OneScotland Gazetteer

Web Service User Guide

Document Version	0.1
Date	July 2010

1 Overview

These web services allow access to the data stored in the OneScotland Gazetteer. They are structured around **datasets**, which contain **fields** which in turn contain the actual data. This approach allows for flexibility, making it easy for suitably authorised users to create new web services or modify existing ones.

This approach means that a single web service is provided to extract data and the caller specifies a dataset on which to operate. So, for example, instead of calling one web service for an address search and a different web service for a UPRN search, you call the same web service for both, but in the first case you specify the "addresses" dataset and in the second you specify the "BLPUs" dataset.

2 Web Services

2.1 Overview

There are two web services available; **sendNGListDataSetsMessage** lists all available datasets whilst **sendNGSearchMessage** allows queries to be performed on those datasets.

So, for example, a dataset of BLPUs might contain fields such as UPRN, Custodian code and last update date. The first web service will let you know that such a dataset exists (and which fields are available) whilst the second web service lets you query that dataset (for example identifying all BLPUs with a last update date of more than a year ago).

Before going on to discuss these web services in detail, we first need to understand how to authenticate for those web services.

2.2 Authentication

In order to use the web service, the user must supply their username and password (the same one they use to log in to the OSG web system). This must be sent for every request and should be supplied in the SOAP header. It consists of a **HeaderLogin** element which contains a **username** and a **password** element, for example:

```
<SoapEnv:Header>
 <ns:HeaderLogin>
  <username>Alice</username>
  <password>secret</password>
 </ns:HeaderLogin>
</SoapEnv:Header>
```

If, for some reason, the authentication fails, the SOAP response will contain an appropriate error message in the header section of the response data. For example, if either the username or password are invalid, the following response would be sent:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="..."
                  xmlns:xsi="... ">
 <SoapEnv:Body>
```

```
  <ns:ListDataSetsResponseMessage
xsi:type="ns1:SearchResponseMessage">
   <NGSearchResponseData>
    <Header>
     <ResultCount>0</ResultCount>
     <ReturnCount>0</ReturnCount>
     <ErrorCode>3</ErrorCode>
     <ErrorMessage>Invalid Username/Password</ErrorMessage>
    </Header>
    <Result>
    </Result>
   </NGSearchResponseData>
  </ns:ListDataSetsResponseMessage>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

2.3 List Datasets (sendNGListDataSetsMessage)

The first of the two available web services is simply to list the available datasets and the fields which those datasets contain. The web service takes no parameters, but does require authentication (as discussed earlier).

```
<SoapEnv:Envelope
   xmlns:SoapEnv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns="http://www.scottishcitizen.gov.uk/improvementservice/
nationalinfrastructure/addresssearch/v0-1">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:listdatasets/>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

Here is an example of what such a call might return (note that future examples will omit the namespace URLs):

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns"...">
 <SoapEnv:Body>
  <ns:ListDataSetsResponseMessage>
   <NGListDataSetsResponseData>
    <DataSet>ADDRESSES</DataSet>
    <Col>Custodian</Col>
    <Col>Easting</Col>
    <Col>Locality</Col>
    <Col>Logical_Status</Col>
    <Col>Northing</Col>
    <Col>PAO_Num</Col>
    <Col>PAO_Text</Col>
    <Col>PostTown</Col>
    <Col>Postcode</Col>
    <Col>SAO_Num</Col>
    <Col>SAO_Text</Col>
    <Col>Street</Col>
    <Col>Town</Col>
```

```
    <Col>UPRN</Col>
   </NGListDataSetsResponseData>
   <NGListDataSetsResponseData>
    <DataSet>BLPU</DataSet>
    <Col>BLPU_State</Col>
    <Col>BLPU_State_Date</Col>
    <Col>Custodian Code</Col>
    <Col>End_Date</Col>
    <Col>Entry_Date</Col>
    <Col>Last_Update_Date</Col>
    <Col>Logical_Status</Col>
    <Col>Parent_UPRN</Col>
    <Col>RPC</Col>
    <Col>Start_Date</Col>
    <Col>UPRN</Col>
    <Col>X_Coordinate</Col>
    <Col>Y_Coordinate</Col>
   </NGListDataSetsResponseData>
  </ns:ListDataSetsResponseMessage>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

The response consists of a **ListDataSetsResponseMessage** element, which contains one or more **NGListDataSetsResponseData** elements. Each of these elements contains information about an individual dataset. The name of the dataset is contained within a **DataSet** element, whilst each column in that dataset is identified by a **Col** element. So, in this example, there are two datasets defined, one called ADDRESSES and one called BLPU. The ADDRESSES dataset contains columns such as Custodian, Easting and Locality whilst the BLPU dataset contains columns such as BLPU_State, BLPU_State_Date and Custodian_Code.

2.4 Query a Dataset (sendNGSearchMessage)

The sendNGSearchMessage allows the user to query a dataset, limiting the results returned based on the data contained within the columns of that dataset. For example, if the BLPU dataset has a Last_Update_Date column, then that could be used to identify any addresses which have been updated in the last month.

A **query** element must be provided for all calls to this web service. The query element requires a mandatory element, **dataset** (the name of the dataset to be queried).

2.4.1 Return a Full Dataset

The simplest query which can be submitted simply asks for all information in a dataset and just consists of the query element and the dataset element (together with the user authentication elements):

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns"...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
```

```
  <ns:query>
   <dataset>BLPU</dataset>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

This example will return the first 250 records in the BLPU dataset (records returned from a dataset are limited to the first 250). The response will be in the following format:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns"...">
 <SoapEnv:Body>
  <ns:SearchResponseMessage>
   <NGSearchResponseData>
    <Header>
     <ResultCount>2824436</ResultCount>
     <ReturnCount>250</ReturnCount>
     <ErrorCode>0</ErrorCode>
     <ErrorMessage>Success</ErrorMessage>
    </Header>
    <Result>

    [CONTENT OMITTED HERE – SEE LATER]

    </Result>
   </NGSearchResponseData>
  </ns:SearchResponseMessage>
 </SoapEnv:Body>
</SOAP-ENV:Envelope>
```

The data returned is contained within an **NGSearchResponseData** element, which is in turn contained within a **SearchResponseMessage** element. The NGSearchResponseData contains two elements, **Header** and **Result**. The Header element contains information about the results returned (notably, how many) and the Result element contains the actual data resulting from the query.

Within the Header element, there are the following elements:

- ResultCount – How many matches were actually found (in this example there are 2,824,436 BLPUs in the dataset).
- ReturnCount – How many of those matches are included in this response. This will either be 250 (the limit on results returned) or the same as ResultCount (if there are less matches than 250).
- ErrorCode – A number indicating if an error was encountered (with 0 representing a successful query).
- ErrorMessage – A textual indication of any errors encountered (or “Success” if no errors occurred).

The result element contains an element for each result returned (i.e. the same number of elements as the value of the ReturnCount element in the Header element).

```
<Result>
 <BLPU>
  <UPRN>906255606</UPRN>
  <Logical_Status>1</Logical_Status>
```

```
  <X_Coordinate>325193</X_Coordinate>
  <Y_Coordinate>673285</Y_Coordinate>
  <RPC>1</RPC>
  <Custodian_Code>9064</Custodian_Code>
  <Start_Date>1993-08-30</Start_Date>
  <Last_Update_Date>2008-11-28</Last_Update_Date>
  <Entry Date>1993-08-30</Entry Date>
 </BLPU>
 <BLPU>
  <UPRN>906255607</UPRN>
  <Logical_Status>1</Logical_Status>
  <X_Coordinate>325160</X_Coordinate>
  <Y_Coordinate>673276</Y_Coordinate>
  <RPC>1</RPC>
  <Custodian_Code>9064</Custodian_Code>
  <Start_Date>1993-08-30</Start_Date>
  <Last_Update_Date>1993-08-30</Last_Update_Date>
  <Entry_Date>1993-08-30</Entry_Date>
 </BLPU>

 [FURTHER RESULTS OMITTED]

</Result>
```

Each element within the Result has the same name as the dataset being queried (BLPU in this example). Each of these elements contains a set of elements, one for each column within the dataset (and with the same name as that column name). For example, within the BLPU dataset there is a UPRN column, so in the results there is a BLPU element for each match returned and within that BLPU element there is a UPRN element.

2.4.2 Attribute-Based Searches

There will be situations in which it is useful to return all results in a dataset (for example “reporting” web services, such as a dataset containing each Custodian and the number of BLPUs they have supplied). However, in most situations users will want to reduce the number of matches returned by providing some sort of search parameters. To achieve this, it is possible to supply attributes and conditions to select subsets of a dataset.

Within the query element of the web service, the caller can supply one or more **attribute** elements. These attribute elements, contain a **name** element (the name of the column that the test is against) and one or more **value** elements (being the value or values used within the test).

Consider a user who wishes to retrieve the information for a specific BLPU for which they know the UPRN. They simply need to add an ‘attribute’ to their query element which contains a ‘name’ element of ‘UPRN’ and a ‘value’ element containing the UPRN they are interested in (for example 906255606). The SOAP request for this would therefore be:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
```

```
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:query>
   <dataset>BLPU</dataset>
   <attribute>
    <name>UPRN</name>
    <value>906255606</value>
   </attribute>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

This is telling the web service to use the dataset 'BLPU' and to only return values where the 'UPRN' is equal to '906255606'. The web service will return a response similar to the following:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Body>
  <ns:SearchResponseMessage>
   <NGSearchResponseData>
    <Header>
     <ResultCount>1</ResultCount>
     <ReturnCount>1</ReturnCount>
     <ErrorCode>0</ErrorCode>
     <ErrorMessage>Success</ErrorMessage>
    </Header>
    <Result>
     <BLPU>
      <UPRN>906255606</UPRN>
      <Logical_Status>1</Logical_Status>
      <X_Coordinate>325193</X_Coordinate>
      <Y_Coordinate>673285</Y_Coordinate>
      <RPC>1</RPC>
      <Custodian_Code>9064</Custodian_Code>
      <Start_Date>1993-08-30</Start_Date>
      <Last_Update_Date>2008-11-28</Last_Update_Date>
      <Entry_Date>1993-08-30</Entry_Date>
     </BLPU>
    </Result>
   </NGSearchResponseData>
  </ns:SearchResponseMessage>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

This is in exactly the same format as the earlier example of a response. They obvious difference is that it now only contains one result (as indicated by both the ResultCount and ReturnCount elements).

It is worth noting briefly what happens in a failure situation. Suppose that we omitted the last digit from the UPRN we supplied as a value. In that case, the response would be as follows:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Body>
  <ns:SearchResponseMessage>
   <NGSearchResponseData>
    <Header>
```

```
      <ResultCount>0</ResultCount>
      <ReturnCount>0</ReturnCount>
      <ErrorCode>0</ErrorCode>
      <ErrorMessage>Success</ErrorMessage>
     </Header>
     <Result>
     </Result>
    </NGSearchResponseData>
   </ns:SearchResponseMessage>
  </SoapEnv:Body>
</SoapEnv:Envelope>
```

This shows that the query was successfully executed (as indicated by the ErrorCode and the ErrorMessage) but that it returned no results (the ResultCount is zero and the Result element contains no sub-elements).

It is also possible for a query to fail, if an unexpected error was encountered. A UPRN should be numeric. If, therefore, a non-numeric UPRN were to be supplied, the web service would instead return a result similar to the following:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Body>
  <ns:SearchResponseMessage>
   <NGSearchResponseData>
    <Header>
     <ResultCount>0</ResultCount>
     <ReturnCount>0</ReturnCount>
     <ErrorCode>2</ErrorCode>
     <ErrorMessage>UPRN: Invalid Number [90625560X]</ErrorMessage>
    </Header>
    <Result>
    </Result>
   </NGSearchResponseData>
  </ns:SearchResponseMessage>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

The difference between this and the previous example is that there is now an ErrorCode and corresponding ErrorMessage, which indicates a failure of the web service to execute.

This is an important distinction to make, since the web service may indicate a successful result which is contrary to user expectations. The web service considers a query to be a success, *even if it returns zero matches*. It will only consider a query to have failed if it cannot execute at all.

Use of the attribute element in this way allows either a single data item to be found (as illustrated above) or multiple data items (for example if the attribute was Logical_Status and the value we supplied was 8, then it would return all historic BLPUs). However, only being able to test for equality is very limiting. It is therefore possible to test attributes in other ways. Consider our earlier example:

```
<ns:query>
 <dataset>BLPU</dataset>
 <attribute>
  <name>UPRN</name>
```

```
  <value>906255606</value>
 </attribute>
</ns:query>
```

This actually uses a special form of the attribute element. The full form of the attribute element also contains a **matchtype** within it. Our example above could be expanded as:

```
<ns:query>
 <dataset>BLPU</dataset>
 <attribute matchtype="equal to">
  <name>UPRN</name>
  <value>906255606</value>
 </attribute>
</ns:query>
```

By supplying other values for matchtype, different tests can be used (although if no matchtype is supplied, it defaults to 'equal to' as illustrated earlier). The possible values of matchtype are:

- equal to – this is a case-insensitive, exact match
- greater than
- greater than or equal to
- less than
- less than or equal to
- not equal to
- in
- not in
- between
- starts with – this is case insensitive
- contains – this is case insensitive

Depending on the matchtype, one or more 'value' elements will need to be supplied. For equal to, greater than, greater than or equal to, less than, less than or equal to, not equal to, starts with and contains, exactly one value element should be supplied. If the match type is 'in' or 'not in', one or more value elements should be supplied. So, the following example will select all BLPUs where the Logical_Status is 1, 5 or 6:

```
<attribute matchtype="in">
 <name>Logical_Status</name>
 <value>1</value>
 <value>5</value>
 <value>6</value>
</attribute>
```

The 'between' matchtype takes exactly two value elements and returns any results where the named attribute is between those two values, *including the values themselves*. So, "between 1 and 3" would include attributes with a value of 1 or 3 as well as attributes with a value of 2.

Note that the number of parameters is not enforced by the web service, but will accept any number of parameters greater than zero. The web services will still behave consistently (although the interpretation may not always be immediately obvious). In general, where a web service expects a single value, if multiple values

are supplied then the web service will return all results which match any of the values (i.e. it is a logical or of the values). So, if the matchtype is “equal to” and the values supplied are “1” and “2”, then the web service will return all records where the field either contains 1 or contains 2.

2.4.3 Combined Searches

A query element can contain multiple attribute elements. In this case, a data item must satisfy all conditions specified across all attributes. The following query would therefore return all historical addresses in Stirling (all matches where Town = STIRLING **and** Logical_Status = 8).

```
  <ns:query>
   <dataset>ADDRESSES</dataset>
   <attribute>
    <name>Town</name>
    <value>STIRLING</value>
   </attribute>
   <attribute>
    <name>Logical_Status</name>
    <value>8</value>
   </attribute>
  </ns:query>
```

2.4.4 Spatial-Based Searches

In addition to searching by attribute, it is also possible to restrict results using spatial searches. These allow searches to be constrained to a given polygon or within a radius of a point.

2.4.4.1 Polygon-Based Searches

This allows the restriction of results to an area as defined by a polygon. Such areas must already be loaded into the system (i.e. they cannot be supplied as part of the web service) and are referred to by name. So, for example, suppose that there exists within the database a polygon named ‘Loch Lomond’. This can then be used to limit the results returned to only those within that polygon.

To limit results to a named polygon, an **area** element must be added to a query, this can either be instead of any **attribute** elements, for example:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:query>
   <dataset>BLPU</dataset>
   <area>1:Loch Lomond NPA</area>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

Or it can be in addition to one or more attribute elements, for example:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:query>
   <dataset>BLPU</dataset>
   <attribute>
    <name>Logical_Status</name>
    <value>8</value>
   </attribute>
   <area>1:Loch Lomond NPA</area>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

This, for example, would return all historical BLPUs contained within the area defined by the Loch Lomond polygon.

If successful (which could include returning zero results), the web service returns a set of elements as described in the earlier attribute search section. On failure, it returns an error code and message. For example, if the user were to supply a name which does not match any polygon in the system, they would receive the following response:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Body>
  <ns:SearchResponseMessage>
   <NGSearchResponseData>
    <Header>
     <ResultCount>0</ResultCount>
     <ReturnCount>0</ReturnCount>
     <ErrorCode>2</ErrorCode>
     <ErrorMessage>Invalid area [X:Loch Lomond NPA]</ErrorMessage>
    </Header>
    <Result>
    </Result>
   </NGSearchResponseData>
  </ns:SearchResponseMessage>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

It is also possible to find all elements which are *outside* the polygon. The ‘area’ tag can take a ‘matchtype’ qualifier, just like the attribute tag. For the ‘area’ tag, the possible values of ‘matchtype’ are ‘in’ and ‘not in’. If no ‘matchtype’ is supplied (as in the examples above), the default is ‘in’. To find all BLPUs outside the Loch Lomond area, the following request could be made:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
```

```
    <username>Alice</username>
    <password>secret</password>
   </ns:HeaderLogin>
  </SoapEnv:Header>
  <SoapEnv:Body>
   <ns:query>
    <dataset>BLPU</dataset>
    <area matchtype="not in">1:Loch Lomond NPA</area>
   </ns:query>
  </SoapEnv:Body>
</SoapEnv:Envelope>
```

The spatial searches will only work on a data set which contains a UPRN field. It will return any data from the dataset where the UPRN for that record represents a BLPU which is within the named polygon.

If a user attempts to query a dataset which does not contain a UPRN, an error will be returned by the web service.

2.4.4.2 Combined Area Searches

Multiple area elements can be included in a dataset query and follow the same principle as combining other elements. Each additional area element forms an additional restriction on the results to be returned. So, a query containing the two elements:

```
<area>Area1</area>
<area>Area2</area>
```

would only return values which are in **both** Area1 and Area2. So, for example, if Area1 is a council boundary and Area2 is all the conservation areas in Scotland, then such a query would return properties in conservation areas in that council area.

A query can also use a "not in" *matchtype* on any or all area elements. So, to continue the above example, we could identify all properties within a council boundary which are not in a conservation area using the following query:

```
<area>Area1</area>
<area matchtype="not in">Area2</area>
```

2.4.4.3 Radius-Based Searches

In a web service call, the user can supply a point (defined by an easting and a northing) and a radius (defined in metres). This will restrict results returned to those within that radius. As with polygon-based searches, this will only work with a data set which contains a UPRN field. Any records in the data set which contain a UPRN field which represents a BLPU within that polygon will be returned.

The radius-based search is invoked by adding a within **element** to the query, which contains an **easting** element, a **northing** element and a **distance** element. For example:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
```

```
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:query>
   <dataset>ADDRESSES</dataset>
   <within>
    <easting>279717</easting>
    <northing>692958</northing>
    <distance>50</distance>
   </within>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

The “within” element can be used by itself (as in the above example) or can be combined with other elements in the query.

If a “within” element is added to a query, then the response will contain an additional element called **Distance** which is not part of the data set. This element contains, for each record, the distance of that item from the point specified by the easting and northing. This can be retrieved like any field from the dataset. Notably, it can also be specified in a “sortField” element, allowing the results to be returned closest first (or furthest first).

2.4.5 Ordering of Results

The results can be sorted by one of the fields in the dataset, by adding a **sortField** element to the query, for example to sort by last update date:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:query>
   <dataset>BLPU</dataset>
   <sortField>Last_Update_Date</sortField>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

By default, this will return results sorted in ascending order (in this example with earliest first). To sort in descending order, a **sortOrder** field can be added to the query and a value of ‘desc’ can be supplied:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
```

```
    <password>secret</password>
   </ns:HeaderLogin>
  </SoapEnv:Header>
  <SoapEnv:Body>
   <ns:query>
    <dataset>BLPU</dataset>
    <sortField>Last Update Date</sortField>
    <sortOrder>desc</sortOrder>
   </ns:query>
  </SoapEnv:Body>
</SoapEnv:Envelope>
```

The sortOrder field can also take a value of ‘asc’ (to return the results in ascending order), which has the same effect as omitting the sortOrder field.

2.4.6 Returning a Count of Matches

Sometimes it is useful to be able to find out how many results match a request, without actually returning the results themselves. To provide such a feature, the ‘query’ tag in the request can take a type=”header” attribute. In this case, the web service only returns the header part of the response and not the matches.

So, for example, if the following request was made:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Header>
  <ns:HeaderLogin>
   <username>Alice</username>
   <password>secret</password>
  </ns:HeaderLogin>
 </SoapEnv:Header>
 <SoapEnv:Body>
  <ns:query type="header">
   <dataset>ADDRESSES</dataset>
  </ns:query>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

The following might be returned:

```
<SoapEnv:Envelope xmlns:SoapEnv="..." xmlns:ns:="...">
 <SoapEnv:Body>
  <ns:SearchResponseMessage>
   <NGSearchResponseData>
    <Header>
     <ResultCount>2197410</ResultCount>
     <ReturnCount>250</ReturnCount>
     <ErrorCode>0</ErrorCode>
     <ErrorMessage>Success</ErrorMessage>
    </Header>
    <Result>
    </Result>
   </NGSearchResponseData>
  </ns:SearchResponseMessage>
 </SoapEnv:Body>
</SoapEnv:Envelope>
```

This indicates that 2,197,410 matches were found (the ‘ResultCount’). It also indicates that if a full request were being made, only 250 of those results would be returned (the ‘ReturnCount’).

Although this example is a basic request for all elements in a dataset, the type=”header” attribute can be added to any query (such as those discussed earlier in this section).

3 Appendices

3.1 Search Request XML (sendNGSearchMessage)

3.1.1 Description

Main entry point into the National Gazetteer Address Search Service. Address search messages should be sent here for the National Gazetteer to process.

3.1.2 Operation type

Request-response. The endpoint receives a message, and sends a correlated message.

3.1.3 SOAP action

```
http://www.scottishcitizen.gov.uk/improvementservice/nationalinfrastructure/
addresssearch/sendNGSearchMessage
```

3.1.4 Input

NGSearchRequestPart *type query*

- dataset – mandatory; *type string*
- attribute - optional, unbounded; *type Attribute_Type*
  - matchtype – optional attribute; *type MatchType*
  - name *type string*
  - value - unbounded; *type string*
- area - optional; *type string*
- within – optional; *type Within_Type*
  - easting *type EastingType*
  - northing *type NorthingType*
  - distance *type decimal*
- sortField - optional; *type string*
- sortOrder - optional; *type SortOrderType*

dataset	The name of the dataset to be queried by the web service
attribute	One or more attributes of the dataset which can be used for filtering the results to be returned. The attributes specified must be from the fields in the dataset. A *matchtype* can be specified (see below for list) with 'equal to' being the default. All comparisons are case-insensitive. For the attribute being specified, a name (of the field) and one or more values must be supplied. Multiple attributes are applied in turn to produce smaller sets of results to be returned (i.e. they are logically anded together).
area	This takes the name of a pre-loaded polygon which can be used to restrict results to items with a UPRN in a geographical area defined by that polygon.

within	If this element is supplied then it must contain three sub-elements, an easting, a northing and a distance. These elements define an area and only items with a UPRN in that area are returned.
sortField	This allows the results to be sorted on one of the fields from the dataset. The value supplied for this must be one of the fields from the dataset.
sortOrder	This takes a value of either asc (for ascending) or desc (for descending) and is applied to the sortField. If no sortOrder is supplied, ascending is used as the default.

MatchType – enum:

- equal to (default)
- greater than
- greater than or equal to
- less than
- less than or equal to
- not equal to
- contains
- between
- in
- not in

SortOrderType - enum:

- asc (default)
- desc

EastingType

- unsignedLong with restriction minInclusive(0) maxInclusive(500000)

NorthingType

- unsignedLong with restriction minInclusive(500000) maxInclusive(1300000)

3.1.5 Output

NGSearchResponsePart *type SearchResponseMessage*

- NGSearchResponseData *type SearchResponseType*
  - Header *type SearchResponseHeaderType*
    - ResultCount *type nonNegativeInteger*
    - ReturnCount *type nonNegativeInteger*
    - ErrorCode *type ErrorCodeType*
    - ErrorMessage *type ErrorMessageType*
  - Result *type SearchResponseResultType*

ResultCount	The number of data items found which match the query
ReturnCount	The number of data items returned by the web service (the minimum of ResultCount and 250)
ErrorCode	Zero for success or non-zero for failure (see later)
ErrorMessage	A message describing an error which occurred (see later)
Result	The data items matching the query. The actual format will depend on the dataset being queried, but it will consist of a set of elements named the same as the dataset, each element containing an

element for each field in the dataset (named the same as that field).

ErrorCodeType - type integer with restriction minInclusive(0) maxInclusive(9999)
ErrorMessageType - type string with restriction maxLength(200) minLength(1)

3.2 WSDL

3.2.1 WSDL URL

The WSDL is available once access to the web services has been approved upon request from the OneScotland Gazetteer Custodian.

- https://portal.onescotlandgazetteer.org.uk/webservice?wsdl

3.2.2 WSDL Content

The WSDL provided by the OneScotland Gazetteer Custodian should be consulted in preference to the version included in this document. At the time of writing, the WSDL is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="NGSearchService"
targetNamespace="http://www.scottishcitizen.gov.uk/improvementservice/nati
onalinfrastructure/addresssearch/v0-1"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://www.scottishcitizen.gov.uk/improvementservice/nationalin
frastructure/addresssearch/v0-1"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types>
    <xsd:schema
targetNamespace="http://www.scottishcitizen.gov.uk/improvementservice/nati
onalinfrastructure/addresssearch/v0-1" elementFormDefault="unqualified"
version="0.1"
xmlns="http://www.scottishcitizen.gov.uk/improvementservice/nationalinfras
tructure/addresssearch/v0-1">
      <xsd:element name="HeaderLogin">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="username" type="xsd:string"/>
            <xsd:element name="password" type="xsd:string"/>
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
      <xsd:simpleType name="SortOrderType">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="asc"/>
          <xsd:enumeration value="desc"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="MatchType">
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="equal to"/>
          <xsd:enumeration value="greater than"/>
          <xsd:enumeration value="greater than or equal to"/>
          <xsd:enumeration value="less than"/>
```

```
          <xsd:enumeration value="less than or equal to"/>
          <xsd:enumeration value="not equal to"/>
          <xsd:enumeration value="contains"/>
          <xsd:enumeration value="starts with"/>
          <xsd:enumeration value="between"/>
          <xsd:enumeration value="in"/>
          <xsd:enumeration value="not in"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="EastingType">
        <xsd:restriction base="xsd:unsignedLong">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="500000"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="NorthingType">
        <xsd:restriction base="xsd:unsignedLong">
          <xsd:minInclusive value="500000"/>
          <xsd:maxInclusive value="1300000"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:complexType name="Attribute_Type">
        <xsd:sequence>
          <xsd:element minOccurs="1" name="name" type="xsd:string"/>
          <xsd:element minOccurs="1" maxOccurs="unbounded" name="value"
                       type="xsd:string"/>
        </xsd:sequence>
        <xsd:attribute name="matchtype" type="MatchType" use="optional"/>
      </xsd:complexType>
      <xsd:complexType name="Within_Type">
        <xsd:sequence>
          <xsd:element minOccurs="1" name="easting" type="EastingType"/>
          <xsd:element minOccurs="1" name="northing" type="NorthingType"/>
          <xsd:element minOccurs="1" name="distance" type="xsd:decimal"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="query">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="1" name="dataset" type="xsd:string"/>
            <xsd:element minOccurs="0" maxOccurs="unbounded"
                         name="attribute" type="Attribute_Type"/>
            <xsd:element minOccurs="0" maxOccurs="1" name="area"
                         type="xsd:string"/>
            <xsd:element minOccurs="0" maxOccurs="1" name="within"
                         type="Within_Type"/>
            <xsd:element minOccurs="0" maxOccurs="1" name="sortField"
                         type="xsd:string"/>
            <xsd:element minOccurs="0" maxOccurs="1" name="sortOrder"
                         type="SortOrderType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="listdatasets">
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="SearchResponseMessage">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="NGSearchResponseData"
                         type="SearchResponseType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="SearchResponseType">
```

```
        <xsd:sequence>
          <xsd:element name="Header" type="SearchResponseHeaderType"/>
          <xsd:element name="Result" type="SearchResponseResultType"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="SearchResponseHeaderType">
        <xsd:sequence>
          <xsd:element name="ResultCount" type="xsd:nonNegativeInteger"/>
          <xsd:element name="ReturnCount" type="xsd:nonNegativeInteger"/>
          <xsd:element name="ErrorCode" type="ErrorCodeType"/>
          <xsd:element name="ErrorMessage" type="ErrorMessageType"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:complexType name="SearchResponseResultType">
        <xsd:sequence>
          <xsd:any minOccurs="0"/>
        </xsd:sequence>
      </xsd:complexType>
      <xsd:element name="SearchException">
        <xsd:complexType>
          <xsd:sequence/>
        </xsd:complexType>
      </xsd:element>
      <xsd:simpleType name="ErrorCodeType">
        <xsd:restriction base="xsd:integer">
          <xsd:minInclusive value="0"/>
          <xsd:maxInclusive value="9999"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType name="ErrorMessageType">
        <xsd:restriction base="xsd:string">
          <xsd:maxLength value="200"/>
          <xsd:minLength value="1"/>
        </xsd:restriction>
      </xsd:simpleType>
      <xsd:element name="ListDataSetsResponseMessage">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element minOccurs="0" maxOccurs="unbounded"
                         name="NGListDataSetsResponseData"
                         type="ListDataSetsResponseType"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="ListDataSetsResponseType">
        <xsd:sequence>
          <xsd:element name="DataSet" type="xsd:string"/>
          <xsd:element minOccurs="1" maxOccurs="unbounded" name="Col"
                       type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </types>
  <message name="HeaderLoginMessage">
    <part name="HeaderLogin" element="tns:HeaderLogin"/>
  </message>
  <message name="NGSearchMessageSoapIn">
    <part name="NGSearchRequestPart" element="tns:query"/>
  </message>
  <message name="NGSearchMessageSoapOut">
    <part name="NGSearchResponsePart"
          element="tns:SearchResponseMessage"/>
  </message>
  <message name="NGSearchFaultSoap">
    <part name="NGSearchFaultDetail" element="tns:SearchException"/>
  </message>
  <message name="NGListDataSetsMessageSoapIn">
```

```
    <part name="NGListDataSetsRequestPart" element="tns:listdatasets"/>
  </message>
  <message name="NGListDataSetsMessageSoapOut">
    <part name="NGListDataSetsResponsePart"
          element="tns:ListDataSetsResponseMessage"/>
  </message>
  <portType name="NGSearchIntf">
    <operation name="sendNGSearchMessage">
      <documentation>Main entry point into the National Gazetteer
                     Search Service. Search messages should be
                     sent here for the National Gazetteer to
                     process.</documentation>
      <input message="tns:NGSearchMessageSoapIn"/>
      <output message="tns:NGSearchMessageSoapOut"/>
      <fault name="fault" message="tns:NGSearchFaultSoap"/>
    </operation>
    <operation name="sendNGListDataSetsMessage">
      <documentation>List searchable data sets provided by the National
                     Gazetteer Search Service.</documentation>
      <input message="tns:NGListDataSetsMessageSoapIn"/>
      <output message="tns:NGListDataSetsMessageSoapOut"/>
    </operation>
  </portType>
  <binding name="SubmitMessageSoapHttp" type="tns:NGSearchIntf">
    <soap:binding style="document"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="sendNGSearchMessage">
      <soap:operation
soapAction="http://www.scottishcitizen.gov.uk/improvementservice/nationali
nfrastructure/addresssearch/sendNGSearchMessage" style="document"/>
      <input>
        <soap:body use="literal"/>
        <soap:header message="tns:HeaderLoginMessage" part="HeaderLogin"
                     use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
      <fault name="fault">
        <soap:fault name="fault" use="literal"/>
      </fault>
    </operation>
    <operation name="sendNGListDataSetsMessage">
      <soap:operation
soapAction="http://www.scottishcitizen.gov.uk/improvementservice/nationali
nfrastructure/addresssearch/sendNGSearchMessage"/>
      <input>
        <soap:body use="literal"/>
        <soap:header message="tns:HeaderLoginMessage" part="HeaderLogin"
                     use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>
  <service name="NGSearchService">
    <documentation>National Gazetteer Search Web Service</documentation>
    <port name="NGSearchService" binding="tns:SubmitMessageSoapHttp">
      <soap:address
location="https://devportal.onescotlandgazetteer.org.uk/webservice"/>
    </port>
  </service>
</definitions>
```

3.3 Response Codes and Messages

The web service can return 4 possible error codes.

An error code of zero indicates a successful submission to the web service.

An error code of 1 indicates that there was an error with the XML sent to the web service. This will return an error message such as:

- `Invalid XML - Schema validation error : <error msg>`

An error code of 2 indicates that the XML was syntactically correct, but some other issue was encountered whilst processing the request (for example requesting a dataset which does not exist). The error messages which can be returned are:

- `Invalid dataset [<dataset>]`
- `Invalid attribute name for dataset <dataset> [<attr_name>]`
- `<attr_name>: Invalid Date [<value>] expected format YYYY-MM-DD`
- `<attr_name>: Invalid Number [<value>]`
- `Invalid area [<area>]`
- `The dataset <dataset> does not support filtering on area`
- `polygon (<area>) awaiting initial index build - please try again later`
- `Invalid sort_field for dataset <dataset> [<sort_field>]`

An error code of 3 indicates an authentication error. The error messages which can be returned are:

- Invalid Username/Password
- The address your computer is accessing this site from (xxx.xxx.xxx.xxx) has been blocked due to excessive failed login attempts. To have this block removed, please contact <contact email>. Alternatively, the block will be automatically removed in two hours time.
- An unknown login failure occurred. The administrators have been notified of the event.
- You do not have permission to use this service.
- Missing authentication header in soap request