Enerline Direct Connect Web Services

Distribution Measurement Service – Usage Guide

Introduction

The Enerline distribution measurement web service is a WCF (Windows Communication Foundation) web service that provides access to the daily measurement data found on the Enerline web site and hourly measurement data for unattended automated processes.

Access and Limitations

In order to use this service, your Enerline user account must have web services access enabled. Please contact Enerline Support for assistance. Note also that before an account can be used with the web services you must have logged into the Enerline website at least once to accept the user agreement.

The web service provides access to the most recent 65 days of detailed measurement data.

The web service must be accessed using a secured (i.e. SSL) connection.

Location and Service Definition

The distribution measurement service is accessible at the following URL on the Enerline web site

https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc

The web service definition (WSDL) file is available at this URL:

https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc?WSDL

Response Data Compression

This web service has support for compressing the response data, which can improve the time to transfer the data. This is enabled by using the “Accept-Encoding” HTTP header in your request. See the Examples below for details.

Web Methods

The distribution web service provides two web methods which are called to retrieve the desired measurement data for daily and hourly respectively. The requests to these methods should be formatted as a standard SOAP document request as specified in the WSDL. Examples are given later.

GetDailyMeasurements Web Method

The GetDailyMeasurements web method retrieves detailed measurement data at daily level and returns it in an XML format.

Parameters

The GetDailyMeasurements web method accepts a single DataContract as parameter. The DataContract contains the following members:

Parameter Name	Member Name	Description
MeasurementRequest	username	Your Enerline user account which has web services access enabled.
	Password	The password that corresponds to the given user account.
	fromDate	Specifies the starting gas day of the requested measurements.
	toDate	Specifies the ending gas day of the requested measurements.
	contractIdList	A list of contracts that should be retrieved (e.g. SA99999). This list can be empty.
	companyIDListd	A list of companies that should be retrieved. All of the accessible distribution contracts are included in the request. Note that this parameter is ignored if **contractIdList** is non-empty

There are three possible ways to request which data you wish to retrieve:

1. Leave both the contractIdList and companyIdList parameters empty. This will retrieve data for all of the contracts that are available to your user account.
2. Specify a list of one or more contracts in the contractIdList parameter. This will return only data for the requested contracts.
3. Specify a list of one or more company IDs in the companyIdList parameter. This will return all data available to your user account for the requested companies. Company ID numbers can be obtained from Enerline Support.

Results

The result will be an XML document containing the detailed measurement data. The structure of this document is described below.

WCF Faults (Exceptions)

In certain cases, this web method will return a fault. These may be due to invalid parameters being given:

Condition	Fault Message
The requested date range is impossible (i.e. **fromDate** is later than **toDate**).	The from date cannot be later than the to date
The requested date range starts in the future.	The from date cannot be in the future
The requested date range ends in the future.	The to date cannot be in the future
The requested date range is earlier than allowed (i.e. **toDate** is more than 65 days in the past).	Measurements earlier than ***<date>*** are not available through this service

They may also be due to authentication or other failures:

Condition	Fault Message
The user could not be logged in.	Invalid login
The user account is locked.	The user account "***<username>***" is locked
The user has not accepted the user agreement	The user account "***<username>***" has not accepted the user agreement on enerline.enbridgegas.com
The user is not authorized to access the web service.	"***<username>***" is not authorized to access this service
An unrecoverable error occurred while processing the request	An unexpected error has occurred. Please contact support.

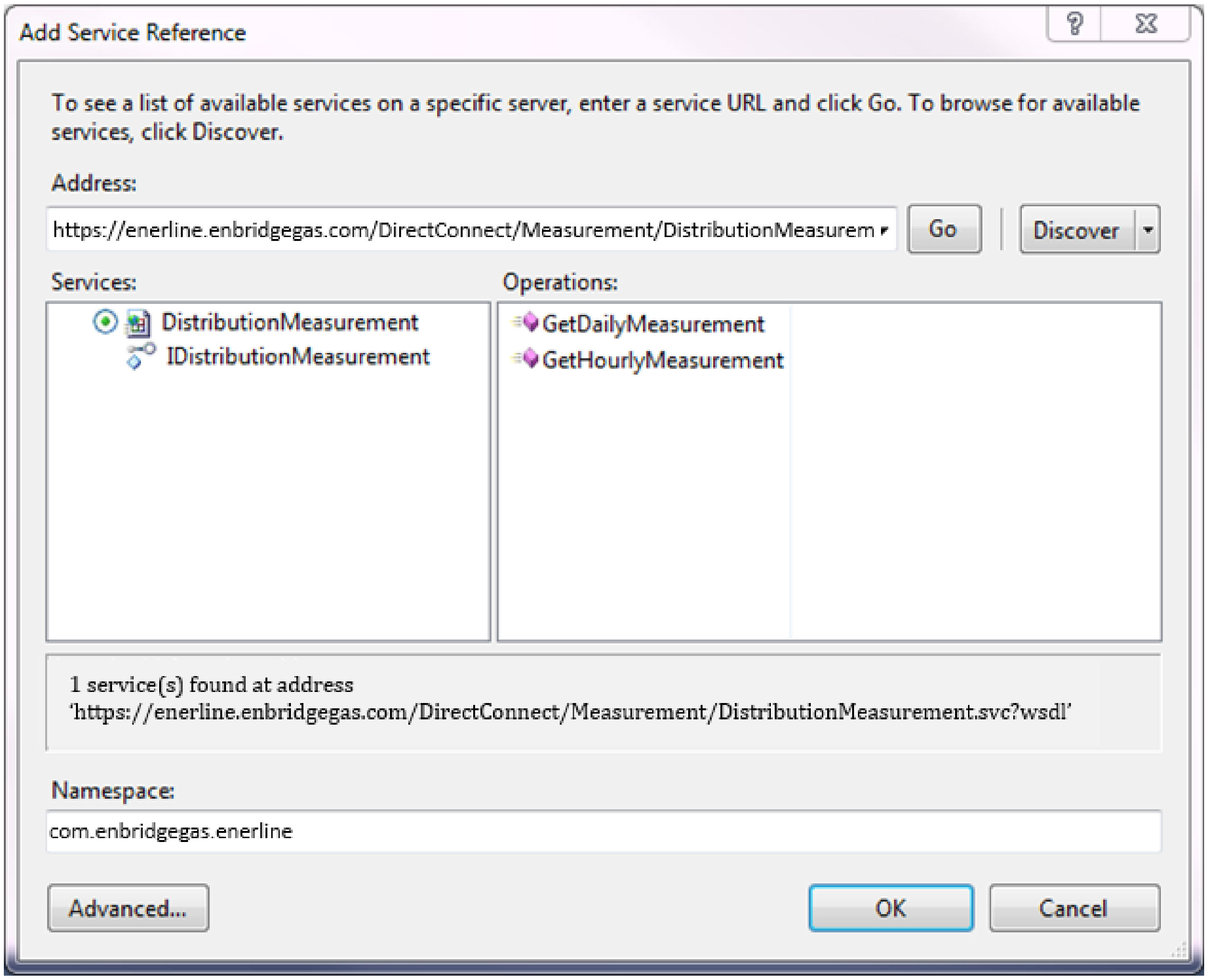
Examples

This section includes some examples of how to get started using the service.

Visual Studio

The steps in this example use Visual Studio 2010. The steps in Visual Studio 2012 are similar.

1. Start a new console application project named "ConsoleApp".
2. Right click on the newly created project in the solution explorer and choose "Add Service Reference…":


Add Service Reference
To see a list of available services on a specific server, enter a service URL and click Go. To browse for available services, click Discover.
Address:
https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurem
Go
Discover
Services:
DistributionMeasurement
IDistributionMeasurement
Operations:
GetDailyMeasurement
GetHourlyMeasurement
Namespace:
com.enbridgegas.enerline
Advanced...
OK
Cancel

1 service(s) found at address 'https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc?wsdl'

Paste the URL from the Location and Service Definition section above into the URL box and click the “Go” button. You might want to change the default name in the “Namespace”box, but in this example we will use the default

3. Click the “OK” button. This will generate a proxy object which will take care of generating the request and processing the result for you.
4. Open the “Program.cs” file and enter the sample code from below to call the WCF web service.
5. Compile and run the application.
6. To prolong the maximum wait time and enable to receive large amount of data, one may need to modify the “<system.serviceModel>” section of the generated “app.config” file.

```
<?xml version="1.0" encoding="utf-8" ?>
   <configuration>
      <system.serviceMo
            del>
           <bindings>
               <basicHttpBinding>
                 <binding name="BasicHttpBinding_IDistributionMeasurement"
                     closeTimeout="00:01:00" openTimeout="00:01:00" receiveTimeout="00:10:00"
                     sendTimeout="00:20:00" allowCookies="false" bypassProxyOnLocal="false"
                     maxBufferPoolSize="5242880" maxReceivedMessageSize="655360"
                     useDefaultWebProxy="true">

                   <readerQuotas maxDepth="32"
                            maxStringContentLength="81920"
                            maxArrayLength="16384" maxBytesPerRead="4096"
                            maxNameTableCharCount="16384" />

                   <security mode="Transport">
                     <transport clientCredentialType="None"
                         proxyCredentialType="None" realm="" />
                     <message clientCredentialType="UserName" algorithmSuite="Default" />
                   </security>
                 </binding>
               </basicHttpBinding>
           </bindings>
         i) <behaviors>
          <endpointBehaviors>

               <behavior name="EndpointBehavior">

                 <dataContractSerializer maxItemsInObjectGraph="2147483647"/>

               </behavior>

          </endpointBehaviors>

           </behaviors>

           <client>
               <endpoint
   address="https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurem
   ent.svc"
                   binding="basicHttpBinding"
   bindingConfiguration="BasicHttpBinding_IDistributionMeasur
   ement"
                   contract="com.enbridgegas.enerline.IDistributionMeas
                   urement"
                   name="BasicHttpBinding_IDistributionMeasurement"
                   behaviorConfiguration="EndpointBehavior" />

           </client>
    </system.serviceModel>
   </configuration>
```

Sample Code if Using Generated WCF Client

This example will call the web service to get data for the past 5 days for the specified contracts. You will need to replace the username and password with those of your Enerline account and then add all or some of your own contracts into the list.

```
        using System;
        using
        System.Collections.Gen
        eric; using
        System.Linq;
        using System.Text;
        using System.ServiceModel;
        using ConsoleApp.com.enbridgegas.enerline;

        namespace ConsoleApp
        {
            class Program
            {
                static void Main(string[] args)
                {
             using (DistributionMeasurementClient service = new DistributionMeasurementClient())
             {
              try {
                service.Endpoint.Address = new
EndpointAddress("https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.
svc?w sdl");
                MeasurementRequest req = new MeasurementRequest();

                Console.WriteLine("Get Daily Measurement --- ");

                req.Username = "jdoe";
                req.Password = "secret";
                req.FromDate = DateTime.Today.AddDays(-5);
                req.ToDate =  DateTime.Today.AddDays(-1);
                req.ContractIds = new string [] { "SA####",
                "SA####" }; req.CompanyIds = new int[] {};

                MeasurementData data = service.GetDailyMeasurement(req);

                foreach (Company company in data.Companies)
                {
                    Console.WriteLine("Company: {0}",
                    company.Name); if (company.Contracts != null)
                    {
                        foreach (Contract contract in company.Contracts)
                        {
                            Console.WriteLine("  Contract: {0} {1}",
                              contract.Id, contract.Name);
                            ShowMeters(contract.Meters);
                            ShowRedeliveryPoints(contract.RedeliveryPoints);
                        }
                    }
                }

                Console.WriteLine("Get Hourly Measurement --- ");

                req.Username = "jdoe";
                req.Password = "secret";
                req.FromDate = DateTime.Today.AddDays(-5);
                req.ToDate = DateTime.Today.AddDays(-5);
             req.ContractIds = new string[] { "SA####", "SA####" };
             req.CompanyIds = new int[] { };

             data = service.GetHourlyMeasurement(req);

             foreach (Company company in data.Companies)
             {
                 Console.WriteLine("Company: {0}",
                 company.Name); if (company.Contract != null)
                 {
                     foreach (Contract contract in company.Contracts)
                     {
                         Console.WriteLine("  Contract: {0} {1}",
                           contract.Id, contract.Name);
                         ShowMeters(contract.Meters);
                         ShowRedeliveryPoints(contract.RedeliveryPoints);
                     }
                 }
              } catch (FaultException<LoginError> ex) {
                    Console.WriteLine("Caught exception : {0}: {1}", ex.Detail.FaultCode,
                                                                  ex.Detail.ErrorMessage);
              } catch (FaultException<CustomError> ex) {
                    Console.WriteLine("Caught exception : {0}: {1}", ex.Detail.Operation,
                                                                  ex.Detail.ErrorMessage);
              }
            }
        }

        static void ShowMeters(Meter[] meters)
        {
            if (meters ==
                null)
                return;

            foreach (Meter meter in meters)
            {
                Console.WriteLine("      Meter: {0} {1} ({2}
                  measurements)", meter.Number, meter.Name,
                  meter.Measurements != null ? meter.Measurements.Length : 0);
            }
        }

        static void ShowRedeliveryPoints(RedeliveryPoint[] rdps)
        {
            if (rdps == null)
                return;

            foreach (RedeliveryPoint rdp in rdps)
            {
                Console.WriteLine("    Redelivery Point: {0}",
                rdp.Name); ShowMeters(rdp.Meters);
            }
        }
    }
}
```

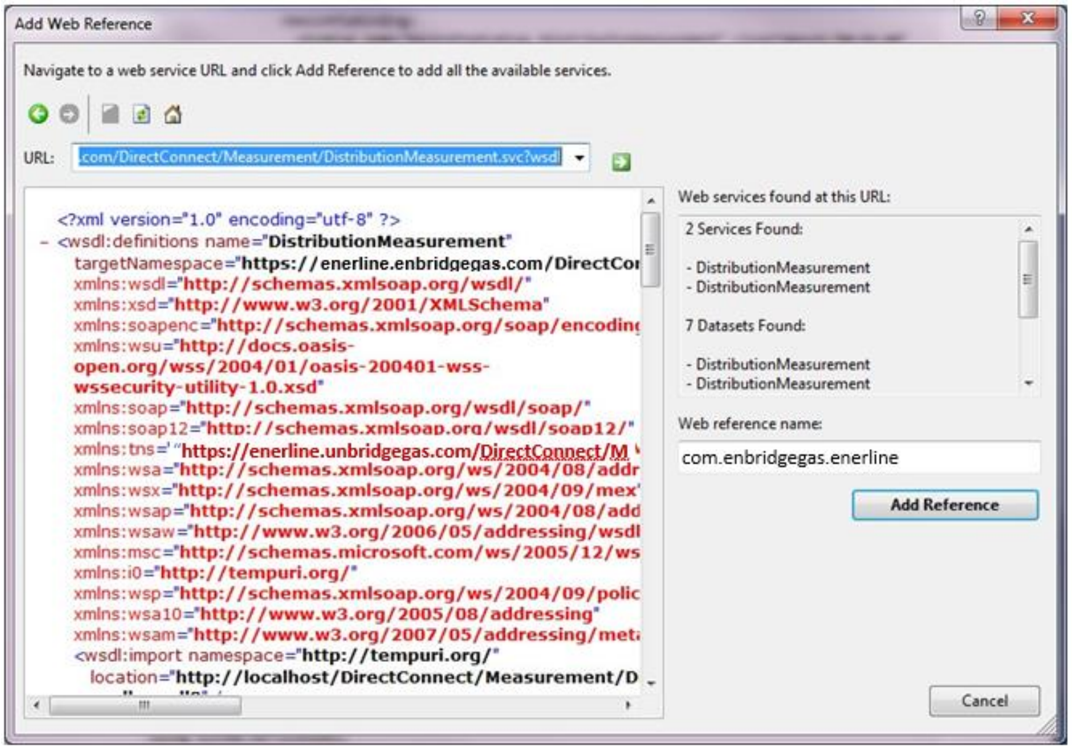
Example Output

This is an example of what the output from the given sample code should look like

```
Get Daily Measurement ---
Company: Your Company Inc.
  Contract: SA#### CONTRACT #1
    Redelivery Point: A Redelivery Point
      Meter: ######## First Meter (5 measurements)
      Meter: ######## Second Meter (5 measurements)
    Redelivery Point: Another Redelivery Point
      Meter: ######## First Meter (5
      measurements) Meter: ######## Second Meter
      (5 measurements)
  Contract: SA#### CONTRACT #2
      Meter: ######## First Meter (5 measurements)
      Meter: ######## Second Meter (5 measurements)
Get Hourly Measurement ---
Company: Your Company Inc.
  Contract: SA#### CONTRACT
  #1
    Redelivery Point: A Redelivery Point
      Meter: ######## First Meter (24 measurements)
      Meter: ######## Second Meter (24 measurements)
    Redelivery Point: Another Redelivery
      Point Meter: ######## First Meter (24
      measurements) Meter: ######## Second
      Meter (24 measurements)
  Contract: SA#### CONTRACT #2
      Meter: ######## First Meter (24
      measurements) Meter: ######## Second Meter
      (24 measurements)
```

Alternate Method

One can also generate Web Reference using old fashion based on .Net Framework 2.0



Navigate to a web service URL and click Add Reference to add all the available services.

URL: .com/DirectConnect/Measurement/DistributionMeasurement.svc?wsdl

```
  <?xml version="1.0" encoding="utf-8" ?>
- <wsdl:definitions name="DistributionMeasurement"
    targetNamespace="https://enerline.enbridgegas.com/DirectCor
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encodinç
    xmlns:wsu="http://docs.oasis-
    open.org/wss/2004/01/oasis-200401-wss-
    wssecurity-utility-1.0.xsd"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:tns=' "https://enerline.unbridgegas.com/DirectConnect/M
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addr
    xmlns:wsx="http://schemas.xmlsoap.org/ws/2004/09/mex'
    xmlns:wsap="http://schemas.xmlsoap.org/ws/2004/08/add
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl
    xmlns:msc="http://schemas.microsoft.com/ws/2005/12/ws
    xmlns:i0="http://tempuri.org/"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/polic
    xmlns:wsa10="http://www.w3.org/2005/08/addressing"
    xmlns:wsam="http://www.w3.org/2007/05/addressing/meta
    <wsdl:import namespace="http://tempuri.org/"
      location="http://localhost/DirectConnect/Measurement/D
```

Web services found at this URL:

2 Services Found:

- DistributionMeasurement
- DistributionMeasurement

7 Datasets Found:

- DistributionMeasurement
- DistributionMeasurement

Web reference name: com.enbridgegas.enerline

Or use the command line tools provided with the .NET platform to generate a proxy for calling the web service, for example:

wsdl.exe
https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc?WSDL
/namespace:ConsoleApp.com.enbridgegas.enerline

Sample Code If Using Generated Soap Client based on .Net 2.0

This example will call the web service that is imported using Web Reference or above wsdl.exe tool. You will need to replace the username and password with those of your Enerline account and then add all or some of your own contracts into the list

```
    using System;
    using
    System.Collections.Generic
    ; using System.Linq;
    using System.Text;
    using System.ServiceModel;

 using ConsoleApp.com.enbridgegas.enerline;

 namespace ConsoleApp
 {
     class Program
     {
         static void Main(string[] args)
         {
             using (DistributionMeasurement service = new DistributionMeasurement())
             {
                 try
                 {

                     service.EnableDecompression = true;
                     service.Url =

                    "https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasureme
                    nt.svc?wsdl";
```

```
            MeasurementData data = service.GetDailyMeasurement(req);

            foreach (Company company in data.Companies)
            {
                Console.WriteLine("Company: {0}", company.Name); if
                (company.Contracts != null)
                {
                    foreach (Contract contract in company.Contracts)
                    {
                        Console.WriteLine("  Contract: {0} {1}",
                          contract.Id, contract.Name);
                        ShowMeters(contract.Meters);
                        ShowRedeliveryPoints(contract.RedeliveryPoints);
                    }
                }
            }

            Console.WriteLine("\nGet Hourly Measurement --- ");

            req.Username = "[redacted]";
            req.Password = "[redacted]";
            req.FromDate = DateTime.Today.AddDays(-5);
            req.ToDate = DateTime.Today.AddDays(-5); req.ContractIds
            = new string[] { "SA[redacted]", "SA[redacted]" }; req.CompanyIds =
            new int[] { };

            data = service.GetHourlyMeasurement(req);

            foreach (Company company in data.Companies)
            {
                Console.WriteLine("Company: {0}", company.Name); if
                (company.Contracts != null)


                   {
                       foreach (Contract contract in company.Contracts)
                       {
                           Console.WriteLine("  Contract: {0}
                             {1}", contract.Id, contract.Name);
                           ShowMeters(contract.Meters);
                           ShowRedeliveryPoints(contract.RedeliveryPoints);
                       }
                   }
               }
           }
           catch (Exception ex)
           {
               Console.WriteLine("Caught exception : {0}: {1}", ex.Message, ex.StackTrace);
           }
       }
   }

   static void ShowMeters(Meter[] meters)
   {
       if (meters ==
           null) return;

       foreach (Meter meter in meters)
       {
           Console.WriteLine("      Meter: {0} {1} ({2}
             measurements)", meter.Number, meter.Name,
                    meter.Measurements != null ? meter.Measurements.Length : 0);
                }
            }

            static void ShowRedeliveryPoints(RedeliveryPoint[] rdps)
            {
                if (rdps == null)
                    return;

                foreach (RedeliveryPoint rdp in rdps)
                {
                    Console.WriteLine("    Redelivery Point: {0}",
                    rdp.Name); ShowMeters(rdp.Meters);
                }
            }
        }
    }
```

Example Output

This is an example of what the output from the given sample code should look like.

```
Get Daily Measurement ---
Company: Your Company Inc.
  Contract: SA#### CONTRACT #1
    Redelivery Point: A Redelivery Point
      Meter: ######## First Meter (5 measurements)
      Meter: ######## Second Meter (5 measurements)
    Redelivery Point: Another Redelivery Point
      Meter: ######## First Meter (5 measurements)
      Meter: ######## Second Meter (5 measurements)
  Contract: SA#### CONTRACT #2
      Meter: ######## First Meter (5
      measurements) Meter: ######## Second
      Meter (5 measurements)
Get Hourly Measurement
--- Company: Your
Company Inc.
  Contract: SA#### CONTRACT #1
    Redelivery Point: A Redelivery Point
      Meter: ######## First Meter (24 measurements)
      Meter: ######## Second Meter (24 measurements)
    Redelivery Point: Another Redelivery Point
      Meter: ######## First Meter (24
      measurements) Meter: ######## Second Meter
      (24 measurements)
  Contract: SA#### CONTRACT #2
      Meter: ######## First Meter (24 measurements)
      Meter: ######## Second Meter (24 measurements)
```

Java 6 JDK

You will need to have the Java6 JDK installed to follow these steps:

1. Use the wsimport tool to generate client code for the web service (enter this entire command on a single line):

   wsimport –p com.enbridgegas.enerline **https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc?WSDL**

   This will generate some compiled .class files containing the code necessary to call the web service. If you would like to keep the source code for these files, you can add the "-keep" option to the above command line.
2. Create a java source file named Program.java containing the sample code given below.
3. Compile the Program.java file and run it.

Sample Code

This example will call the web service to get data for the past 5 years for the specified contracts. You will need to replace the username and password with those of your Enerline account and then add all or some of your own contracts into the list.

```
import java.util.*;
import
javax.xml.bind.JAXBElement;
import javax.xml.datatype.*;
import javax.xml.ws.*;
import com.enbridgegas.enerline.*;

public class Program {

     public static void main(String[]
             args) { try {
                     DistributionMeasurement service = new DistributionMeasurement();

                     IDistributionMeasurement proxy = service
                                     .getBasicHttpBindingIDistributionMeasurement();

                      ((BindingProvider)proxy).getRequestContext().
                             put(
                             BindingProvider.ENDPOINT_ADDRESS_PROPE
                             RTY,

 "https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc?wsdl");

                 //Prepare request parameters
                 MeasurementRequest req = new MeasurementRequest();
                 req.setUsername("jdoe");
                 req.setPassword("secret");

                 DatatypeFactory factory = DatatypeFactory.newInstance();
                 req.setFromDate(factory
                                 .newXMLGregorianCalendar(GetDateByOffsetInDays(-55)));
                 req.setToDate(factory
                                 .newXMLGregorianCalendar(GetDateByOffsetInDays(-51)));

                 ObjectFactory objectFactory = new ObjectFactory();
                 ArrayOfstring contracts = objectFactory.createArrayOfstring();
                 contracts.getString().add("SA####");
                 contracts.getString().add("SA####");
```

```
                JAXBElement<ArrayOfstring> sp = objectFactory
                                .createMeasurementRequestContractIds(contracts);
                req.setContractIds(sp);

                ArrayOfint companies = objectFactory.createArrayOfint();
                JAXBElement<ArrayOfint> ip = objectFactory
                                .createMeasurementRequestCompanyIds(companies);
                req.setCompanyIds(ip);

                System.out.println("Get Daily Measurement ------------> " );
                MeasurementData data = proxy.getDailyMeasurement(req);

                for (Company company : data.getCompanies().getValue().getCompany())
                {
                        System.out.println("Company: " + company.getName().getValue());
                        for (Contract contract : company.getContracts().getValue().getContract())
                        {
                                System.out.println("  Contract: " + contract.getId().getValue()
                                                + " " + contract.getName().getValue());
                                ShowMeters(contract.getMeters().getValue().getMeter());
                                ShowRedeliveryPoints(
                                        contract.getRedeliveryPoints()
                                                .getValue().getRedeliveryPoint());
                        }
                }

                System.out.println("\nGet Hourly Measurement ------------> " );
                data = proxy.getHourlyMeasurement(req);

                for (Company company : data.getCompanies().getValue().getCompany())
                {
                        System.out.println("Company: " + company.getName().getValue());
                        for (Contract contract : company.getContracts().getValue().getContract())
                        {
                                System.out.println("  Contract: " + contract.getId().getValue()
                                                + " " + contract.getName().getValue());
                                ShowMeters(contract.getMeters().getValue().getMeter());
                                ShowRedeliveryPoints(
                                        contract.getRedeliveryPoints()
                                                .getValue().getRedeliveryPoint());
                        }
                }

        } catch (Exception ex) {
                ex.printStackTrace();
        }
}

static void ShowMeters(List<Meter> meters) {
        for (Meter meter : meters) {
                System.out.println("      Meter: " + meter.getNumber().getValue() + " "
                                + (meter.getName().getValue() != null ?
                                         meter.getName().getValue() : "") + " ("
                                + meter.getMeasurements().getValue().getMeasurement().size()
                                + " measurements)");
        }
}

static void ShowRedeliveryPoints(List<RedeliveryPoint> rdps) {
        for (RedeliveryPoint rdp : rdps) {
                System.out.println("    Redelivery Point: " + rdp.getName().getValue());
                ShowMeters(rdp.getMeters().getValue().getMeter());
        }
        }

        private static GregorianCalendar GetDateByOffsetInDays(int offsetInDays) {
                GregorianCalendar result = new GregorianCalendar();
                result.add(GregorianCalendar.DAY_OF_MONTH, offsetInDays);
                return result;
        }
    }
```

Example Output

This is an example of what the output from the given sample code should look like.

```
Get Daily Measurement ------------>
Company: [redacted]
  Contract: [redacted]
    Redelivery Point: [redacted]
      Meter: [redacted]
    Redelivery Point: [redacted]
      Meter: [redacted] (0 measurements)
    Redelivery Point: [redacted]
      Meter:[redacted]  (1 measurements)
      Meter:[redacted]  (1 measurements)
      Meter:[redacted]  (1 measurements)
    Redelivery Point: [redacted]
      Meter: [redacted]  (0 measurements)

Get Hourly Measurement ------------>
Company: [redacted]
  Contract: [redacted]
    Redelivery Point: [redacted]
      Meter: [redacted] (0 measurements)
    Redelivery Point: [redacted]
      Meter: [redacted] (0 measurements)
    Redelivery Point: [redacted]
      Meter:[redacted] (24 measurements)
      Meter: [redacted] (24 measurements)
      Meter: [redacted]  (24 measurements)
       Redelivery Point: [redacted]
         STN) Meter:[redacted]  (0 measurements)
```

Using HTTPS Directly

It is not necessary to use a web services framework to call the service. Instead, the service can be called by directly submitting a POST request to the service over HTTPS. The posted request should have the following form:

```
POST /DirectConnect/Measurement/DistributionMeasurement.wcf
HTTP/1.1 Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction:
"https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData.xsd/IDistributionMeas
urement/ GetDailyMeasurement"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
    <GetDailyMeasurement
xmlns="https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData.xsd">
      <request xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsr="https://enerline.enbridgegas.com/DirectConnect/Measurement/Measurement
Request">
        <xsr:Username>jdoe</xsr:Username>
        <xsr:Password>Secret</xsr:Password>
        <xsr:FromDate>2013-05-08T00:00:00</xsr:FromDate>
        <xsr:ToDate>2013-05-12T00:00:00</xsr:ToDate>
        <xsr:ContractIds xmlns:xsa="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
      <xsa:string>SA####</xsa:string>
      <xsa:string>SA####</xsa:string>
        </xsr:ContractIds>
        <xsr:CompanyIds xmlns:xsa="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
      </request>
    </GetDailyMeasurement>
  </s:Body>
</s:Envelope>
```

The response will be a standard HTTP response containing a SOAP response.

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope>...</soap:Envelope>
```

The actual XML content of the SOAP response can be seen in the Raw XML Sample below.

Sample Code

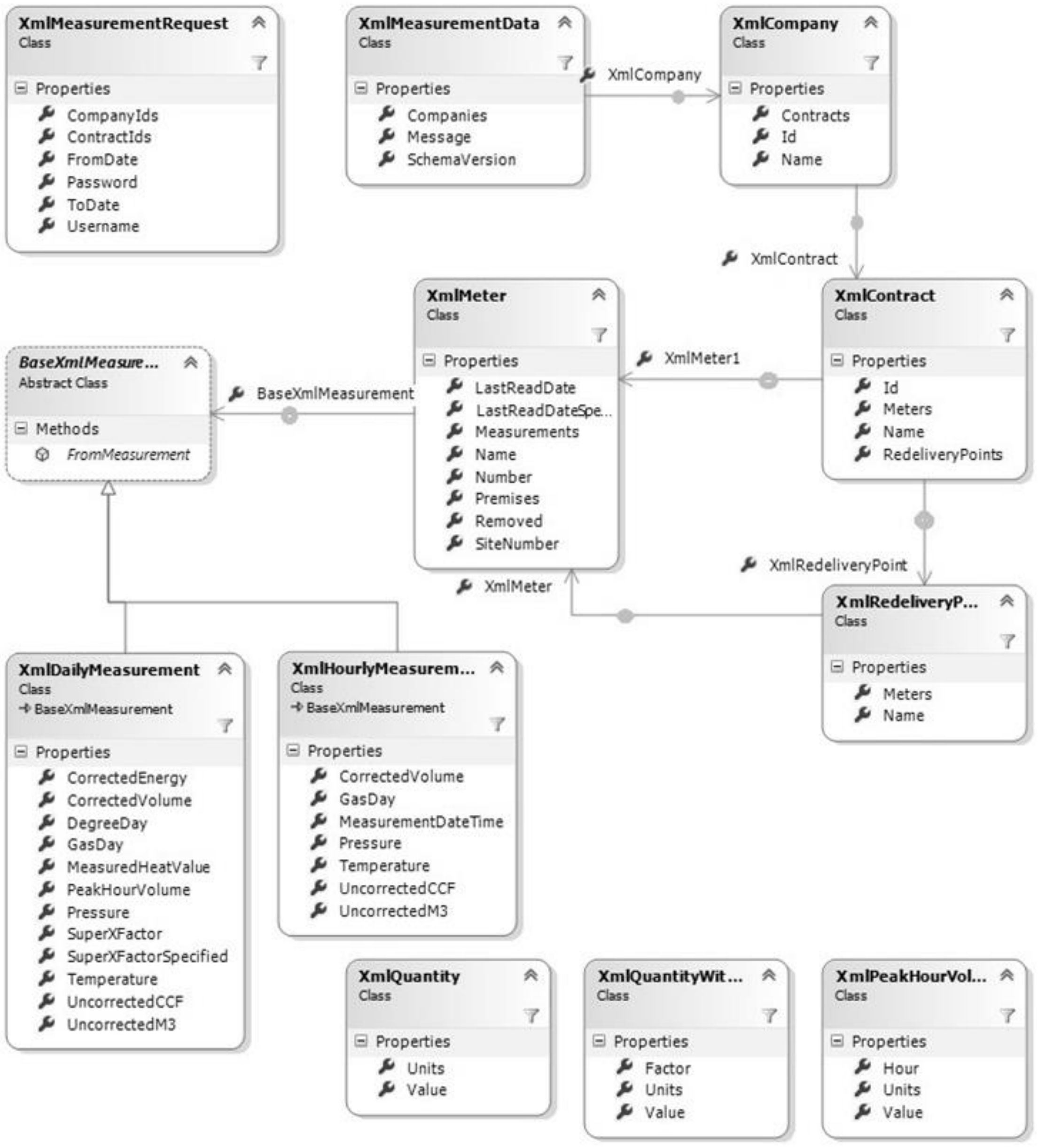
The following example is written in C# for illustrative purposes, but this can be accomplished in a similar manner from any language platform that provides services for making an HTTPS request. This example will call the web service to get data for the past 5 days for the specified contracts. You will need to replace the username and password with those of your Enerline account and then add all or some of your own contracts into the list.

```
using System;
using
System.Collections.Generic;
using System.Linq;
using
System.Net;
using System.IO;
using
System.IO.Compression;
using System.Text;
using System.Xml;

namespace ConsoleApp
{
class Program
  {
    const string MEASUREMENTS_NAMESPACE =
      "https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData.xsd"
      ;
    const string MEASUREMENTS_URL =
      "https://enerline.enbridgegas.com/DirectConnect/Measurement/DistributionMeasurement.svc?wsdl";

    const string SOAP_NAMESPACE= "http://schemas.xmlsoap.org/soap/envelope/";
    const string SCHEMA_MS_ADDR= "http://schemas.microsoft.com/ws/2005/05/addressing/none";
    const string SCHEMA_INSTANCE_NAMESPACE = "http://www.w3.org/2001/XMLSchema-instance";
    const string SCHEMA_NAMESPACE = "http://www.w3.org/2001/XMLSchema";
    const string SCHEMA_REQ_NAMESPACE =
"https://enerline.enbridgegas.com/DirectConnect/Measurement/MeasurementRequest";
    const string SCHEMA_ARRAYS = "http://schemas.microsoft.com/2003/10/Serialization/Arrays";

    static void Main(string[] args)
    {
      string user = "jdoe";
      string password =
      "Secret";
      DateTime fromDate = DateTime.Today.AddDays(-5);
      DateTime toDate = DateTime.Today.AddDays(-1);
      string[] contracts = { "SA####", "SA####" };
      int[] companies = { };

      //Bypass SSL validation
      //ServicePointManager.ServerCertificateValidationCallback =
      //       new System.Net.Security.RemoteCertificateValidationCallback(AcceptAllCertifications);

      WebRequest request = WebRequest.Create(MEASUREMENTS_URL);
      request.Method = WebRequestMethods.Http.Post;

      //Get Daily Measurement
      request.Headers.Add("SOAPAction", MEASUREMENTS_NAMESPACE +
"/IDistributionMeasurement/GetDailyMeasurement");

      //Get Hourly Measurement
      //request.Headers.Add("SOAPAction", MEASUREMENTS_NAMESPACE +
"/IDistributionMeasurement/GetHourlyMeasurement");

      request.Headers.Add(HttpRequestHeader.AcceptEncoding, "gzip,deflate");
      request.ContentType = "text/xml; charset=utf-8";
```

```
          using (XmlWriter writer = new XmlTextWriter(request.GetRequestStream(), Encoding.UTF8))
          {
            WriteGetMeasurementsRequest(


           writer, user, password, fromDate, toDate, contracts, companies);
        }

        WebResponse response; try
        {
          response = request.GetResponse();
        }
        catch (WebException ex)
        {
          Console.WriteLine(ex.Message);
          response = ex.Response;

        }

        Stream responseStream = response.GetResponseStream(); if
        (response.Headers["Content-Encoding"] != null)
        {
          if (response.Headers["Content-Encoding"].ToLower().Contains("gzip"))
            responseStream = new GZipStream(responseStream, CompressionMode.Decompress);
          else if (response.Headers["Content-Encoding"].ToLower().Contains("deflate"))
            responseStream = new DeflateStream(responseStream, CompressionMode.Decompress);
        }

        using (StreamReader reader = new StreamReader(response.GetResponseStream(), Encoding.UTF8))
        {
          Console.WriteLine(reader.ReadToEnd());
        }
      }

      private static void WriteGetMeasurementsRequest(XmlWriter writer, string user, string password,
        DateTime fromDate, DateTime toDate, string[] contracts, int[] companies)
      {
        writer.WriteStartDocument();
        writer.WriteStartElement("s", "Envelope", SOAP_NAMESPACE);

        writer.WriteStartElement("s", "Body", SOAP_NAMESPACE);

        //Get Daily Measurement
        writer.WriteStartElement("GetDailyMeasurement", MEASUREMENTS_NAMESPACE);
        //Get Hourly Measurement
        //writer.WriteStartElement("GetHourlyMeasurement", MEASUREMENTS_NAMESPACE);


        writer.WriteStartElement("request", MEASUREMENTS_NAMESPACE); writer.WriteAttributeString("xmlns",
        "xsi", null, SCHEMA_INSTANCE_NAMESPACE); writer.WriteAttributeString("xmlns", "xsr", null,
        SCHEMA_REQ_NAMESPACE); writer.WriteElementString("Username", SCHEMA_REQ_NAMESPACE, user);
        writer.WriteElementString("Password", SCHEMA_REQ_NAMESPACE, password);
        writer.WriteElementString("FromDate", SCHEMA_REQ_NAMESPACE, fromDate.ToString("yyyy-MM-dd") +
"T00:00:00");
        writer.WriteElementString("ToDate", SCHEMA_REQ_NAMESPACE, toDate.ToString("yyyy-MM-dd") +
"T00:00:00");
        WriteSequence(writer, "ContractIds", "string", contracts);
        WriteSequence(writer, "CompanyIds", "int", companies);
        writer.WriteEndElement();
      writer.WriteEndElement();
      writer.WriteEndElement();
      writer.WriteEndDocument();


    }

    private static void WriteSequence<T>(XmlWriter writer, string elementName, string childName,
      IEnumerable<T> collection)
    {
      writer.WriteStartElement(elementName, SCHEMA_REQ_NAMESPACE);
      writer.WriteAttributeString("xmlns", "xsa", null, SCHEMA_ARRAYS);
      foreach (T item in collection)
          writer.WriteElementString(childName, SCHEMA_ARRAYS, item.ToString());
      writer.WriteEndElement();
    }

    public static bool AcceptAllCertifications(object sender,
                System.Security.Cryptography.X509Certificates.X509Certificate certification,
                System.Security.Cryptography.X509Certificates.X509Chain chain,
                System.Net.Security.SslPolicyErrors sslPolicyErrors)
    {
      return true;
    }
 }
}
```

Measurement Data

The response from the web service will be an XML document contained within a standard SOAP envelope. The following diagram shows the structure of this response (arrows with a dot represent a collection of elements). A detailed description of each element follows.



Raw XML Sample

Below is an example showing the structure of the raw XML result for daily measurement return:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Header />
   <s:Body>
     <GetDailyMeasurementResponse
 xmlns="https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData.xsd">
       <GetDailyMeasurementRes
                 ult
 xmlns:a="https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData
 " xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
        <a:SchemaVersion>2.0</a:SchemaVersion>
         <a:Message xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
         <a:Companies
 xmlns:b="http://schemas.datacontract.org/2004/07/EnbridgeGas.enerline.Infrastructure.WCFServiceModel
 .Cont ract.Measurement">
           <b:Company>
             <b:Id>number</b:Id>
            <b:Name>company name</b:Name>
             <b:Contracts>
             <b:Contract>
               <b:Id>SAnumber</b:Id>
                <b:Name>contract name</b:Name>
               <b:RedeliveryPoints>
                 <b:RedeliveryPoint>
                    <b:Name>
                        redelivery point name
                    </b:Name>
                   <b:Meters>
                     <b:Meter>
                        <b:Number>meter number</b:Number>
                        <b:Premises>premises number</b:Premises>
                      <b:Name>meter name</b:Name>
                        <b:SiteNumber>site number</b:SiteNumber>
                        <b:Measurements>
                         <b:Measurement i:type="b:DailyMeasurement">
                            <b:GasDay>YYYY-MM-DDT00:00:00</b:GasDay>
                            <b:UncorrectedCCF>
                              <b:Value>number</b:Value>
                              <b:Units>CCF</b:Units>
                            </b:UncorrectedCCF>
                            <b:UncorrectedM3>
                              <b:Value>number</b:Value>
                              <b:Units>M3</b:Units>
                            </b:UncorrectedM3>
                            <b:CorrectedVolume>
                              <b:Value>number</b:Value>
                              <b:Units>M3</b:Units>
                            </b:CorrectedVolume>
                            <b:CorrectedEnergy>
                              <b:Value>number</b:Value>
                              <b:Units>GJ</b:Units>
                            </b:CorrectedEnergy>
                            <b:MeasuredHeatValue>
                              <b:Value>number</b:Value>
                              <b:Units>GJ/1000M3</b:Units>
                            </b:MeasuredHeatValue>
                            <b:Pressure>
                              <b:Value>number</b:Value>
                              <b:Units>kPa</b:Units>
                              <b:Factor>number</b:Factor>
                            </b:Pressure>
                            <b:Temperature>
                              <b:Value>number</b:Value>
                              <b:Units>C</b:Units>
                              <b:Factor>number</b:Factor>
                            </b:Temperature>
                            <b:SuperXFactor>number</b:SuperXFactor>
                            <b:PeakHourVolume>
                              <b:Hour>number</b:Hour>
                              <b:Value>number</b:Value>
                              <b:Units>M3</b:Units>
                            </b:PeakHourVolume>
                            <b:DegreeDay>number</b:DegreeDay>
                          </b:Measurement>
                          <b:Measurement i:type="b:DailyMeasurement">
                            ...
                          </b:Measurement>
                        </b:Measurements>
                        <b:Removed>false</b:Removed>
                        <b:LastReadDate i:nil="true" />
                      </b:Meter>
                      <b:Meter>
                        <b:Number>meter number</b:Number>
                        <b:Premises>premises number</b:Premises>
                        <b:Name>meter name</b:Name>
                        <b:SiteNumber>site number</b:SiteNumber>

                        <b:Removed>true</b:Removed>
                        <b:LastReadDate>date</b:LastReadDate>
                       </b:Meter>
                      <b:Meter>...</b:Meter>
                     </b:Meters>
                   </b:RedeliveryPoint>
                  <b:RedeliveryPoint>...</b:RedeliveryPoint>
                </b:RedeliveryPoints>
                <b:Meters />
              </b:Contract>
               <b:Contract>
                <b:Id>SAnumber</b:Id>
                <b:Name>contract name</b:Name>
```

```
                    <b:Meters>...</b:Meters>
                  </b:Contract>
                  <b:Contract>...</b:Contract>
                </b:Contracts>
              </b:Company>
              <b:Company>...</b:Company>
            </a:Companies>
          </GetDailyMeasurementResult>
        </GetDailyMeasurementResponse>
      </s:Body>
    </s:Envelope>
```

Below is an example showing the structure of the raw XML result for hourly measurement return:

```
    <s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
   <s:Header />
   <s:Body>
     <GetHourlyMeasurementResponse
xmlns="https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData.xsd">
       <GetHourlyMeasurementResult
xmlns:a="https://unionline.uniongas.com/DirectConnect/Measurement/MeasurementData"
xmlns:i="http://www.w3.org/2001/XMLSchema-instance">
         <a:SchemaVersion>2.0</a:SchemaVersion>
         <a:Message xmlns:b="http://schemas.microsoft.com/2003/10/Serialization/Arrays" />
         <a:Companies
xmlns:b="http://schemas.datacontract.org/2004/07/EnbridgeGas.enerline.Infrastructure.WCFServiceModel.Cont
ract.Measurement">
           <b:Company>
             <b:Id>number</b:Id>
             <b:Name>company name</b:Name>
             <b:Contracts>
               <b:Contract>
                 <b:Id>SAnumber</b:Id>
                 <b:Name>contract name</b:Name>
                 <b:RedeliveryPoints>
                   <b:RedeliveryPoint>
                     <b:Name>
                    redelivery point name
                     </b:Name>
                     <b:Meters>
                       <b:Meter>
                         <b:Number>meter number</b:Number>
                         <b:Premises>premises number</b:Premises>
                         <b:Name>meter name</b:Name>
                         <b:SiteNumber>site number</b:SiteNumber>
                         <b:Measurements>
                           <b:Measurement i:type="b:HourlyMeasurement">
                             <b:GasDay>YYYY-MM-DDT00:00:00</b:GasDay>
                             <b:MeasurementDateTime>YYYY-MM-DDT00:00:00</b:MeasurementDateTime>
                             <b:UncorrectedCCF>
                               <b:Value>number</b:Value>
```

```
                  <b:Units>CCF</b:Units>
                </b:UncorrectedCCF>
                <b:UncorrectedM3>
                  <b:Value>number</b:Value>
                  <b:Units>M3</b:Units>
                </b:UncorrectedM3>
                <b:CorrectedVolume>
                  <b:Value>number</b:Value>
                  <b:Units>M3</b:Units>
                </b:CorrectedVolume>
                <b:Pressure>
                  <b:Value>number</b:Value>
                  <b:Units>kPa</b:Units>
                  <b:Factor>number</b:Factor>
                </b:Pressure>
                <b:Temperature>
                  <b:Value>number</b:Value>
                  <b:Units>C</b:Units>
                  <b:Factor>number</b:Factor>
                </b:Temperature>
              </b:Measurement>
              <b:Measurement i:type="b:HourlyMeasurement">
                ...
                 </b:Measurement>
               </b:Measurements>
               <b:Removed>false</b:Removed>
               <b:LastReadDate i:nil="true" />
             </b:Meter>
             <b:Meter>
               <b:Number>meter number</b:Number>
               <b:Premises>premises number</b:Premises>
               <b:Name>meter name</b:Name>
               <b:SiteNumber>site number</b:SiteNumber>

               <b:Removed>true</b:Removed>
               <b:LastReadDate>date</b:LastReadDate>
          </b:Meter>
             <b:Meter>...</b:Meter>
        </b:Meters>
      </b:RedeliveryPoint>
         <b:RedeliveryPoint>...</b:RedeliveryPoint>
       </b:RedeliveryPoints>
       <b:Meters />
     </b:Co
     ntract
        >
<b:Contract>
       <b:Id>SAnumber</b:Id>
       <b:Name>contract name</b:Name>
       <b:Meters>...</b:Meters>
     </b:Co
     ntract
```

```
                      >
                    <b:Contract>...</b:Contract>
              </b:Contracts>
               </b:Company>
               <b:Company>...</b:Company>
          </a:Companies>
        </GetHourlyMeasurementResult>
      </GetHourlyMeasurementResponse>
    </s:Body>
</s:Envelope>
```

XML Element Reference

A detailed description of each element follows.

MeasurementData Element

This element is the root of the response.

Attributes

- SchemaVersion – Contains the schema version number for the result. Currently this will be “2.0”.

Children

- **Message** – May be repeated zero or more times. Contains any message generated while processing the request which may have prevent a complete response from being produced. See the remarks below.
- **Companies** – May be repeated zero or more times. Groups the results by company.

Remarks

The following messages may be reported in the response:

Condition	Message
The requested date range was trimmed to fit in the allowed window.	The from date was changed as measurements earlier than ***<date>*** are not available through this service
The requested company is not valid.	The company party id ***<number>***is not valid
The requested company does not have any distribution contracts.	The company with party id ***<number>***has no active distribution contracts
The requested contract is not valid.	The contract id SA***<number>*** is not valid
The requested contract is not a distribution contract.	The contract with id SA***<number>*** is not a valid distribution contract
The requested contract was terminated before the requested date range.	The contract with id SA***<number>*** was terminated before ***<date>***

Company Element

This element groups the results by company.

Children

- Id – The party ID number for the company.
- Name – The name of the company.
- Contracts – May be repeated 1 or more times. Groups the results by contract.

Contract Element

This element groups the results by contract.

Children

- **Id** – The contract ID. This will be of the form “SA<number>”.
- **Name** – The name of the contract.
- **Meters** – May be repeated zero or more times. Groups the results by meter. This element will not be used if the contract has redelivery points.
- **RedeliveryPoints** – May be repeated zero or more times. Groups the results by redelivery point. *This element will only be used if the contract has redelivery points.*

Meter Element

This element groups the results by meter.

Children

- **Name** – The name of the meter.
- **Number** – The number that identifies the physical meter.
- **Premises** – The premises number that identifies the location of the meter.
- **SiteNumber** – The meter site number.
- **Removed** – If this element is present, the meter has been removed from the premises.
- **LastReadDate** – If the meter has been removed: indicates the date that it was last read.
- **Measurements** – May be repeated zero or more times. Contains detailed measurement data for a single gas day or a single gas hour. If the web method is GetDailyMeasurement, the returned measurement type is **DailyMeasurement**. Otherwise, if the web method is GetHourlyMeasurement, the returned measurement type is **HourlyMeasurement**.

RedeliveryPoint Element

This element groups the results by redelivery point.

Children

- **Name** – The name of the redelivery point.
- **Meters** – May be repeated 1 or more times. Groups the results by meter.

DailyMeasurement Element

This element contains the detailed measurement data for a single gas day.

Children

- **GasDay** – Specifies which gas day the measurement is for. In XML, this will be formatted as YYYY-MM-DDT00:00:00.
- **UncorrectedCCF** – The uncorrected volume measurement in CCF. See Quantity Elements below.
- **UncorrectedM3** – The uncorrected volume measurement in m3. See Quantity Elements below.
- **CorrectedVolume** – The corrected volume measurement in m3. See Quantity Elements below.
- **CorrectedEnergy** – The corrected energy consumption value in GJ. See Quantity Elements below.
- **MeasuredHeatValue** – The measured heat value in GJ/103m3. See Quantity Elements below.
- **Pressure** – The measured barometric pressure in kPa. See QuantityWithFactor Elements below.

- **Temperature** – The measured temperature in °C. See QuantityWithFactor Elements below.
- **SuperXFactor** – The SuperX factor.
- **PeakHourVolume** – The peak volume in m3 and the hour in which it occurred. See PeakHourVolume Element below.
- **DegreeDay** – The decimal DegreeDay information.

HourlyMeasurement Element

This element contains the detailed measurement data for a single gas hour.

Children

- **GasDay** – Specifies the Gas Day of the measurement is for. In XML, this will be formatted as YYYY-MM-DDT00:00:00.
- **MeasurementDateTime** – Indicate the time when the measured hour has ended (e.g. 11:00 = measurement from 10:00-11:00). In XML, this will be formatted as YYYY-MM-DDT00:00:00. Please note there will normally have 24 hourly measurement data records returned for a specific gas day.
- **UncorrectedCCF** – The uncorrected volume measurement in CCF. See Quantity Elements below.
- **UncorrectedM3** – The uncorrected volume measurement in m3. See Quantity Elements below.
- **CorrectedVolume** – The corrected volume measurement in m3. See Quantity Elements below.
- **Pressure** – The measured barometric pressure in kPa. See QuantityWithFactor Elements below.
- **Temperature** – The measured temperature in °C. See QuantityWithFactor Elements below.

Remarks

Some values may not be applicable to certain types of contracts, in which case they will not be present in the results.

Quantity Elements

This describes the UncorrectedCCF, UncorrectedM3, CorrectedVolume, CorrectedEnergy and MeasuredHeatValue elements, all of which share a common structure.

Attributes

- Units – The units of the quantity. Depending on the element, this may be “M3”, “CCF”, “GJ” or “GJ/1000M3”.

Text Value

The text value of this element is the numerical value of the quantity, in the units indicated by the Units attribute.

QuantityWithFactor Elements

This describes the Pressure and Temperature elements, which share a common structure.

Attributes

- Units – The units of the quantity. Depending on the element, this may be “kPa” or “C”.

Children

- **Value** – The numerical value of the quantity, in the units indicated by the Units attribute.
- **Factor** – The factor associated with the quantity.

PeakHourVolume Element

This describes a peak volume measurement and the hour during which it occurred.

Attributes

- **Units** – The units of the quantity. The value will be "M3".

Children

- **Hour** – The hour of the day during which the peak volume was measured. This will be a value between 0 and 23 inclusive.
- **Volume** – The numerical value of the volume measurement

Appendix: Comparison of V.2 to V.1

1. Added Degree Day element to the Daily measurement web method.
2. Added web method for hourly measurement data.

3. Changed original 6 parameters (username, password, from date, to date, contract ids, and company ids) to a single parameter as a structure which encloses all previous six parameters.
4. Rewrite the web service with WCF technology.
5. Because the new web service uses WCF DataContract Serializer, original data fields that are displayed as the XML Attributes are now displayed as XML Elements.
6. The new web service will display all elements even if it is Nil due to the WCF DataContract Serialization behaviour.
7. 7All date elements of the new web service are displayed as default XML dateTime element type
8. with format YYYY-MM-DD"T"hh:mm:ss. Although only the date part is effective.
9. All elements in a group (Company, Contract, Redelivery Point, Meter, Measurement) are now placed in their container elements (Companies, Contracts, Redelivery Points, Meters, Measurements).