|  |  |
| --- | --- |
|  | **Unidev** |
|  | **Unified Development, Inc.**  16690 Swingley Ridge Road  Suite 260  Chesterfield, MO 63017  (636) 532-4424 |

|  |
| --- |
|  |
|  |

**Duncan Solutions, Inc**.

Web Services

March 2014

Revision 1.00

**Prepared by:**

Caleb Miller

Unified Development

**Table of Contents**

[1. Overview 4](#_Toc382218577)

[1.1 Goal 4](#_Toc382218578)

[2. Target Audience 5](#_Toc382218579)

[3. Field Maintenance Services 6](#_Toc382218580)

[3.1 External Services 6](#_Toc382218581)

[3.2 Internal Services 6](#_Toc382218582)

[3.2.1 Deployment 7](#_Toc382218583)

[4. Testing 10](#_Toc382218584)

**Revision** History

|  |  |  |  |
| --- | --- | --- | --- |
| Revision | Author | Description | Revision Date |
| 1.00 | Caleb Miller | Initial version | 03/10/2014 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Overview

## Goal

The purpose of this document is to explain the concepts and methodologies used with the web services in the PEMS system. There are two basic categories of services that we have to deal with: Internal and External. Internal services are any service that we are exposing to the outside world, and External services are any service that does not reside within the application and we must consume / interact with from a different source.

At the time of this writing, we are only using Restful web services, but the application is not limited to this and may use SOAP services in the future. The type of each service will be noted in its description. This document will be kept up to date as services are added.

The following items will be covered:

1. Field Maintenance Web Services
   1. Internal Web Services
      1. Deployment
   2. External Web Services
      1. Settings and location

# Target Audience

The target audience of this document is person or persons who have:

1. Experience in the following technologies:
   1. C# /.Net /
   2. Restful Web services
   3. T-SQL, SSMS, SqlServer, Stored Procedures and Views
2. The user of this document has a full understanding of the Duncan PEMS project. This includes database and system architecture knowledge, desired business rules of the application, etc.
3. Microsoft SQL Server administration and understand rights, database creation and administration, and are able to use either SSMS or SQL command line interface.
4. Visual Studio 2012

# Field Maintenance Services

The purpose of the web services that have been created for Field Maintenance is to keep all of the business logic for determining event UIDs, SLA Deadlines, etc. in one centralized location. There are 4 services that perform this action. Here is the spec provided by Duncan:

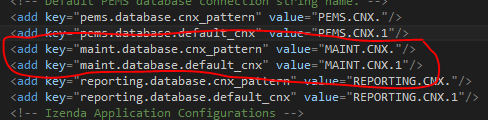


## Connection Strings

The following connection strings must be setup correctly in order for the web services to work correctly. These are located in the connectionStrings.Config file and web.config file

### Web Config

We need to add or verify the existing items reside within the <appSettings> node in the web config:



The maint.database.cnx\_pattern defines the pattern for the maintenance connection strings for multiple customers, and the maint.database.default\_cnx is what the connection string name for Field maintenance is defaulted to. Currently the work orders and field maintenance information resides within the RBAC database, so only one is needed, but the system is designed to accommodate customer specific databases for field maintenance work order information.

### Connection Strings

Now that we have defined connection string names, we must add / verify these named connecitons exist in the system. This file is located at the web root and is called connectionStrings.config. Here is the connection string for Maint.CNX.1.

<!--MAINTENANCE GROUP CONNECTION STRINGS-->

<!--These connection strings are based on the maint.database.cnx\_pattern app setting in the web config (MAINT.CNX. at the time of this writing) and will be the connection strings used for MAINTENANCE GROUPS.

These must point to the location of the maintenance group work order database, currently residing in the RBAC DB.-->

<add name="MAINT.CNX.1" connectionString="metadata=res://\*/RBAC.MaintenanceEntities.csdl|res://\*/RBAC.MaintenanceEntities.ssdl|res://\*/RBAC.MaintenanceEntities.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=$$DataSource$$;initial catalog=$$Database$$;user id=$$Username$$;password=$$Password$$;MultipleActiveResultSets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

The following replacement values will have to be updated for the specific environment:

1. $$DataSource$$ - the name of the DB server that the DB is hosted on.
2. $$Database$$ - The name of the database. Currently this is set as the name of the RBAC database.
3. $$Username$$ - username of the account the application uses to connect to the DB.
4. $$Password$$ - password of the account the application uses to connect to the DB.

## External Services

The PEMs application is using two external services for field maintenance. Both of these services are RESTful and the locations of the services are located in the appSettings node of the web config file. Currently they are pointing to the test service spun up during the development phase:

<!-- Field Maintenance Web service Urls -->

<add key="pems.webservices.createalarm" value="http://115.119.37.66:8080/ripnet/RClAl/createalarm.rmws"/>

<add key="pems.webservices.closealarm" value="http://115.119.37.66:8080/ripnet/RClAl/closealarm.rmws"/>

The Typical lifecycle for an External web service is:

1. Generate the request
   1. This builds the “Data” object we are sending to the web service that includes all of the items that we need to act upon
2. Serialize the request
   1. This serializes the request into something the web service can understand: first a string representation of the Data object, then a byte array of that object to use as the request body to pass off to the service.
3. Post the request
   1. This generates a HTTPWebRequest object for the service. This includes the byte array of the serialized Data object, the location of the web service (found in the web config), and posting the data to the correct url
4. Generate the response
   1. Once we get a response from the posted request, we have to de-serialize the request back into a Data object, parse it and determine the response the application needs to know about. Then the response is sent back to the application for processing.

All of the logic for both of these web services is located in the Duncan.Pems.Business ->WebServices-> WebServiceFactory.cs file.

## Internal Services

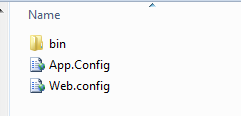
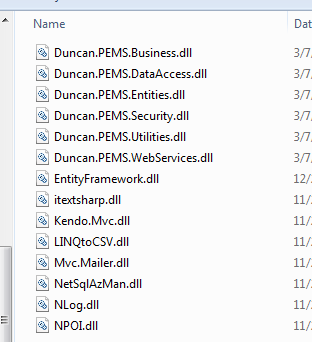
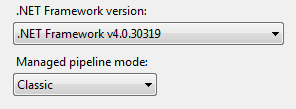
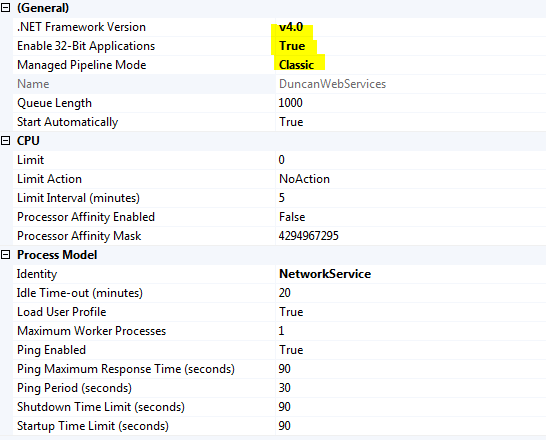
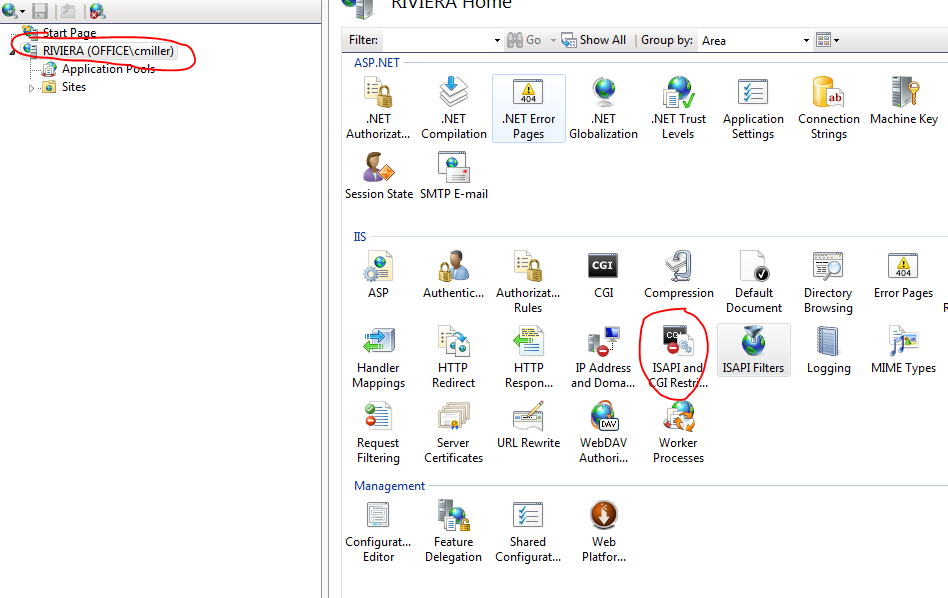
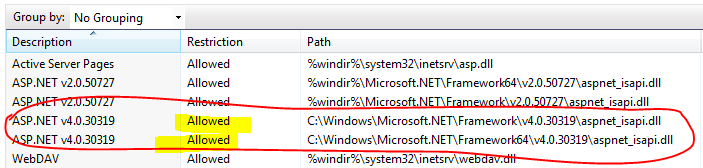
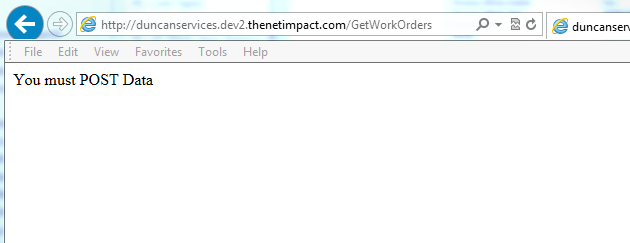
The PEMs application is using two internal services for field maintenance: Clear Alarms and Get Work orders. These are both setup as RESTful services. Duncan will be setting up these services internally and currently will only be accessed internally via code. All of the logic for the internal web services for field maintenance is located here: Duncan.PEMS.Webservices -> FieldMaintenance -> ClearAlarms.cs and GetWorkOrders.cs.

The typical lifecycle of an internal service is:

1. The request is received
   1. We check to make sure the data is done via “Post”
2. The request is deserialized into the “Data” object
3. The Data object is passed to the Web Service Factory that handles the request
   1. The request is parsed and checked for issues
   2. Then it is processed (either the alarms are cleared or the work order is created / retrieved) based on the business rules for the web service (as defined in the web services document referenced above)
4. Generate the Response
   1. A response is built based on the result of the request processing
   2. The serialized version of the response is written to the current HttpContext Response. This will allow the calling party to determine the response of their initial request.

### Deployment

In order to deploy the internal web services, the following steps must be taken (.net 4.0 has to be installed on the server as well):

1. **Publish the services**
   1. The Duncan.PEMS.WebServices project must be built and published to a folder.
   2. The root of the published folder should look like:
      1. 
   3. And the bin folder should include:
      1. 
2. **Update the Web Config**
   1. The web config is where the connection strings are located for the various databases. This should mimic the connectionstrings.Config connections from the main PEMS application. Go through the web config and make updates to ensure all of the items inderneath the <connectionStrings> node are correct.
   2. Do not modify the scriptProcessor attribute of the <system.webServer> / <handlers> / <add> nodes unless the .Net isapi handler is located somewhere else. This is the default location.
3. **Setup IIS**
   1. Application pool
      1. 
      2. 
   2. Create a new site that points to the published folder from Step 1
   3. We must allow the Isapi module for .net 4.0 on the IIS Server
   4. 
   5. We have to make sure the v4.0 are allowed
   6. 
4. Navigate to the services. For example, if your Sites url is http://InternalServices/, then you would navigate to <http://InternalServices/ClearAlarms> or <http://InternalServices/GetWorkOrders> and you should see a screen similar to this:
5. 

# Testing

There is a testing application that has been setup in order to facilitate testing each service call as if we were doing it from the outside. This is located at: Duncan.PEMS.WebServiceTest. It includes some sample files to test with (to mimic requests coming in) and a simple program that allows you to call each method programmatically.