Padarn Developers Guide

Using OpenNETCF's Padarn Web Server for Windows CE

Document Revision 1.9

April 3, 2009

Covers Padarn version 1.2.0

http://www.opennetcf.com

Table of Contents

Document History

Revision Number	Date	Notes
1.0	October 30, 2007	Initial Release
1.1	December 13, 2007	Added “Detecting Client Browser Information” section and BrowserDefinitions element under the configuration file section.
1.2	March 20, 2008	Added new configuration settings for Padarn 1.1 Added “Adjusting the Debugging Environment” Added Feature Change List
1.3	April 28, 2008	Added “Deploying the Padarn Web Server” section
1.4	May 28, 2008	Added information on configuring Basic and Digest authentication
1.5	July 28, 2008	Added sections describing configuration of Virtual Directories
1.6	December 18, 2008	Added LocalIP configuration Added Cookies configuration
1.7	January 6, 2009	Added TempRoot configuration
1.8	March 18, 2009	Added section on custom log providers
1.9	April 3, 2009	Added section on Caching and description of caching configuration section.

Introduction

Developing and debugging Padarn web sites typically requires a solution with at least three separate projects:

1. An ASPX page project
2. A code-behind project
3. A Padarn bootstrap project

In this guide we will walk through the purpose of each of these projects as we illustrate how to create and debug a basic web site with Microsoft Visual Studio 2005, explain how to configure Padarn and discuss the features Padarn currently supports.

Building and Debugging a Padarn Site

The Padarn SDK makes building and debugging ASP.NET web sites a simple process. In this section we will outline the steps for creating a full Solution for a Padarn web site.

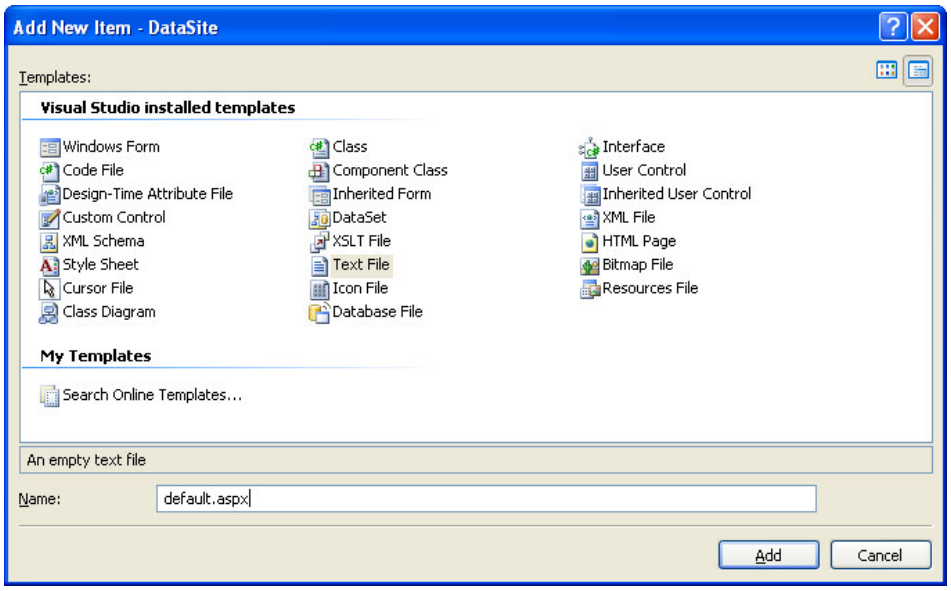
Project 1: ASPX Page Code

The first project you will need will be the project that holds the ASP.NET web page or pages (ASPX files) for your site. The reason we create this project first is simply because the naming of the solution and the namespaces is often related to the name of the site itself.

The following steps will guide you through the generation of this project.

1. Create a Smart Device Class Library project (Compact Framework 2.0).
2. Delete the default Class1.cs file that the New Project Wizard creates.
3. Add a new folder to the project called “Inetpub”
4. Add a new web page in the Inetpub folder. To create a page use the Add New Item dialog and select a “Text File” template then create the file with an “.aspx” extension. Visual Studio will

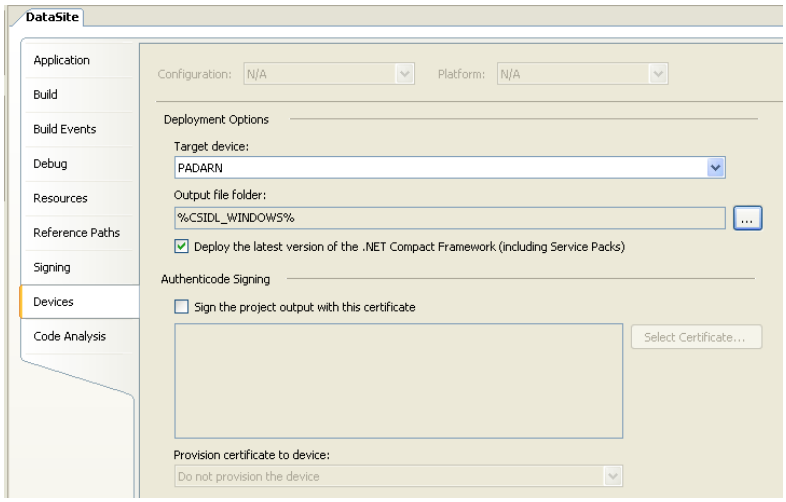
automatically recognize this as a web page file.



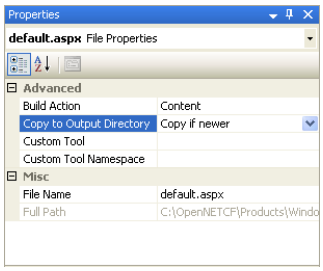
5. Add code to the ASPX page. Padarn currently supports only code-behind, so the actual code in the ASPX file is very simple. It contains the code-behind assembly name to load and the class name within that assembly that this specific page you are creating will instantiate.
   The name of the assembly and class will come from Project 2 and can be edited later. Below is an example of the entire code listing of a Padarn-supported ASP.NET page:

```
<%@ Page CodeBehind="DataSite.dll" Inherits="DataSite.Default" %>
```

6. Add any additional subfolders that your web site might use beneath the Inetpub folder. For example folders such as ‘images’ or ‘css’ are common collection points. These folders will be used to hold page resources for your web site.
7. Change the Device Output File Folder for the project to the \Windows folder (or whatever folder on the device that contains the Padarn Inetpub folder). This is done through the Project Properties dialog on the Devices tab, as seen in the figure below.



8. Adjust the 'Copy To Output Directory' property for all files to ensure that Visual Studio will push them to the target device when you Deploy.



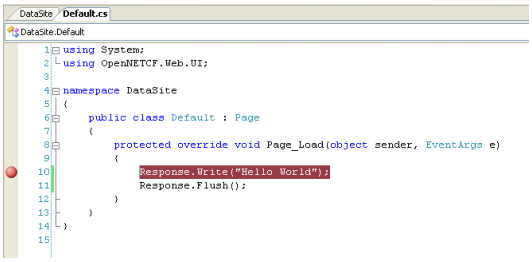
9. Deploy the project and verify that Visual Studio pushes the project files into the correct device location.

Project 2 : Code-Behind Logic

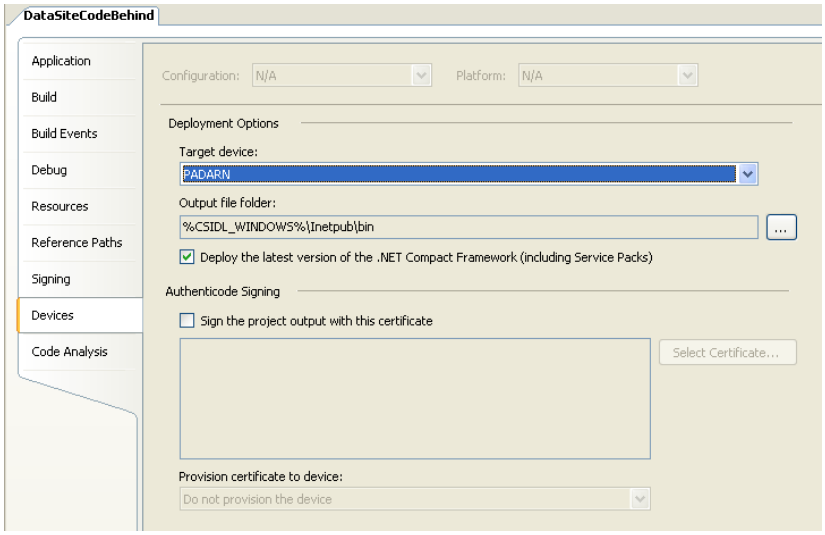
[explain/intro]

1. Add another Smart Device Class Library project to your Solution. Make sure that the naming of this project – or at least the output assembly name – matches the CodeBehind member used in your ASPX file (Step 5 in Project 1)
2. For the new project, add a reference to the OpenNETCF.Web.dll assembly
3. Rename the default class to match the "Inherits" member of your ASPX file (Step 5 in Project 1)
4. Modify the class code so that the class derives from OpenNETCF.Web.UI.Page
5. Override the Page_Load method and add a couple initial lines of code. This method will be the sole entry point for all logic of your Padarn-driven web page.

6. Set a break point in the first line of code in PageLoad. Your class code page will now look something like the figure below.



7. Change the output folder for the code-behind project to point to the Inetpub\bin folder on the device. Note that this is one folder below the Padarn server root folder and that the name of this folder must be 'bin'.



8. Deploy the project and verify that Visual Studio pushes the project files into the correct device location.

Project 3 : The Padarn Hosting Application

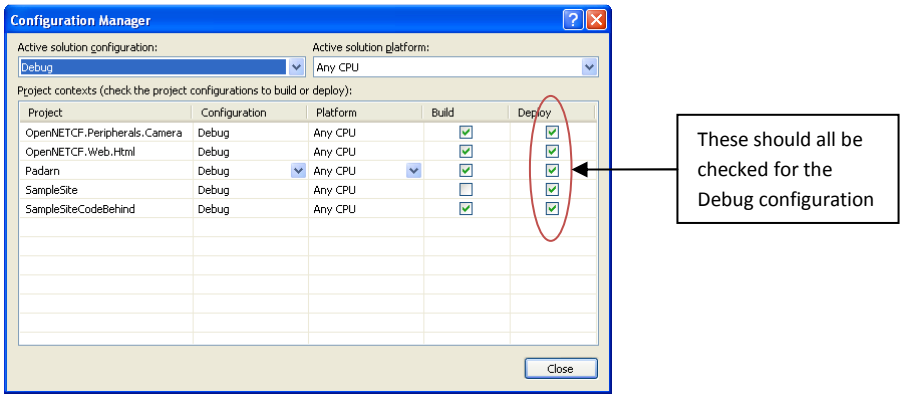
Padarn is a class library intended to be hosted in an executable (see the Padarn Server Hosting section for more information). Both the Padarn SDK and Padarn SampleSite ship with a sample application for hosting the server called Padarn.csproj. This project enables you to run your solution from Visual Studio and use live debugging for your pages.

1. Add the OpenNETCF-supplied Padarn project to your Solution.
2. Set the Padarn project as the Startup project for the Solution.
3. Begin debugging. Visual Studio will deploy the Padarn executable.
4. Using any browser, browse to your ASPX page on the device. Visual Studio will stop at the breakpoint at the top of Page_Load in your code-behind assembly project.

Adjusting the Debugging Environment

Once you have all of your projects created, you need to make some modifications to the solution configuration to make debugging a little easier. Since your startup executable project (the Padarn project in the SampleSite solution) doesn't directly reference the ASPX page project (SampleSite) or the code-behind assembly project (SampleSiteCodeBehind), Visual Studio will not automatically push these files down when debugging. To get it to push them down without having to manually deploy them, you should adjust the deployment behavior of solution.

Open the Configuration Manager in Visual Studio (pull down the toolbar ComboBox where you can choose Debug or Release and at the bottom is an item for Configuration Manager) and you will see a Window like this:



Select the "Debug" configuration from the "Active solution configuration" dropdown, and then ensure that all of the checkboxes in the "Deploy" column are checked.

Notes on Debugging

Below are some notes that might aid in getting started with creating and debugging web pages and sites to run under Huron.

- If the debugger does not stop on your code-behind break points (typically the break point will not be solid red, but "hollow") it is likely because the assembly on the device does not match the assembly on the development machine. This is usually due to a failure to deploy the code-behind assembly after making changes. Either manually deploy the project or verify that the current configuration is set to automatically deploy all of your projects (see the section entitled "Adjusting the Debugging Environment").

Padarn Server Hosting

The Padarn server is encapsulated in the OpenNETCF.Web.dll assembly which provides the ability to host Padarn in any Compact Framework 2.0 or later managed executable. The Padarn SDK ships with a sample source project called Padarn.csproj that will generate a runnable server executable that can be used as-is or as a template for creating your own hosting application.

Padarn Secure Socket Layer (SSL) Support

Beginning with version 1.1.0, Padarn supports serving pages over a secure socket. A single instance of Padarn can serve either secure or insecure pages – it cannot serve both. For details on configuring Padarn to use SSL, see the section entitled "Padarn Web Server Configuration".

Padarn Authentication Support

Beginning with version 1.1.1, Padarn supports both Basic and Digest authentication modes. Authentication is for the entire server (across all virtual directories) on the device. For details on configuring Padarn to use Authentication, see the section entitled "Padarn Web Server Configuration".

Padarn Virtual Directory Support

Beginning with version 1.1.50, Padarn supports virtual directories, which can be set individually to optionally use authentication as well. For details on configuring Padarn to use Virtual Directories, see the section entitled "Padarn Web Server Configuration".

Padarn Cookie Support

Beginning with version 1.1.75, Padarn supports storing Cookies on the client browser. Padarn supports HTTP-only and SSL-requiring cookies. For details on configuring Padarn's Cookie usage, see the section entitled "Padarn Web Server Configuration".

Padarn Custom Logging Support

Beginning with version 1.1.91, Padarn supports plugging in customized log providers. Padarn has a built-in log provider that logs information about pages served as well as error conditions. If the Padarn's default behavior does not meet your specific needs, you can create an assembly that contains a public class that implements the OpenNETCF.Web.Logging.ILogProvider interface and Padarn will call your log assembly instead of using the default provider.

For details on configuring Padarn to use a custom log provider, see the section entitled "Padarn Web Server Configuration".

Padarn Web Service Support

Web services come in many varieties, but generally fall into one of three categories: RPC, SOA or REST. Beginning with version 1.1.75, Padarn supports creating RPC-based web services, though it does not currently support WSDL generation or service introspection.

This means that you can create a Web Service (see the ExampleService page in the SampleSite code) but it is not the familiar SOAP-based XML Web Services typical of larger-scale IIS servers. Since Padarn is designed to be a lightweight service running on a resource-constrained embedded device, we continue

to evaluate the various Web Service options in the market are weigh them against the amount of processing resources we believe that will be required of the device to implement them.

Padarn Caching Support

Since Padarn is designed to run on devices with limited resources it is important to try to minimize any unnecessary work for the server to get the best possible client experience. One way to get significant improvements in many scenarios is to prevent Padarn from serving up static content like images and style sheets every time a page is requested, but instead let the client browser cache the static content for later use. As of version 1.2.0, Padarn supports two overall mechanisms for adding caching to your solution.

Page-settable Caching

The code-behind assembly for any give ASPX page can affect the caching for the current page by using the Cache property of the HttpResponse class (available in the code-behind as the Response object).

Global Caching Profiles

Padarn can be configured to globally set the caching behavior for files with a specific extension by setting a global caching policy. For example Padarn can be configured to have all client browsers cache all files with the extension ".css" for a specific period of time. For details on configuring Padarn's global caching profiles, see the section entitled "Padarn Web Server Configuration".

Serving non-ASPX Pages and Files with Padarn

The Padarn server can serve basic HTML pages as well as ASPX pages with code-behind. Simply add the pages into the DocumentRoot folder or one of its subfolders as you would with any other web server. For ease of debugging and deployment, it works best to add these files to the same project that your ASPX page is in (Project 1 in the sample outlines above).

ASP.NET Controls and Events

The current version of Padarn does not support hosting ASP.NET controls in a web page. The only supported event in Page_Load.

If your solution requires specific events or controls, please contact us at padarn@opennetcf.com. We are always looking at improving Padarn. Knowing which features and controls are key to your success helps us to use resources to improve the most important parts of Padarn.

Inline code and Uncompiled Code-Behind Files

Since the .NET Compact Framework does not support Compiler Services, Padarn does not support inline code inside an ASPX file or uncompiled code-behind files. All code-behind must be in a managed class library build for the Compact Framework version 2.0 or earlier.

Padarn Web Server Configuration

The Padarn web server behavior can be adjusted through an application configuration file. The configuration file that is provided with the Padarn SDK is `ocfhttpd.exe.config`.

Below is a sample configuration file and an explanation of its contents (line numbers are for reference only and are not to be included in the actual file):

```
<?xml version="1.0" encoding="utf-8" ?>
 <configuration>
  <configSections>
   <section name="WebServer"
      type="OpenNETCF.Web.Configuration.ServerConfigurationHandler,
        OpenNETCF.Web" />
   <section name ="httpRuntime"
      type ="OpenNETCF.Web.Configuration.HttpRuntimeConfigurationHandler,
        OpenNETCF.Web"/>
  </configSections>

  <WebServer
    LocalIP="0.0.0.0"
    DefaultPort="80"
    MaxConnections="20"
    DocumentRoot="\Windows\Inetpub\"
    TempRoot="\Storage Card\ASP.NET Temp Files"
    Logging="true"
    LogFolder="\Temp\Logs"
    LogExtensions="aspx;html;htm;zip"
    LogProvider=”\Windows\MyLogger.dll”
    BrowserDefinitions="\Windows\Inetpub\config\browsers"
    UseSsl ="true"
    CertificateName = "\Windows\certificate\server.pfx"
    CertificatePassword = "padarn"
   >

   <DefaultDocuments>
     <Document>default.aspx</Document>
     <Document>default.html</Document>
   </DefaultDocuments>

   <Authentication Mode="Digest" Enabled="false" Realm="Padarn Test Site">
     <Users>
       <User Name="UserA" Password="g0ba1th" />
       <User Name="UserB" Password="123abc456" />
     </Users>
   </Authentication>

   <VirtualDirectories>
     <Directory
       VirtualPath="admin"
       PhysicalPath="\Windows\WebAdmin\"
       RequireAuthentication="true"
     />
   </VirtualDirectories>

   <Cookies
     Domain="169.0.0.2"
     RequireSSL="false"
     HttpOnlyCookies="false"
   />

   <Caching>
     <Profiles>
       <add extension=".css" location="Client" duration="00:00:45" />
     </Profiles>
   </Caching>

  </WebServer>

```

```
62  <httpRuntime
63     maxRequestLength="4096"
64     requestLengthDiskThreshold="256"
65   />
66  </configuration>
```

Line 1: The configuration file is XML and must contain an XML document header

Lines 2 and 59: The configuration must be wrapped in a `configuration` node.

Lines 3-10: This is the section name and Type of the ConfigurationHandler that Padarn uses for parsing the configuration. These should not be modified. If you are hosting Padarn in your own executable and you want to add your own configuration settings and handlers, simply add new `section` nodes between lines 9 and 10 and then the section data outside of the `WebServer` or `httpRuntime` nodes.

Lines 12-53: These are the configuration variables for the Padarn Web Server. All supported variables are listed in this sample and the explanation for each is as follows:

LocalIP:	The local IP address to which Padarn will bind. Use an address of 0.0.0.0 to bind to all available addresses on the local machine or a specific IP adddress to ensure Padarn only services requests on a specific network interface.
DefaultPort:	The port on which Padarn will listen for HTTP page requests. This value is required and must be a positive numeric value. Typically you will use a value of 80 for standard, non-secure web pages or a value of 443 for a SSL protected site.
MaxConnections:	The maximum number of concurrent connections that Padarn will serve. Adjust this variable based on your usage and hardware profile. This value is required and must be a positive numeric value.
DocumentRoot:	The folder path on the Padarn Server device from which pages are served. This is the fully-qualified path to the server root and your static page files must reside in this folder or one of its subfolders. This value is required and must be a valid string folder path that already exists on the device.
TempRoot:	The folder path on the Padarn Server device where temporary files are stored (e.g. during file uploads). This value is optional and omitting it will default to '\Windows\Temp\ASP.NET Temp Files". If the folder does not exist it will be created.
Logging:	Determines whether or not Padarn will log files served. This value is optional and omitting it will default logging to *false*. Logs are saved into text files, with one file being generated per day, and log entries being appended

to the bottom of the log file. If provided, the value must be either *true* or *false*.

LogFolder: The output folder for all log files. This folder is an absolute path on the device and is not tied to the *DocumentRoot* variable above, so logs can be stored anywhere on the device. This variable is optional. If omitted and *Logging* is set to *true*, then it will default to *Temp**PadarnLogs*. If the folder name provided does not exist, Padarn will attempt to create it.

LogExtensions: A semicolon-delimited list of file extensions to log. This value is optional. If omitted, Padarn will log every file served, and if pages uses image files and style sheets, the logs can grow very large very rapidly with data that may not be of much value.
By providing an explicit list of file extensions to log, you can help control the size of your log files. The extensions listed must be semicolon delimited and must not contain spaces. Extensions are not case sensitive and may or may not include the leading period (so '.html' and 'HTML' yield the same result). Log filtering by anything other than file extension is not supported.

LogProvider: The absolute path to a file that contains a custom ILogProvider implementation. This value is optional. If omitted, logging is provided by Padarn's default log handler, which uses the other logging-related configuration values above.

BrowserDefinitions: The folder to search for browser definition files. This folder is an absolute path on the device and is not tied to the *DocumentRoot* variable above, so logs can be stored anywhere on the device. This value is optional and if omitted, no client browser information will be determined by the Page.Request.Browser property. For more information, see the section entitled "Detecting Client Browser Information."

UseSsl: Determines whether or not Padarn will use Secure Sockets (SSL) for communication with connected client browsers. This value is optional and omitting it will default logging to *false*. If provided, the value must be either *true* or *false*.

CertificateName: The fully qualified path to the server's SSL certificate file. This is an absolute path on the device and is not tied to the *DocumentRoot* variable above, so the certificate can be stored anywhere on the device. This value is required if *UseSsl* is set to *true*.

CertificatePassword: The password for the server's SSL certificate file. This value is required if *UseSsl* is set to *true*.

DefaultDocuments: This section lists the default document names that Padarn will look for when a directory URL is provided without any specific target document (i.e. http://www.Padarn.net/). Padarn will search the specified directory for any one of the default documents, starting with the first, and display the first one it finds. DefaultDocuments are set server-wide, meaning that they apply to all physical and virtual directories on the server.

Authentication: This section describes the Authentication mode settings for the Padarn Web Server. The Authentication node requires the following attributes:

Mode: This can be set to *Digest* or *Basic*
Enabled: This can be set to *true* or *false*.
Realm: This is the Realm name for the authenticated session

Inside the *Authentication* node is a list of usernames and passwords that are used for authentication if the *Enabled* attribute is set to *true*.

VirtualDirectories: This optional section provides a list of Virtual Directories that the Padarn Web Server will use. This section contains a list of *Directory* nodes that describe the properties of each desired virtual directory as follows:

VirtualPath: This is the virtual path under the server root for the virtual. For example a value of "admin" maps to a directory URL of http://www.padarn.net/admin.

PhysicalPath: This is the fully-qualified file system path to the physical directory on the device containing the virtual directory's files.

RequiresAuthentication: Sets whether or not authentication is enabled for the virtual directory. If *true*, the authentication mode is determined by the mode set in the *Authentication* configuration section.

Cookies: This optional section provides configuration attributes for how the Padarn Web Server will handle Cookies. This section contains one or more of the following attributes:

Domain:	This is the domain to associate with all Cookies stored by the Padarn Web Server on client browsers. This value is required if the Cookies section exists.
RequireSSL:	Indicates whether or not Cookies require the use of Secure Sockets (SSL). This value is optional and omitting it will default to *false*. If provided, the value must be either *true* or *false*.
HttpOnlyCookies:	Indicates whether or not the support for the browser's HttpOnly cookie is enabled. This value is optional and omitting it will default to *false*. If provided, the value must be either *true* or *false*.

Caching: This optional section provides configuration attributes Padarns global caching profiles. If this section exists, it must contain a node named *Profiles* with at least one Profile. A Profile must be defined with the node name *add* and use the following attributes:

extension:	The extension of the files to which this cache policy will be applied. This attribute is required.
location:	The location where the caching should take place. This can be set to *Client*, *Downstream* or *None*. This attribute is optional and if omitted will default to C*lient*.
duration:	A time span specifying the amount of time the page or resource is cached. This value must be in the format *hh:mm:ss*. This attribute is optional and if omitted will default to 00:00:30.

Padarn Web Server Deploying the Padarn Web Server

In this section we will outline the minimum software requirements to successfully deploy and run a site hosted by a Padarn Web Server. Keep in mind that your specific page content and implementation of the hosting application could expend the list of requirements, but the items below will provide a minimum working set for you to start with.

Windows CE Image Requirements

The Padarn Web Server requires an underlying operating system of Windows CE 5.0 or higher. Windows CE versions 4.2 and earlier are unsupported.

Below is a list of SYSGEN variables that need to be set during the build of your device's Windows CE image for the Padarn Web Server to be functional. If Padarn does not run on your device, contact your device vendor to ensure that your device's CE image meets these requirements.

> *Note: The SYSGEN lists below is a preliminary known-good set. Actual requirements may be a smaller list, but we've not yet got it refined. Keep in mind that these SYSGENS may pull in or require additional SYSGENS not in the list.*

```
SYSGEN_DOTNETV2
SYSGEN_DOTNETV2_SUPPORT
SYSGEN_ETHERNET
SYSGEN_FATFS
SYSGEN_MSXML_DOM
SYSGEN_SHELL
SYSGEN_STDIO
SYSGEN_TCPIP
SYSGEN_WININET
SYSGEN_WINSOCK
```

If you intend to use Padarn's SSL capabilities, the following SYSGENs must have also been set during the build of your Windows CE image.

```
SYSGEN_AUTH
SYSGEN_AUTH_SCHANNEL
SYSGEN_CERTS
SYSGEN_CERTS_PFX
SYSGEN_CREDMAN
SYSGEN_CRYPTMSG
SYSGEN_CRYPTO
```

Padarn Server Requirements

The list of CE image requirements for running Padarn are outlined in the section titled "Windows CE Image Requirements." This section describes the specific layout of files required for Padarn and how they are related and interdependent, allowing you to deploy a Padarn site in a custom location for your specific application.

.NET Compact Framework

The Padarn Web Server itself is a hosted assembly written in managed code (specifically C#). It is build against version 2.0 of the Microsoft .NET Compact Framework, which means that the device must have the version 2.0 or later of the Compact Framework installed.

This also has implications for your custom assemblies. Since Padarn is hosted inside another assembly, the host executable (ocfhttpd.exe for the shipped SampleSite solution) must be built against CF 2.0 **or later**. It also means that all of the code-behind assemblies that Padarn serves up (and any assemblies they reference) must be built against CF 2.0 **or earlier** (a 2.0 assembly cannot load a 3.5 or later assembly). For simplicity and continuity, OpenNETCF recommends that both the hosted assembly and the pages be built against version 2.0 of the Compact Framework.

File Dependencies and Layout

The Padarn Web Server is extremely flexible in how and where web site files may be deployed. Most of the behavior of Padarn is adjustable by modifying the configuration file deployed with the hosting application. This section outlines a typical installation and how to configure your target server device.

The Padarn Hosting assembly and Configuration File

The executable that hosts the Padarn engine can be stored and run from any location on the device. The SampleSite CAB file installs it in the \Windows folder, but this is simply because all devices are known to have a \Windows folder. For fielded applications where the engine is not in ROM, it may be useful to put the host executable into either a local persistent storage location (such as an IPSM or FlashFX Disk folder) or in mountable storage such as a hard disk or an insertable Compact Flash or USB storage device.

What is key is that the Padarn Web Server configuration file **must** reside in the same folder as the hosting application. As an example, the SampleSite project uses ocfhttpd.exe as the hosting assembly. The configuration file **must** be named ocfhttpd.exe.config and **must** reside in the same folder as ocfhttpd.exe, but the pair can be placed anywhere within the device's file system.

The Document Root, Page Content and code behind Files

Just like in other common web servers, the content for a Padarn web site is rooted to one server folder. The location of this root folder is called the DocumentRoot and can be located anywhere in the Padarn device's file system. It does not have any relation to where the hosting application resides. Your site's base address points to this root folder and all content must reside in either this root folder or one of its subfolders.

This can be illustrated with the following example. Assume your Padarn device IP address is 192.168.10.1 and your configuration file sets the DocumentRoot to "\Hard Disk". A browser navigating to http://192.168.10.1/default.aspx will load the file located at `\Hard Disk\default.aspx`. A browser navigating to http://192.168.10.1/Application/default.aspx will load the file located at `\Hard Disk\Application\default.aspx`.

As with Padarn's desktop counterpart IIS, code-behind assemblies must reside in a folder named "bin" that is a direct subfolder of the calling ASPX page file. Assuming the same Padarn Server setup as the previous example, the code-behind assembly for the page found at http://192.168.10.1/default.aspx must reside in the folder `\Hard Disk\bin` and the code-behind assembly for the page found at http://192.168.10.1/Application/default.aspx must reside in the folder `\Hard Disk\Application\bin`.

Logs, Certificates and Browser Definitions

For improved security, the following Padarn-related files can be placed outside of the DocumentRoot folder, making them non-browsable and non-available to Web clients: Padarn log files, certificates files used for SSL and browser definition files. Note, however, that none of these files are content files – instead they are consumed by the Padarn Server engine itself.

The location of these files is set by fully-qualified path entries in the Padarn Server Configuration File.

Detecting Client Browser Information

Starting with Padarn build number 1.0.5010 the Page.Request object contains a Browser property. This property is analogous to the Page.Request.Browser property from the ASP.NET 2.0 desktop object model. For information on the usage of the object, see Microsoft's documentation of the property. The information returned in this property, however, is obtained in a slightly different manner than under IIS.

Like IIS, Padarn uses browser configuration files to enable you to affect and tune what capabilities different client browsers report. For simplicity and compatibility, the browser configuration files Padarn uses are direct copies from IIS (by default found on an IIS machine at `C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\CONFIG\Browsers`).

The first time a page accesses the Page.Request.Browser after Padarn starts, Padarn will compile, parse and cache all of the information stored in the browsers configuration files (see "Padarn Server Configuration" for the location of these files). This process can be time consuming, especially if there are a lot of browser files such as with the SampleSite code base.

Once any page accesses the property, Padarn will cache the results. This means subsequent page accesses of the Browser property are much faster. It also means that any updates to the browsers files require a restart of the Padarn server to take effect.

If you need to minimize this load time it is recommended that you reduce the number and size of the browsers files to match only the expected set of browsers that will be connecting to your server. For example, if you are in an internal installation where you know only Firefox and Internet Explorer will be used, remove all other browser files.

Note that Padarn parses the browser XML configuration files by using the "id" and "parentID" attributes of the browser nodes. There must always be a node with no parentID (the "default" browser in the case of SampleSite or the default IIS files). There must also be a valid chain of parent-child relationships.

For example, if you want to have support for Internet Explorer using the default browser files, you must keep Default, Mozilla and IE since that is the order of the relationships, though the contents of the files can be reduced depending on the versions you might want to support. Again, Padarn follows the same rules as IIS, so any documentation on how these files work for IIS also applies to Padarn.

Padarn Feature Change List

Below is a list of feature changes in Padarn and the versions in which they appeared. This list is provided solely for reference and may not cover all features. For a full list of what Padarn supports, reference the release documentation.

Version Number	Features Changes
1.0.5000	Initial Release
1.0.5010	Added • BrowserCapabilities • Logging
1.0.5020	Added • UserHostAddress
1.0.5030	Added • HttpResponse.BinaryWrite, Cache and Redirect
1.1.0	Added • Secure Sockets Layer (SSL) support • HTTP POST (HttpRequest.Form) support • HTTP File Upload (HttpPostedFile) support • Multiple default document support • Virtual Directory Support • HttpRuntime support • HttpUtility class
1.1.1	Added • Basic Authentication • Digest Authentication
1.1.50	Added • Virtual Directory Support
1.1.71	Added • LocalIP Server Configuration • Configuration Property on WebServer class • Ability to reload the server configuration without restarting the Padarn server
1.1.75	Added • Cookie support • RawQueryString Property on HttpRequest class
1.1.78	Added • Configurable Temp folder
1.1.91	Added • Improved error page outputs to client browser • Support for Custom log providers
1.2.0	Added • Added SetCacheability to HttpResponse • Added Global Cache Policy support • Improvements to the HostingEnvironment class