

## **Introduction**

The Unionline distribution measurement web service is a SOAP (Simple Object Access Protocol) web service that provides access to the measurement data found on the Unionline web site for unattended automated processes.

## **Access and Limitations**

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

Unionline Direct Connect provides access to the most recent 65 days of detailed measurement data.

Unionline Direct Connect 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 Unionline web site:

<https://unionline.uniongas.com/Esurf.Webservice/External/DistributionMeasurement.asmx>

Unionline Direct Connect definition (WSDL) file is available at this URL:

<https://unionline.uniongas.com/Esurf.Webservice/External/DistributionMeasurement.asmx?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 a single web method which is called to retrieve the desired measurement data. The requests to these methods should be formatted as a standard SOAP document request as specified in the WSDL. Examples are given later.

### GetMeasurements Web Method

The GetMeasurements web method retrieves detailed measurement data and returns it in an XML format.

#### Parameters

The GetMeasurements web method accepts the following parameters:

Parameter Name	Description
<b>userName</b>	Your Unionline user account which has web services access enabled.
<b>password</b>	The password that corresponds to the given user account.
<b>fromDate</b>	Specifies the starting date of the requested measurements.
<b>toDate</b>	Specifies the ending date of the requested measurements.
<b>contractIdList</b>	A list of contracts that should be retrieved (e.g. SA99999). This list can be empty.
<b>companyIdList</b>	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 the <b>contractIdList</b> 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 customer service.

#### Results

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



**SOAP Faults (Exceptions)**

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

Condition	Fault Message
The requested date range is impossible (i.e. <b>fromDate</b> is later than <b>toDate</b> ).	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. <b>toDate</b> is more than 65 days in the past).	Measurements earlier than <b>&lt;date&gt;</b> 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 " <b>&lt;username&gt;</b> " is locked
The user has not accepted the user agreement	The user account " <b>&lt;username&gt;</b> " has not accepted the user agreement on unionline.uniongas.com
The user is not authorized to access the Unionline Direct Connect.	" <b>&lt;username&gt;</b> " is not authorized to access this service
An unrecoverable error occurred while processing the request	An unexpected error has occurred. Please contact support.

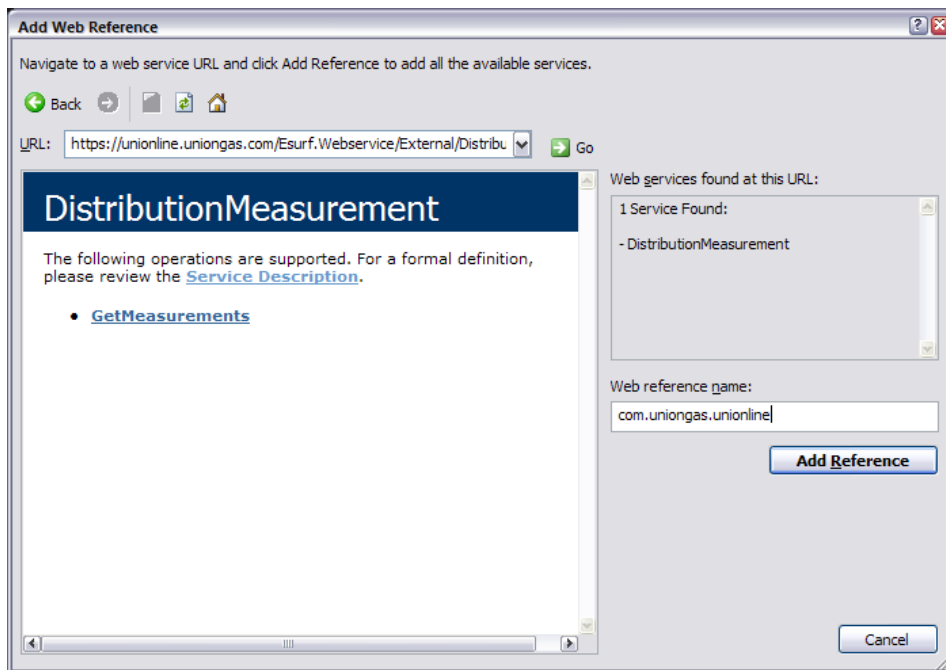
## Examples

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

### Visual Studio

The steps in this example use Visual Studio 2005. The steps in Visual Studio 2008 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 Web Reference...”:



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 “Web reference name” box, but in this example we will use the default.

3. Click the “Add Reference” 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 Unionline Direct Connect.
5. Compile and run the application.

### Alternate Method

An alternative to using the web reference in the steps above is to use the command line tools provided with the .NET platform to generate a proxy for calling Unionline Direct Connect, for example:

```
wsd1.exe https://unionline.uniongas.com/Esurf.Webservice/External/DistributionMeasurement.asmx?WSDL
```

### Sample Code

This example will call Unionline Direct Connect 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 Unionline account and then add all or some of your own contracts into the list.

```
using System;
using System.Collections.Generic;
using System.Text;
using ConsoleApp.com.uniongas.unionline;

namespace ConsoleApp
{
    class Program
    {
        static void Main(string[] args)
        {
            using (DistributionMeasurement service = new DistributionMeasurement())
            {
                service.EnableDecompression = true;

                string user = "jdoe";
                string password = "secret";
                DateTime fromDate = DateTime.Today.AddDays(-5);
                DateTime toDate = DateTime.Today.AddDays(-1);
                string[] contracts = { "SA####", "SA####" };
                int[] companies = { };
                MeasurementData data = service.GetMeasurements(
                    user, password, fromDate, toDate,
                    contracts, companies);

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

        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.Measurement != null ? meter.Measurement.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.Meter);
    }
}
}
```

### Example Output

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

```
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)
```

### 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 Unionline Direct Connect (enter this entire command on a single line):

```
wsimport -p com.uniongas.unionline
https://unionline.uniongas.com/Esurf.Webservice/External/DistributionMeasurement.asmx?WSDL
```

This will generate some compiled .class files containing the code necessary to call Unionline Direct Connect. 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 Unionline Direct Connect 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 Unionline account and then add all or some of your own contracts into the list.

```
import java.util.*;
import javax.xml.datatype.*;
import com.uniongas.unionline.*;

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

            DistributionMeasurementSoap proxy = service.getDistributionMeasurementSoap();

            String user = "jdoe";
            String password = "secret";

            DatatypeFactory factory = DatatypeFactory.newInstance();
            XMLGregorianCalendar fromDate =
                factory.newXMLGregorianCalendar(GetDateByOffsetInDays(-55));
            XMLGregorianCalendar toDate =
                factory.newXMLGregorianCalendar(GetDateByOffsetInDays(-51));

            ArrayOfString contracts = new ArrayOfString();
            contracts.getString().add("SA####");
            contracts.getString().add("SA####");

            ArrayOfInt companies = new ArrayOfInt();

            MeasurementData data = proxy.getMeasurements(
                user, password, fromDate, toDate, contracts, companies);

            for (Company company : data.getCompany()) {
                System.out.println("Company: " + company.getName());
                for (Contract contract : company.getContract()) {
                    System.out.println(" Contract: " + contract.getId() + " " + contract.getName());
                    ShowMeters(contract.getMeter());
                    ShowRedeliveryPoints(contract.getRedeliveryPoint());
                }
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    static void ShowMeters(List<Meter> meters)
    {
```

```
for (Meter meter : meters) {
    System.out.println("    Meter: " +
        meter.getNumber() + " " +
        (meter.getName() != null ? meter.getName() : "") +
        " (" + meter.getMeasurement().size() + " measurements)");
}
}

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

```
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)
```

### 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 /Esurf.Webservice/External/DistributionMeasurement.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: Length
SOAPAction: "https://unionline.uniongas.com/Esurf.Web/Schemas/MeasurementData.xsd/GetMeasurements"
```

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetMeasurements xmlns="https://unionline.uniongas.com/Esurf.Web/Schemas/MeasurementData.xsd">
      <userName>username</userName>
      <password>password</password>
      <fromDate>YYYY-MM-DD</fromDate>
      <toDate>YYYY-MM-DD</toDate>
      <contractIdList>
        <string>SA####</string>
        <string>SA####</string>
      </contractIdList>
      <companyIdList>
        <int>number</int>
        <int>number</int>
      </companyIdList>
    </GetMeasurements>
  </soap:Body>
</soap: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 Unionline Direct Connect 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 Unionline account and then add all or some of your own contracts into the list.

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

namespace ConsoleApp
{
  class Program
```

```
{
    const string MEASUREMENTS_NAMESPACE =
        "https://unionline.uniongas.com/Esurf.Web/Schemas/MeasurementData.xsd";
    const string MEASUREMENTS_URL =
        "https://localhost/Esurf.Webservice/External/DistributionMeasurement.asmx"

    const string SOAP_NAMESPACE= "http://schemas.xmlsoap.org/soap/envelope/";
    const string SCHEMA_INSTANCE_NAMESPACE = "http://www.w3.org/2001/XMLSchema-instance";
    const string SCHEMA_NAMESPACE = "http://www.w3.org/2001/XMLSchema";

    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 = { };

        WebRequest request = WebRequest.Create(MEASUREMENTS_URL);
        request.Method = WebRequestMethods.Http.Post;
        request.Headers.Add("SOAPAction", MEASUREMENTS_NAMESPACE + "/GetMeasurements");
        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;
        }

        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("soap", "Envelope", SOAP_NAMESPACE);
        writer.WriteAttributeString("xmlns", "xsi", null, SCHEMA_INSTANCE_NAMESPACE);
        writer.WriteAttributeString("xmlns", "xsd", null, SCHEMA_NAMESPACE);
    }
}
```

```
writer.WriteStartElement("soap", "Body", SOAP_NAMESPACE);

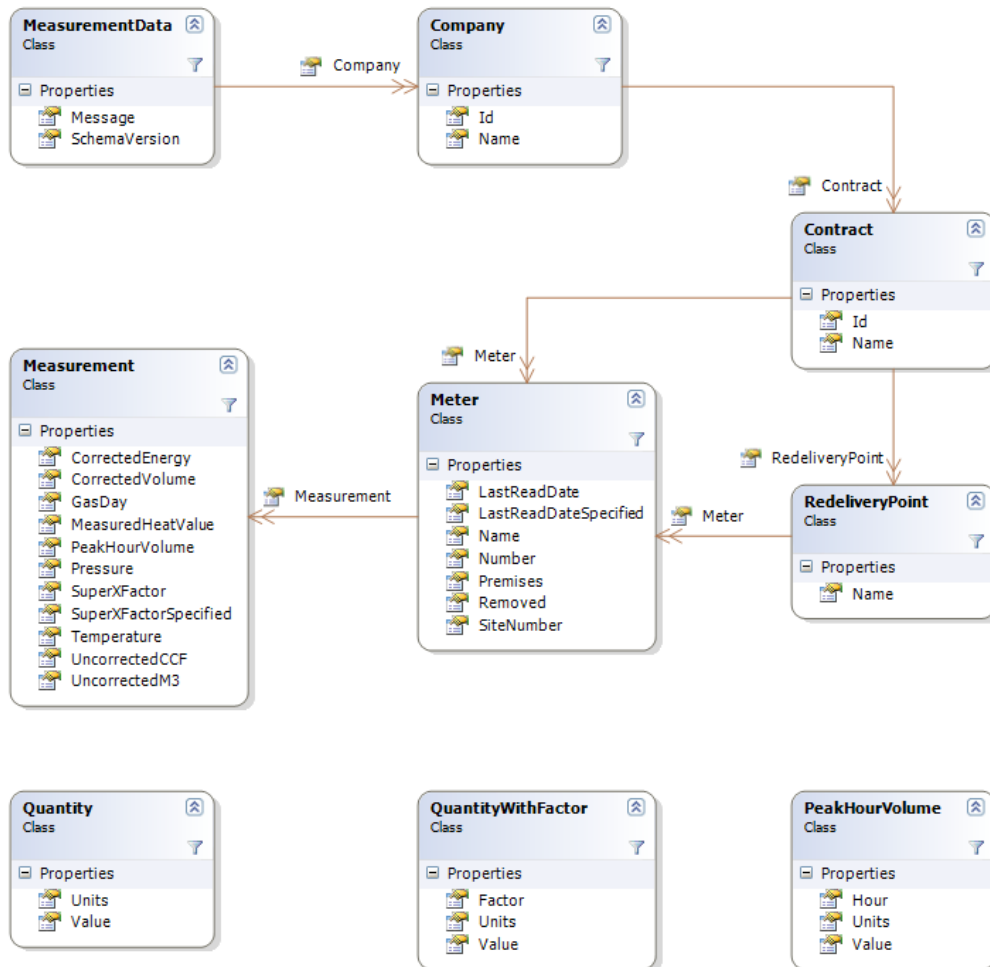
writer.WriteStartElement("GetMeasurements", MEASUREMENTS_NAMESPACE);
writer.WriteElementString("userName", user);
writer.WriteElementString("password", password);
writer.WriteElementString("fromDate", fromDate.ToString("yyyy-MM-dd"));
writer.WriteElementString("toDate", toDate.ToString("yyyy-MM-dd"));
WriteSequence(writer, "contractIdList", "string", contracts);
WriteSequence(writer, "companyIdList", "int", companies);

writer.WriteEndElement();
}

private static void WriteSequence<T>(XmlWriter writer, string elementName, string childName,
    IEnumerable<T> collection)
{
    writer.WriteStartElement(elementName);
    foreach (T item in collection)
        writer.WriteElementString(childName, item.ToString());
    writer.WriteEndElement();
}
}
```

## Measurement Data

The response from Unionline Direct Connect will be an XML document contained within a standard SOAP envelope. The following diagram shows the structure of this response (arrows with double heads 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:

```

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <GetMeasurementsResponse
      xmlns="https://unionline.uniongas.com/Esurf.Web/Schemas/MeasurementData.xsd">

```

```

<MeasurementData SchemaVersion="1.0">
  <Company>
    <Id>number</Id>
    <Name>company name</Name>
    <Contract>
      <Id>SAnumber</Id>
      <Name>contract name</Name>
      <RedeliveryPoint>
        <Name>redelivery point name</Name>
        <Meter>
          <Number>meter number</Number>
          <Premises>premises number</Premises>
          <Name>meter name</Name>
          <SiteNumber>site number</SiteNumber>
          <Measurement>
            <GasDay>YYYY-MM-DD</GasDay>
            <UncorrectedCCF Units="CCF">number</UncorrectedCCF>
            <UncorrectedM3 Units="M3">number</UncorrectedM3>
            <CorrectedVolume Units="M3">number</CorrectedVolume>
            <CorrectedEnergy Units="GJ">number</CorrectedEnergy>
            <MeasuredHeatValue Units="GJ/1000M3">number</MeasuredHeatValue>
            <Pressure Units="kPa">
              <Value>number</Value>
              <Factor>number</Factor>
            </Pressure>
            <Temperature Units="C">
              <Value>number</Value>
              <Factor>number</Factor>
            </Temperature>
            <SuperXFactor>number</SuperXFactor>
            <PeakHourVolume Units="M3">
              <Value>number</Value>
              <Hour>number</Hour>
            </PeakHourVolume>
          </Measurement>
        </Meter>
      </RedeliveryPoint>
    </Contract>
    <Contract>
      <Id>SAnumber</Id>
      <Name>contract name</Name>
      <Meter>...</Meter>
    </Contract>
    <Contract>...</Contract>
  </Company>
  <Removed/>
  <LastReadDate>date</LastReadDate>
</MeasurementData>

```

```
<Company>...</Company>
</MeasurementData>
</GetMeasurementsResponse>
</soap:Body>
</soap: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 “1.0”.

#### 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.
- **Company** – 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 <b>&lt;date&gt;</b> are not available through this service
The requested company is not valid.	The company party id <b>&lt;number&gt;</b> is not valid
The requested company does not have any distribution contracts.	The company with party id <b>&lt;number&gt;</b> has no active distribution contracts
The requested contract is not valid.	The contract id SA <b>&lt;number&gt;</b> is not valid
The requested contract is not a distribution contract.	The contract with id SA <b>&lt;number&gt;</b> is not a valid distribution contract
The requested contract was terminated before the requested date range.	The contract with id SA <b>&lt;number&gt;</b> was terminated before <b>&lt;date&gt;</b>

### Company Element

This element groups the results by company.

#### Children

- **Id** – The party ID number for the company.

- **Name** – The name of the company.
- **Contract** – 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.
- **Meter** – May be repeated zero or more times. Groups the results by meter. *This element will not be used if the contract has redelivery points.*
- **RedeliveryPoint** – 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.
- **Measurement** – May be repeated zero or more times. Contains detailed measurement data for a single gas day.

### RedeliveryPoint Element

This element groups the results by redelivery point.

#### Children

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

### Measurement 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-DD.
- **UncorrectedCCF** – The uncorrected volume measurement in CCF. See Quantity Elements below.

- **UncorrectedM3** – The uncorrected volume measurement in  $m^3$ . See Quantity Elements below.
- **CorrectedVolume** – The corrected volume measurement in  $m^3$ . See Quantity Elements below.
- **CorrectedEnergy** – The corrected energy consumption value in GJ. See Quantity Elements below.
- **MeasuredHeatValue** – The measured heat value in  $GJ/10^3m^3$ . 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  $m^3$  and the hour in which it occurred. See PeakHourVolume Element 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.