# RBDMS Data Mining Configuration

*RBDMS Training 2010*


Coordinate solutions and Virtual Engineering Solutions, Inc.

www.VirtualES.com

# Table of Contents

# Introduction

RBDMS Data Mining provides users the ability to search, review, and visualize RBDMS data via a Web browser. RBDMS Data Mining is a specific implementation of several base RBDMS.NET projects: RBDMSBase, RBDMSWeb, and RBDMSWebGIS.

RBDMS Data Mining is a presentation layer written in ASP.NET 2.0. The underlying search, results, and GIS functionality are contained in other base RBDMS projects. RBDMS Data Mining is configured by means of several XML files at runtime.

- The Full Text Search and Result Configuration file controls full text search operation and results.
- The Details Configuration files control results for each entity type in Nodes.xml.
- The RBDMSWebGIS Configuration file controls RBDMSWebGIS operation and interaction with RBDMS Data Mining.

The remainder of this document discusses the file structure, contents of the configuration files, and other requirements for their use.

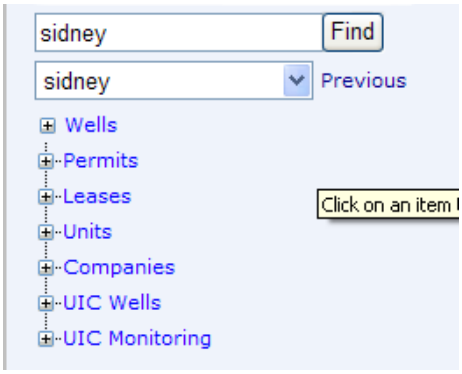
# Full Text Search and Result Configuration

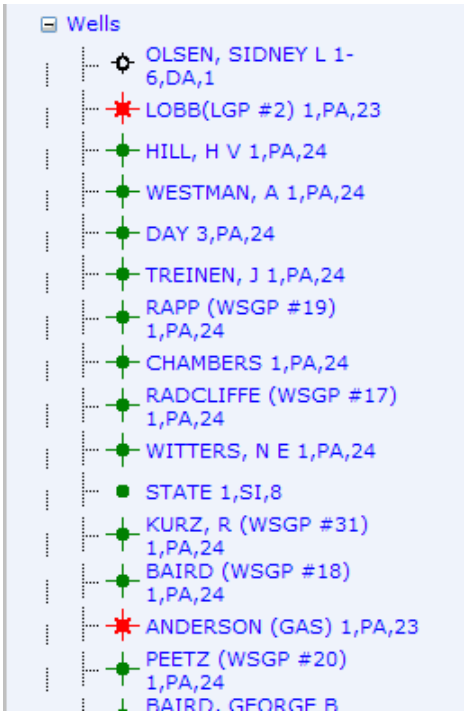
The full text search and results configuration file governs the behavior of the RBDMSWebControls.FullTextSearch web control. This control handles the preparation, execution and display of full text searches against multiple entities simultaneously.

The control allows a user to enter key words and receive results matching those criteria for multiple entity types (e.g., wells, permits, leases, companies, etc.).

The control can be configured either to expand or collapse tree nodes on click or to perform specific operations such as generating grid-style reports.

The results tree can be expanded to show individual results for an entity type. Individual results contain descriptive text and images (optionally, see FullTextThemeItem).



As other RBDMS configuration files, the full text search and results configuration file is a serialized object. The FullTextSearch Web control contains a FullTextNodesPersist object that has one top-level object: FullTextNodes, which is a collection of FullTextNode objects.

Each FullTextNode object represents a single entity and has the following properties:

- Name: The entity name, used as the text for the root node.
- NavURL: JavaScript action to perform on click of individual results
    - Contains a parameter %NavUrl% that will be replaced with the key column's value after a search is performed.
- NavRootURL: JavaScript action to perform on click of the root node.
- SQL: The base SQL for the full text search to be performed.
    - Contains a parameter %Search% that will be replaced with the full text search keywords generated from user input.
- KeyName: The column in the results used as primary key.
- KeyType. The data type of the key column.

- ThemeColumn. The column in the results used to determine FullTextThemeItem.
- ThemeItems. The collection of FullTextThemeItem items.

FullTextThemeItem is used to precede the individual results' text with an image based on a field value. It has the following properties:

- Value. The value this theme item will be used to represent.
    - Can be comma-delimited list of values if this image will represent more than one value.
- Image. The URL of the image to display for this value or values.

See Appendix I: Full Text Search Samples for more information.

## *Configuring SQL Server Full Text Indexing*

Clearly, configuring full text indexing in SQL Server is necessary before a full text search will do much good. The following discussion assumes the use of SQL Server Management Studio (for SQL Server 2005). However, the principles apply to SQL Server 2000 (and Enterprise Manager) as well.

### Create a Full Text Table for Each Entity

While it is certainly possible (and, in fact, easier) to full text index an existing table, you can control the results more finely if you generate a full text table for each entity.

```
CREATE TABLE [dbo].[FTWell](
	[PKey] [int] IDENTITY(1,1) NOT NULL,
	[ParentKey] [varchar](14) COLLATE SQL_Latin1_General_CP1_CI_AS
NOT NULL,
	[FT] [text] COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL,
 CONSTRAINT [PK_FTWell] PRIMARY KEY CLUSTERED
(
	[PKey] ASC
) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

### Populate the Full Text Table for Each Entity

We will concatenate every field we want to query for an entity into the FT field in the FT[Entity] table created in the previous step. For example, to populate the FTWell table created above, we could use the following SQL (easily converting it into a DTS step or scheduled job as desired):

```
delete ftwell
go
insert into ftwell (parentkey, ft)
select
api_wellno,
ft=
coalesce(api_wellno,'') + ' ' +
coalesce(well_nm,'') + ' ' +
coalesce(lease_nm,'') + ' ' +
coalesce(well_no,'') + ' ' +
coalesce(well_typ,'') + ' ' +
coalesce(wl_status,'') + ' ' +
coalesce(wl_permit,'') + ' ' +
coalesce(wellpermitoriginal,'') + ' ' +
coalesce((select name_ from tblrefcounty where cnty_apino=cnty),'') + '
' +
coalesce((select coname from tblrefcompany where cono=opno),'') + ' ' +
```

```
coalesce((select pool_nm from tblgeopool where
poolno=tblwellmaster.poolno),'') + ' ' +
coalesce((select name_ from tblgeofields where st_fldno=field_no),'')
from tblwellmaster
```

## Create and Schedule the Full Text Index

With SQL Server Management Studio (SQLMS) open and connected to your database:

1. Enable Full Text Indexing (if necessary)
    - Right-click the database and choose Properties
    - Select the Files page
        - Check Use full-text indexing.
        - Click OK.
2. Create the Full Text Catalog
    - Expand database -> Storage
    - Right-click Full Text Catalogs and choose New Full Text Catalog
        - Full-text catalog name: DataMining
        - Catalog location: D:\MSSQL\SQLDATA\MSSQL\FTDATA
        - Click OK
3. Create the Full Text Index
    - Right-click the DataMining full-text catalog and choose properties.
    - Select the Table/Views page.
        - Select dbo.FTWell in the all object column
        - Click the -> button to move it to the catalog column
        - Place a check next to the FT column in the eligible columns area.
        - Click OK.
4. Schedule Population
    - Right-click the DataMining full-text catalog and choose properties.
    - Select the Population Schedule page.
        - Name the schedule (DataMiningSchedule)
        - Schedule type: recurring
        - Frequency: Daily at midnight.

- Click OK.

## Create Full Text SQL Statements for Each Entity

Each entity will require a single full text query to populate the results tree. These queries are relatively straightforward: the select list will contain only the items you wish to display in the tree or use to choose an accompanying image. The only new concept is likely the CONTAINSTABLE clause (http://msdn2.microsoft.com/en-us/library/ms189760.aspx).

Using SQLMS, test the full text queries for each entity using the following as a starting point:

```
select w.api_wellno, w.well_nm, w.wl_status
from tblwellmaster w
      inner join ftwell ftw
            on w.api_wellno=ftw.parentkey
      inner join containstable(FTWell, FT, '"union*" AND "oil*" and
"forest*"') c
            on ftw.pkey=c.[key]
order by rank desc
```

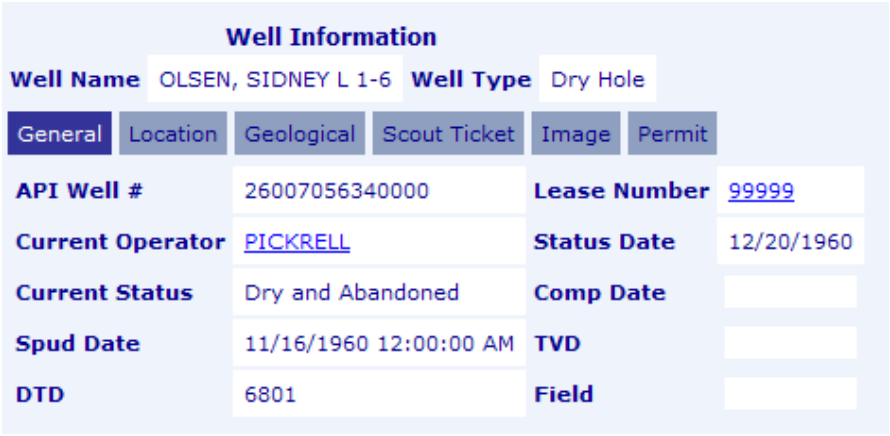
Replace the words “union”, “oil”, and “forest” with ones applicable to your database and entity and test; the results should be representative of those you would expect a user to receive after a search.

Finally, modify the SQL replacing the third parameter of the CONTAINSTABLE clause with ‘%Search%’ and place this into the appropriate SQL property of your FullTextNode object.

# Detail Configuration

Detail configuration files are used to configure the RBDMSWebControls.EntityDetails web control. The control facilitates standardized, tabbed display of results for individual entities (e.g., wells, permits, operators, etc.).

As other RBDMS configuration files, the details configuration file is a serialized object. The EntityDetails web control is populated with an EntityDetailsPersist object, which has two top-level objects:




- MainTab: Controls title (Well Information) and “always visible” information (Well Name and Well Type).
- Subtabs: The collection of EntityDetailsItem controls.

```
<?xml version="1.0" encoding="utf-8"?>
<EntityDetailsPersist xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MainTab>
  </MainTab>
  <Subtabs>
    <!-- GENERAL -->
    <EntityDetailsItem>
    </EntityDetailsItem>
  </Subtabs>
</EntityDetailsPersist>
```

Each tab (MainTab or EntityDetailsItem) has the following properties:

- Query: The object specifying the base sql for the tab and how the where clause will be constructed.
  - KeyNameAllowUpdate: Boolean (default = “false”). Can the calling application change the key name?
  - KeyName: The name of the key column.

    - KeyValue: The value in the key column we want to display. The KeyValue will always come from the calling application.
    - KeyType: The data type of the key column.
    - BaseQuery: The base SQL to which the constructed where clause will be appended.
    - OrderBy: The order by clause containing the text “order by.”
- RelatedTables: The collection of RelatedTable controls.
- ColumnsWide. The integer specifying how many fields per row are rendered. For example, if there are seven fields selected in the SQL statement and ColumnWide= “3,” then there will be 3 rows, the last containing only one field.
- TableTitle. The title of the details page if in MainTab or the tab title text if in EntityDetailsItem.

RelatedTable items are used to display lists of results (e.g., wells in field, leases for an operator, etc.). Each RelatedTable has the following properties:

- Query: See Query under MainTab / EntityDetailsItem).
- Title: The title string.

**See Appendix II: Details Samples for more information.**

## RBDMSWebGIS Configuration

The RBDMSWebGIS Configuration file is discussed in detail in the document *RBDMSWebGIS XML Configuration*.

# Appendix I: Full Text Search Samples

Wells FullTextNode (Complex):

```
    <FullTextNode>
      <Name>Wells</Name>
      <FilterName />
      <NavURL>javascript:window.parent.FillED('API_WELLNO','%NavUrl%','
        STRING','WellDetails.xml','ctl00_PageBody_WebPartManager1_gwpPan
        elDetails_DetailsFrame');</NavURL>
      <NavRootURL>javascript:window.parent.FillReport('Filter=NOGCC_Wel
        lList.xml&amp;Report=Well
        List.grid&amp;Key=Wells','ctl00_PageBody_WebPartManager1_gwpPane
        lDetails_DetailsFrame'); </NavRootURL>
      <Sql>
        <![CDATA[
        SELECT API_WellNo, Well_Nm, Wl_Status,
        GISSymbol=cast(
              Case
When (Well_typ='Dry Hole') Then 1
When (Well_typ='Expired Location' and wl_status in ('EX', NULL)) Then 3
When (Well_typ='Natural Gas Well' and wl_status not in ('PA')) Then 4
When (Well_typ='Geothermal') Then 5
When (Well_typ='Gas Injection Well') Then 6
--PLUGGED
When (wl_status in ('PA')) Then 23
--DEFAULT
Else 0
              End
           as varchar(2))
        FROM tblWellMaster WHERE CONTAINS (*,'%Search%')
        ]]>
    </Sql>
      <KeyName>API_WellNo</KeyName>
      <KeyType>String</KeyType>
      <SearchType>FullText</SearchType>
      <ThemeColumn>GISSymbol</ThemeColumn>
      <ThemeItems>
        <FullTextThemeItem>
          <Value>1</Value>
          <Image>~/images/wells/dh.png</Image>
        </FullTextThemeItem>
        <FullTextThemeItem>
          <Value>3</Value>
          <Image>~/images/wells/exp.png</Image>
        </FullTextThemeItem>
            ...
      </ThemeItems>
    </FullTextNode>
```

Permits FullTextNode (Basic):

```
<FullTextNode>
  <Name>Permits</Name>
  <FilterName />
  <NavURL>javascript:window.parent.FillED('API_WELLNO',%NavUrl%,'ST
   RING','PermitDetails.xml','ctl00_PageBody_WebPartManager1_gwpPan
   elDetails_DetailsFrame');</NavURL>
  <NavRootURL />
  <Sql>
    SELECT
    PKEY,API_WELLNo,  Well_Nm,CurrentStatus
    FROM tblWellPermits WHERE CONTAINS (*,'%Search%')
  </Sql>
  <KeyName>API_WELLNo</KeyName>
  <KeyType>String</KeyType>
  <SearchType>FullText</SearchType>
  <ThemeColumn>CurrentStatus</ThemeColumn>
  <ThemeItems>
    <FullTextThemeItem>
      <Value>*</Value>
      <Image />
    </FullTextThemeItem>
  </ThemeItems>
</FullTextNode>
```

# Appendix II: Details Samples

Company:

```
<?xml version="1.0" encoding="utf-8"?>
<EntityDetailsPersist xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MainTab>
    <Query>
      <KeyName>Cono</KeyName>
      <KeyValue xsi:type="xsd:string">01000</KeyValue>
      <KeyType>String</KeyType>
      <BaseQuery>
        <![CDATA[
        select
        [Company Name]=CoName
        from tblRefCompany
        ]]>
      </BaseQuery>
    </Query>
    <RelatedTables />
    <Tabs />
    <ColumnsWide>-1</ColumnsWide>
    <TableTitle>Company Information</TableTitle>
  </MainTab>
  <Subtabs>
    <!-- General -->
    <EntityDetailsItem>
      <Query>
        <KeyName>CoNo</KeyName>
        <KeyValue xsi:type="xsd:string">01000</KeyValue>
        <KeyType>String</KeyType>
        <BaseQuery>
          <![CDATA[
          select
          [Address]=Addr1,
          [Address2]=Addr2,
          [City]=City,
          [State]=State,
          [Zip]=PostalCode,
          [Phone]=Phone,
          [Phone Ext]=PH_Ext,
          [Fax]=fax
          from tblRefCompany
          ]]>
        </BaseQuery>
      </Query>
      <RelatedTables/>
      <Tabs />
      <ColumnsWide>2</ColumnsWide>
      <TableTitle>General</TableTitle>
    </EntityDetailsItem>
    <!-- Company Wells -->
```

```
    <EntityDetailsItem>
      <Query>
        <KeyName>Lease_Unit</KeyName>
        <KeyValue xsi:type="xsd:string">01000</KeyValue>
        <KeyType>String</KeyType>
        <BaseQuery />
      </Query>
      <RelatedTables>
        <RelatedTable>
          <Query>
            <KeyNameAllowUpdate>false</KeyNameAllowUpdate>
            <KeyName>OpNo</KeyName>
            <KeyValue xsi:type="xsd:string">01000</KeyValue>
            <KeyType>String</KeyType>
            <BaseQuery>
              <![CDATA[
              SELECT
              [API WELL NO.] = '<a
href="javascript:parent.FillED(''API_WELLNO'',''' + API_WELLNO +
''',''STRING'',''WellDetails.xml'',''ctl00_PageBody_WebPartManager1_gwp
PanelDetails_DetailsFrame'');">' + API_WellNo + '</a>',
              [WELL NAME] = Well_Nm
              from tblWellMaster
              ]]>
            </BaseQuery>
            <OrderBy>Well_Nm</OrderBy>
          </Query>
        </RelatedTable>
      </RelatedTables>
      <Tabs />
      <ColumnsWide>-1</ColumnsWide>
      <TableTitle>Company Wells</TableTitle>
    </EntityDetailsItem>
  </Subtabs>
</EntityDetailsPersist>
```

Wells (Basic):

```
<?xml version="1.0" encoding="utf-8"?>
<EntityDetailsPersist xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MainTab>
    <Query>
      <KeyName>API_WELLNO</KeyName>
      <KeyValue xsi:type="xsd:string">26001210010000</KeyValue>
      <KeyType>String</KeyType>
      <BaseQuery>
        <![CDATA[
        select
          [Well Name] = Well_Nm,
          [Well Type] = Well_Typ
        from tblWellMaster
        ]]>
      </BaseQuery>
    </Query>
    <RelatedTables />
```

```
    <Tabs />
    <ColumnsWide>-1</ColumnsWide>
    <TableTitle>Well Information</TableTitle>
  </MainTab>
  <Subtabs>
    <!-- GENERAL -->
    <EntityDetailsItem>
      <Query>
        <KeyName>API_WELLNO</KeyName>
        <KeyValue xsi:type="xsd:string">26001210010000</KeyValue>
        <KeyType>String</KeyType>
        <BaseQuery>
          <![CDATA[
           select
          [API Well #] = API_WellNo,
          [Lease Name] = Lease_Nm,
          [Current Operator] = '<a
href="javascript:parent.FillED(''CoNo'',' + Cast(OpNo as varchar(25)) +
',''STRING'',''CompanyDetails.xml'',''ctl00_PageBody_WebPartManager1_gw
pPanelDetails_DetailsFrame'');">' + (select CoName from tblRefCompany
where CoNo=OpNo)+ '</a>',
          [Status Date]= Convert(char(10),Dt_Status,101),
          [TVD]=TVD,
          [DTD]=DTD
          from tblWellMaster
          ]]>
        </BaseQuery>
      </Query>
      <RelatedTables />
      <Tabs />
      <ColumnsWide>2</ColumnsWide>
      <TableTitle>General</TableTitle>
    </EntityDetailsItem>
  </Subtabs>
</EntityDetailsPersist>
```

# Appendix III. Using a WCF for Some or All Data Access

In 2010, two projects necessitated adding the ability to consume WCF services from the Data Mining: Mississippi and Oklahoma. Mississippi's need arose from the desire to serve images to the Internet from their Laserfiche Imaging Service, which is only accessible from their internal network. Additionally, access to the Laserfiche web service from the RBDMS.Net WPF application was accomplished via a WCF service layer, and it made sense to reuse the service in Data Mining. Oklahoma's need arose from their IT requirement that there be no direct database access from any machine accessible from the Internet.

Though numerous modifications were made to the Data Mining to accommodate the use of WCF services, it is fully backward-compatible and uses the same configuration files for the Full Text Search and Details configurations.

The Data Mining application must have a service reference added to the RbdmsWebControls project. The service reference name is used in the configuration.

## *Full Text Search Example*

The only difference from the direct data access is in the SQL and SQLAdv elements. The formatting of these elements when using a web service is as follows:

- WebServiceCall|ServiceNamespace|ServiceReferenceName|MethodName|QueryName
- WebServiceCall: Literally says to the Data Mining application that you want it to get the data from a service (i.e., this is not SQL).
- ServiceNamespace: The namespace of the referenced service. This is entered when you add the service reference.
- ServiceReferenceName: Typically the ServiceNamespace & “Client”
- MethodName: The name of the service method to be called (as defined in the service contract).
- QueryName: The parameter to be passed to the service method.

```
<FullTextNode>
    <Name>Wells</Name>
    <FilterName />
```

```
<NavURL>javascript:window.parent.FillED('PKey',%NavUrl%,'Integer','WellDetails.xml','ctl00_PageBody_WebPartManager1_gwpPanelDetails_DetailsFrame');</NavURL>
    <?NavRootURL
>javascript:window.parent.FillReport('Filter=&amp;Report=OperatorWells2000.rdl&amp;Key=Companies&amp;Database=NOGCCOnline','ctl00_PageBody_WebPartManager1_gwpPanelDetails_DetailsFrame');</NavRootURL?>
    <Sql>

<![CDATA[WebServiceCall|DataMiningService|DataMiningServiceClient|FindQuery|Wells]]>
    </Sql>
    <SqlAdv>

<![CDATA[WebServiceCall|DataMiningService|DataMiningServiceClient|AdvancedQuery|Wells]]>
    </SqlAdv>
    <KeyName>w.PKey</KeyName>
    <KeyType>Integer</KeyType>
    <SearchType>FullText</SearchType>
    <ThemeColumn></ThemeColumn>
    <ThemeItems>
     <FullTextThemeItem>
      <Value>*</Value>
      <Image>~/images/Ref1.gif</Image>
     </FullTextThemeItem>
    </ThemeItems>
   </FullTextNode>
```

## *Details Example*

In the same way that the Full Text example only modifies the SQL and SQLAdv elements, the Details.xml file is only altered in the baseQuery elements. The formatting of these elements when using a web service is the same as the Full Text search, as follows:

- WebServiceCall|ServiceNamespace|ServiceReferenceName|MethodName|QueryName
- WebServiceCall: Literally says to the Data Mining application that you want it to get the data from a service (i.e. this isn't SQL).
- ServiceNamespace: The namespace of the referenced service. This is entered when you add the service reference.
- ServiceReferenceName: Typically the ServiceNamespace & “Client.”
- MethodName: The name of the service method to be called (as defined in the service contract).

- QueryName: The parameter to be passed to the service method.

```
<?xml version="1.0" encoding="utf-8"?>
<EntityDetailsPersist xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <MainTab>
    <Query>
      <KeyName>PKey</KeyName>
      <KeyValue xsi:type="xsd:integer">1</KeyValue>
      <KeyType>Integer</KeyType>
      <BaseQuery>

<![CDATA[WebServiceCall|DataMiningService|DataMiningServiceClient|DataMiningQuery|EntityI
nformation]]>
      </BaseQuery>
    </Query>
    <RelatedTables />
    <Tabs />
    <ColumnsWide>-1</ColumnsWide>
    <TableTitle>Entity Information</TableTitle>
  </MainTab>
  <Subtabs>
    <!-- General -->
    <EntityDetailsItem>
      <Query>
        <KeyName>EntityKey</KeyName>
        <KeyNameAllowUpdate>false</KeyNameAllowUpdate>
        <KeyValue xsi:type="xsd:integer">1</KeyValue>
        <KeyType>Integer</KeyType>
        <BaseQuery>

<![CDATA[WebServiceCall|DataMiningService|DataMiningServiceClient|DataMiningQuery|Entity
General]]>
        </BaseQuery>
      </Query>
      <RelatedTables>
        <RelatedTable>
          <Query>
            <KeyNameAllowUpdate>false</KeyNameAllowUpdate>
            <KeyName>EntityKey</KeyName>
            <KeyValue xsi:type="xsd:integer">1</KeyValue>
            <KeyType>Integer</KeyType>
            <BaseQuery>

<![CDATA[WebServiceCall|DataMiningService|DataMiningServiceClient|DataMiningQuery|Entity
Communication]]>
            </BaseQuery>
          </Query>
          <Title>Communication</Title>
        </RelatedTable>
      </RelatedTables>
      <Tabs />
      <ColumnsWide>2</ColumnsWide>
      <TableTitle>General</TableTitle>
    </EntityDetailsItem>
```

```
    <!-- Entity Wells -->
    <EntityDetailsItem>
      <Query>
        <KeyName>EntityKey</KeyName>
        <KeyNameAllowUpdate>false</KeyNameAllowUpdate>
        <KeyValue xsi:type="xsd:integer">1</KeyValue>
        <KeyType>Integer</KeyType>
        <BaseQuery />
      </Query>
      <RelatedTables>
        <RelatedTable>
          <Query>
            <KeyNameAllowUpdate>false</KeyNameAllowUpdate>
            <KeyName>Operator</KeyName>
            <KeyValue xsi:type="xsd:integer">1</KeyValue>
            <KeyType>Integer</KeyType>
            <BaseQuery>

<![CDATA[WebServiceCall|DataMiningService|DataMiningServiceClient|DataMiningQuery|Entity
Wells]]>
            </BaseQuery>
          </Query>
        </RelatedTable>
      </RelatedTables>
      <Tabs />
      <ColumnsWide>2</ColumnsWide>
      <TableTitle>Wells</TableTitle>
    </EntityDetailsItem>
  </Subtabs>
</EntityDetailsPersist>
```

## *Data Mining WCF Service Contract*

```
<ServiceContract()> _
Public Interface IDataMiningService
    <OperationContract()> _
    Function DataMiningQuery(ByVal tokenID As Guid, ByVal name As String, ByVal keys() As
KeyData) As System.Data.DataSet

    <OperationContract()> _
    Function FindQuery(ByVal tokenID As Guid, ByVal name As String, ByVal keys() As KeyData)
As System.Data.DataSet

    <OperationContract()> _
    Function AdvancedQuery(ByVal tokenID As Guid, ByVal name As String, ByVal where As
String) As System.Data.DataSet

    <OperationContract()> _
    Function GetLookup(ByVal tokenID As Guid, ByVal name As String) As KeyData()
End Interface
```

```
KeyData Class Definition (as used in IDataMiningService)

Public Class KeyData
   <DataMember()> _
   Public KeyName As String
   <DataMember()> _
   Public KeyType As String
   <DataMember()> _
   Public KeyValue As Object
End Class
```

## *Coding the WCF Service*

Though the inner workings of the service are immaterial to the Data Mining client (i.e., the ASP.NET website), some discussion of the way the service was implemented for Oklahoma is warranted. Basically, we moved the xml configuration files from the client to the server, so now the server has the SQL statements and direct connections to the database. The App_Data folder of the service contains a StateSpecific folder:

- Nodes.xml
- DetailsXML (folder)
  - EntityDetails.xml
  - WellDetails.xml
  - etc...

Several helper classes are included to assist in locating the correct section in the configuration files to pull the SQL from. The results of the SQL are then formatted and returned to the client as a DataSet with a single Table to be used by the caller.

For a full text or advanced query, the rule is that the QueryName (e.g., Wells) must match the FullTextNode.Name value in the Nodes.xml. In the current implementation, the "full text search" is actually performed with LINQ-to-SQL queries as opposed to full text indexing. Therefore, any SQL present in the <Sql> element of the Nodes.xml is only used as a fallback for advanced queries when the <SqlAdv> element is not present.

For a details type query, the rule is that the QueryName (e.g., EntityInformation) must match the Title or TableTitle value in the Details.xml.