need to be modified when targeting a different environment. To target a different environment, modify the following parameters:

* data source: update this to your target server name.
* initial catalog: update this to your target database name.
* user id: update this to the name of the SQL Server service account you’ll be using to read the B1PERMIT table. This account should have limited access to the Accela database, and only needs to have SELECT permissions on the B1PERMIT table.
* password: update this to the password you’ll be using with your SQL Server service account.

### Register QueryStringModule in the HTTP module pipeline

The QueryStringModule is added and removed to the HTTP module pipeline through configuration. The sample below shows it being registered as the first HTTP module in the pipeline. Order shouldn’t be particularly important, but because the exact function of the modules already registered was unclear, QueryStringModule was registered as the first module in the pipeline.

## <httpModules>

## <add name="QueryStringModule" type="Woolpert.ACAModules.QueryStringModule"/>

## <add name="CustomPageModule" type="Accela.ACA.Web.Common.CustomizeUrlRoutingModule"/>

## <add name="RedirectModule" type="Accela.ACA.Web.Common.RedirectModule"/>

## <add name="ActionFilterModule" type="Accela.ACA.Web.Common.ActionFilterModule"/>

</httpModules>

## Recycle the ACA Site’s Application Pool

You will need to recycle the ACA site’s application pool after QueryStringModule has been deployed.

# Troubleshooting

To troubleshoot QueryStringModule, it is recommended that you try the following:

1. Lower the Log4Net logging level to “DEBUG”

QueryStringModule makes use of Log4Net logging. In addition to logging at the “ERROR” level when an error is encountered, it logs at the “DEBUG” level, and when this is enabled, will log the received QueryString object before and after it is transformed.

2. Validate the Connection String

Assuming that logging is enabled and you have set the logging level to “DEBUG”, you should see errors in the error log if your connection string is incorrect. The usual suspects should be looked at, i.e. server name, database name, login, and password, to make sure your connection string matches the server you are targeting.

3. Validate your Web.config.

As described in the section above, updating Web.config is a multi-step process, and omissions are unfortunately sometimes hard to spot. Carefully check your Web.config to make sure all entries have been made. I.e. are all application settings there? Is your connection string there? Are your entity framework registration and definition elements there? Did you register the QueryStringModule HTTP module?

4. Check the System Event Viewer

If your log4net logs aren’t showing anything amiss, the last place to check would be the System event viewer. Check both the System and Application logs for error messages that might point to the cause of any issues you’re encountering.

# Summary

This document has sought to provide a concise description of the QueryStringModule HTTP module, whose function it is to rewrite Query String parameters to allow systems that do not have access to Accela three-part key information, or module information. Basic operation, dependencies, deployment, and troubleshooting were briefly described.