

# **Electronic Transfer of Geotechnical and Geoenvironmental Data using XML data format**

**(August 2005  
Progress Report)**

2005  
Commissioned by  
Association of Geotechnical and Geoenvironmental Specialists



## ACKNOWLEDGEMENTS

This document has been prepared by the Association of Geotechnical and Geoenvironmental Specialists (AGS) with the encouragement and support of the working party members. The AGS acknowledges the generous time and resources given to the project by the individual members and their employers. Without their enthusiastic support this ongoing project would not be possible.

Comment and feedback on this document is encouraged from the wider geotechnical industry.

## AGSML Working Party Members

Roger Chandler  
David Toll  
Paul Quinn  
Dan Cornford  
Aleid Bosch  
James Passmore

Key Systems Geotechnical Ltd / Keynetix Ltd  
Durham University  
Key Systems Geotechnical Ltd  
Aston University  
NTO (Netherlands)  
BGS

## Amendments

Edition No.	Date of issue	Amendment
1	August 2005	Original Issue

## Association of Geotechnical and Geoenvironmental Specialists

Forum Court  
83 Copers Cope Road  
Beckenham  
Kent BR3 1NR  
UNITED KINGDOM

Tel.: 020 8658 8212 Fax.: 020 8663 0949

Email: [ags@geotechnical.demon.co.uk](mailto:ags@geotechnical.demon.co.uk)

website: <http://www.ags.org.uk>

Although every effort has been made to check the accuracy of the information and validity of the guidance given in this document, neither the members of the Working Party nor the Association of Geotechnical and Geoenvironmental Specialists accept any responsibility for misstatements contained herein or misunderstanding arising here from.

<b>1. INTRODUCTION .....</b>	<b>5</b>
<b>2. GEOGRAPHICAL MARKUP LANGUAGE (GML) .....</b>	<b>5</b>
2.1. Locating boreholes .....	6
<b>3. DATA DICTIONARY .....</b>	<b>7</b>
3.1. Group Names .....	7
3.2. Field Names .....	8
3.3. Tags vs Attributes? .....	8
<b>4. DATA VALIDATION .....</b>	<b>8</b>
<b>5. ADVANCED AGS FEATURES .....</b>	<b>8</b>
5.1. Multiple Projects.....	8
5.2. The AGS CODE group.....	9
5.3. The AGS ABBR group .....	9
5.4. The AGS FILE group.....	9
5.5. The AGS UNIT group .....	10
5.6. User Defined Fields.....	10
<b>APPENDIX A .....</b>	<b>11</b>
Schema Implementation Notes .....	11
A.2 Schema Breakdown .....	11
A.3 GML.....	11
<b>APPENDIX B .....</b>	<b>13</b>
B.1 Extending AGSML with user defined types .....	13
B.2 PROJ Group.....	13
B.3 HOLE Group.....	14
B.4 SAMP Group.....	15
B.5 CLSS Group .....	16
B.6 DICT Group .....	17
B.7 ABBR and CODE Groups.....	18

---

<b>B.8 UNITS Group .....</b>	<b>19</b>
<b>B.9 FILE Group .....</b>	<b>20</b>
<b>APPENDIX C .....</b>	<b>22</b>
<b>AGSML Data Structure .....</b>	<b>22</b>

## 1. Introduction

In 2003 the AGS commissioned a working party to review the capabilities of XML data and advise the main AGS Data Format committee on whether the use of XML in the next version of AGS (or alongside the current version) would give members of the AGS additional benefits.

The working party is made up of existing members of the AGS Data Format group, leading academics and international representation.

The working party reviewed the capabilities of XML and came to the following conclusions:-

- By adopting an XML file format the AGS would be able to embrace, and be compatible with, other XML based formats, such as GML. It could also be defined as a sub set of a larger Geotechnical data format that may include sub structure information.
- By using existing AGS field and group names and the existing data hierarchy the format would be fully backwardly compatible.
- By embracing XML the AGS would be ideally positioned to work on international data format projects that could benefit AGS members either directly or indirectly.
- The working party should continue its work to outline how an XML implementation of AGS could be achieved.

This report is the first published report from the working party and its primary aim is to give AGS members and external organisations an update on the working party's current work and proposed methods of implementation.

A set of schemas and example files have been produced to accompany this report and these files can be downloaded from the AGSML website at [www.ags.org.uk/agsml](http://www.ags.org.uk/agsml)

The working party welcomes feedback to the ideas and methods detailed in this report. Feedback can either be done publicly using the AGS discussion boards or can be privately made by email the comments to [agsxml@ags.org.uk](mailto:agsxml@ags.org.uk).

It should be noted that since the end of the working party first phase the group has been working with COSMOS, Federal Roads Authority and Florida University in the preparation of an international format. This group's work will build on the findings in this report and progress will be posted on the AGSML website and in the next report.

## 2. Geographical Markup Language (GML)

GML is data transfer format used for transporting geographic information. It provides a means to define geometry, topology, coordinates reference systems, time, units of measure, generalized values and features types in a consistent manner. A GML feature is a meaningful object in the domain being modelled. This could be a road, city, person, project, boundary or a borehole.

GML is an international standard maintained by the Open Geospatial Consortium and it has a growing number of software tools available to process, analyse and display it. Incorporating GML into AGSML allows AGS data to be used by any application that can process GML.

GML has predefined types for coordinates and features and AGSML can make use of these, rather than re-inventing the wheel.

GML is maintained by the Open Geospatial Consortium. Their website is a good starting point for information on GML: <http://www.opengeospatial.org/> . The official documentation for GML 3.1 from the Open Geospatial Consortium can be found at [http://portal.opengeospatial.org/files/?artifact\\_id=4700](http://portal.opengeospatial.org/files/?artifact_id=4700) , and all of the GML schemas can be downloaded from <http://schemas.opengis.net/gml/3.1.1/base/> .

There are two options available to make AGSML GML compatible.

- 1) Only the groups that contain data that is geographically located are made GML compatible (i.e HOLE and PROJ), or
- 2) Every table is made GML compatible.

The working party recommends and has implemented option 2 for the following reasons.



- 1) Software capable of processing GML features, such as a Web Feature Service, can process AGSML and return data from any table within the AGSML file. If GML was only implemented at the borehole level, then GML compatible software could only process borehole data rather than the individual tables.
- 2) It enables flexibility for the future, and the option to easily add GML elements to any of the tables. These GML elements could be used to add Meta data, specify coordinates, assign ID's or add time stamps.
- 3) Making each group GML compatible does not add any additional complexity to an AGSML file, as all the work is included in the schema file.

### 2.1. Locating boreholes

AGS 3.1 defines the coordinates of a borehole by fields in the HOLE table. National coordinates are given by the HOLE\_NATE, HOLE\_NATN and HOLE\_GL headings respectively. Local coordinates are given by X, Y and Z values, under the HOLE\_LOCX, HOLE\_LOCY and HOLE\_LOCZ headings, respectively.

GML allows us to use its predefined mechanisms for specifying coordinates, which would replace the headers currently used in AGS.

In AGSML, national and local coordinates can be specified with the holeLoc element. This element is a GML point array property, and is used to specify coordinates as below:

```
<holeLoc>
  <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
    <gml:pos>123 456</gml:pos>
  </gml:Point>
</holeLoc>
```

The <gml:pos> element contains the X, Y and if necessary Z coordinates of the borehole, separated by a blank space. If 3 dimensional coordinates are used, the optional attribute srsDimension can be used as shown below:

```
<gml:pos srsDimension="3">123 456 789</pos>
```

The coordinate space is defined by the attribute srsName (where "srs" stands for spatial reference system). This attribute locates an XML file that describes the coordinate reference system (CRS) being used. This allows users to choose an existing CRS or to define and publish their own.

A borehole can be located using more than one coordinate reference system. For example, GML elements can be used to define national and local coordinates. This is done by having multiple <gml:Point> elements under a holes <holeLoc> element, with each one referencing a different coordinate systems. This is shown below:

```
<Hole>
  .....
  <holeLoc>
    <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:pos>123 456</gml:pos>
    </gml:Point>
    <gml:Point srsName="http://www.localRefSystem.xml/#p2">
      <gml:pos>789 998</gml:pos>
    </gml:Point>
  </holeLoc>
</Hole>
```

However, it is not necessary to have the srsName attribute on every gml:Point element. In GML, when a CRS has been defined at one level, all coordinates under that level use that CRS by default, unless another CRS is stated. The highest level this can be done in an AGSML file is by having bounding box under the root element. A bounding box, declares the area which all points must lie within. In an AGSML file, this will be the perimeter of the sites being investigated. This is shown below:

```
<SiteInvestigation>
  <gml:boundedBy>
    <gml:Envelope srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:lowerCorner>0 0</gml:lowerCorner>
      <gml:upperCorner>100 100</gml:upperCorner>
    </gml:Envelope>
  </gml:boundedBy>
  <!--Rest of AGSML file goes here-->
</SiteInvestigation>
```

From the above example, all of the gml points will now use the CRS defined at <http://www.opengis.net/gml/srs/epsg.xml#4326> by default. An individual point can use a different CRS by explicitly including the srsName attribute on the gml:Point element.

### 3. Data Dictionary

#### 3.1. Group Names

In AGS group names are upper-case, for example HOLE, SAMP and GEOL. However this is considered bad practice in XML and so the group names are in Pascal case (this means the first letter is upper-case and the remaining letters are lower case, for example Hole, Samp or Geol). As XML is case sensitive it is important that this convention is adhered to. A file containing the element <hole>, instead of <Hole> would be invalid.

The core concept in GML is an object, and an object has properties. This object-property model repeats over and over throughout a GML file. The main benefit of the object-property model is that we can describe the role of the objects contained by another object.

So if we simply had

```
<Hole>
  <gml:pos>123 456</gml:pos>
</Hole>
```

there is very little meaning that we can say other than the hole contains coordinates, but what do the coordinates mean? Are they part of a national or local CRS? Is it a single point, or is it part of a line or polygon? Adding some properties allows us to provide some meaning to the role of the <gml:pos> in <Hole>.

```
<Hole>
  <holeLoc>
    <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:pos>123 456</gml:pos>
    </gml:point>
  </ holeLoc >
</Hole>
```

<Hole> is an object representing a borehole. <holeLoc> is a property whose value is the location of a borehole. <gml:Point> is an object representing the point the borehole exists. <gml:pos> is a property whose value is "123 456". It is now clear that what the meaning of the <gml:pos> element is. In AGSML, property elements are in lower camel case. This means that all letters are lower case, except the first letter of the second word.

The hierarchy of tables in AGSML is identical to AGS, for example the SAMP table is still a child of the HOLE table, but each entry in a SAMP table is contained within a property element, which describes the relationship between HOLE and SAMP. This is shown below:

```
<Proj>
  <holes>
    <Hole>
      <holeInformation>
        <Samp>
          <laboratoryTesting>
            <Clss/>
            <Cbrg/>
          </laboratoryTesting >
        <Samp>
      </holeInformation >
    </Hole>
  </holes>
</Proj>
```

In the above example, <Hole> and <Samp> are GML objects, and <holeInformation> is a property identifying the relationship between the two.

<Clss> and <Cbrg> are GML objects, and < laboratoryTesting > is the relationship they have with their parent, <Samp>.

The breakdown of object-property relationships of AGS tables is shown in appendix C.

### 3.2. Field Names

Field names are in lower camel case and the underscores have been removed. For example, HOLE\_FDEP is now written holeFdep.

The only fields to have been changed are the coordinate fields HOLE\_NATN, HOLE\_NATE, HOLE\_LOCX, HOLE\_LOCY as these have been replaced by the GML positioning methods. AGS elements such as HOLE\_GL and HOLE\_Z will still be used, however users can add a third dimension to the <gml:pos> element if they wish by specifying the srsDimension attribute on the gml:pos element.

### 3.3. Tags vs Attributes?

XML allows data to be held either as the contents of an element or as an attribute of an element. Below is an example using the content of an element:

Example 1:

```
<Hole>
  <holeId>BH501</holeId>
  <holeType>CP</holeType>
  etc.....
</Hole>
```

Below are two examples using attributes:

Example 2:

```
<Hole>
  <holeId id="BH501"/>
  <holeType type="CP"/>
  etc....
</Hole>
```

Example 3:

```
<Hole id="BH501" type="CP" etc.... />
```

AGSML uses elements (example 1) as this is the recommended method for GML applications.

## 4. Data Validation

The rules to which a valid AGS 3.1 file must conform to are written in the AGS 3.1 documentation. However the rules an XML file must conform to are declared in a file called a schema. This includes rules to ensure key fields are unique, all the values are the correct type (a number, date or letters), and that the parent – child relationship is maintained between tables.

The rules in the schema can be used to validate an AGSML file automatically by standard XML software tools (such as Internet Explorer) or websites such as <http://apps.gotdotnet.com/xmltools/xsdvalidator/Default.aspx> and produce a list of error messages where appropriate.

## 5. Advanced AGS Features

### 5.1. Multiple Projects

You can have more than one project in an AGSML file by enclosing the project elements within <SiteInvestigation> and <projects> elements. <SiteInvestigation> was chosen as the root element because a single site investigation can include several projects.

Example:

```
<SiteInvestigation>
  <projects>
    <Proj>
      <projId>BH501</projId>
      <projName>Name of Project</projName>
    </Proj>
```

```

    <Proj>
      <projId>TP502</projId>
      <projName>Another Project Name</projName>
    </Proj>
  </projects>
</SiteInvestigation>

```

## 5.2. The AGS CODE group

The AGS 3.1 CODE group is used to store the abbreviations used in the CNMT (Contamination and Chemical Testing) group. For example, "AL" is the abbreviation for Aluminium and "CTET" is the abbreviation for Carbon tetrachloride.

In AGSML, all of the standard codes are stored in a separate file called codes.xml. This is available on the AGS website <http://www.ags.org.uk/agsml/v1/codes.xml>. If an AGSML file only uses the standard codes then it will reference this file with the above URL in the code attribute of the Proj element:

```
code = "http://www.ags.org.uk/agsml/v1/codes.xml"
```

If a file uses user defined codes then the file must reference a different code.xml file either on a website or sent locally with the file. If an absolute path is used, then this is used to locate the file, if only a file name is used then the file is assumed to be in the same directory as the AGSML file. This allows the software processing an AGSML file to check whether the standard set of codes are being used, and if not, where to get non-standard list of codes from.

## 5.3. The AGS ABBR group

The ABBR table is implemented identically to the CODE table.

The ABBR table is used to store the abbreviations used in the AGSML file. For example, D under the DISC\_TERM heading is the abbreviation for "Terminates against another discontinuity", and INST under the HOLE\_TYPE headings is the abbreviations for "Instrument".

In AGSML, all of the standard abbreviations are stored in a separate file called abbrs.xml. This will be available on the AGS website <http://www.ags.org.uk/agsml/v1/abbrs.xml>. If an AGSML file is using the standard abbreviations it will reference abbrs.xml on the AGS website from the abbr attribute on the Proj element

```
abbr = "http://www.ags.org.uk/agsml/v1/abbrs.xml"
```

If a file uses user defined codes then the file must reference a different abbrs.xml file either on a website or sent locally with the file. If an absolute path is used, then this is used to locate the file, if only a file name is used then the file is assumed to be in the same directory as the AGSML file. This allows software processing an AGSML file to check whether the standard set of abbreviations are being used, and if not, where to get non-standard codes.

## 5.4. The AGS FILE group

AGS 3.1 allows external files to be included within a data file using an FSET field in each group that references a file in the FILE group.

AGSML allows users to include external files using a File element, which has attributes locating and describing the attached file. This is shown below:

```

<Hole>
  <HoleId>TP501</HoleId>
  <HoleType>TP</HoleType>

  Hole Data.....

  <files>
    <File xlink:type="locator" xlink:href="Photo.jpg"
      xlink:label="Photo1"
      xlink:title="Photo of Borehole TP501">
      <openWith>PaintShop Pro</openWith>

```

```

        <date>2004-11-24</date>
        <type>jpg</type>
    </File>
</files>
</Hole>

```

This uses an XML technology called XLink to reference external files. In the above example, all attributes beginning with “xlink:” belong to the XLink namespace. Elements have been added to further describe the file to ensure backwards compatibility with AGS 3.1. In the above example, the file “Photo.jpg” has been added to the borehole with the id TP501. The href parameter may refer to either the local path or a URL for files accessible on the internet.

### 5.5. The AGS UNIT group

The file units.xml is available on the AGS website at <http://www.ags.org.uk.agsml/v1/units.xml>. This file declares all of the units used by AGS in a standard format. Unless specified otherwise within an AGSML file, these standard units will be used. If a header within an AGSML file requires units, the attribute UoM (Units of Measurement) is used to specify what the unit is. This is shown below:

```

<Geol>
    <geolTop UoM="#ft">1.5</geolTop>
    .....More Geol Data
</Geol>

```

The above fragment of AGSML specifies that geolTop unit of measure is “ft”, where “ft” is specified as “Feet” in the standard dictionary.

If non-standard units are used, a user can define their own Units.xml file, with their own unit declarations. The “UoM” attribute can then point to the user defined unit file. This is shown below:

```

<Geol>
    <GeolTop UoM="http://MyCompany.com/Units.xml#mFt">1.5</GeolTop>
    .....More Geol Data
</Geol>

```

The above fragment specifies that the unit of measure is “mFt” as defined in the “units.xml” file available at <http://MyCompany.com>. If the “UoM” attribute is missing, the default AGS units for that header are assumed.

The working party recognise that there are standard methods for specifying units and will be looking at this in the next phase.

### 5.6. User Defined Fields

User defined fields can be achieved in AGSML by extending the rules defined in the AGSML Schema. A schema is a set of rules defining the parent – child relationship of tables, the data types each headers can take and which fields form the unique key. If you placed your own tables or headers within an AGSML file, validation errors would occur, as the schema would not recognise your tables and headers. This can be resolved by referencing your own schema which in turn references the AGSML schema and includes your own groups or headers and information on how they relate to existing groups. When the AGSML file is validated the new groups and headers will be recognised from your schema, and the file will be valid.

## Schema Implementation Notes

This section of the report covers the technical schema details of the items above. Some of the information is repeated for clarity.

Information is included for the PROJ, HOLE, SAMP and CLSS groups. Information for the other groups will follow these examples and have therefore not been included in this document.

### A.2 Schema Breakdown

The AGSML schema is broken down into 7 separate schemas. The reasons for this were as follows

- 1) Easier to maintain seven smaller schema files than one very large file.
- 2) User are able to import only the schema files they need for their XML instance file, rather than the entire set of AGS tables.

The tables are broken down into the following categories: field testing, ground information, hole information, monitoring and laboratory testing. The break down of tables is shown in appendix C. There is also a base AGSML schema file that is included by all AGSML schema files and a top level schema file. The base schema includes elements and type definitions used by all other schema files. The top level schema defines the SiteInvestigation, Proj and Hole types, and includes all other schemas.

### A.3 GML

GML is a means of transporting geographic information and types. GML provides predefined geometry and features types. This allows AGSML to be consumed by any application that can process GML. It also enables software to consume a web service via a standard interface returning standard geographic types. The Open Geospatial Consortium has defined a Web Feature Service. This provides a standard interface for accessing GML data via a web service, which retrieves GML Feature objects. By implementing a standard interface, anyone can write applications to access GML and process the data for their needs.

GML also has predefined types for coordinates and features. AGSML can make use of these, rather than re-inventing the wheel.

Each AGSML table will be defined as a GML feature to provide compatibility with GIS systems. GML coordinates will be used to identify their locations.

It has been proposed that the AGS coordinate headers (HOLE\_NATE, HOLE\_NATN, HOLE\_GL, HOLE\_LOCX, HOLE\_LOCY, HOLE\_LOCZ) will be replaced with the GML position elements, and that HOLE\_LETT will not be required. This was proposed so AGSML would rely on the GML coordinate system, rather than the AGS system, to force users to comply with the GML standard.

Other entities defined by depth below surface (Samp, geolTop, geolBase) will still be defined by depth since determination of coordinates requires calculation (particularly for inclined holes). GML position elements could be added in future versions for these entities but would not be required.

Alternative coordinate sets will be provided, each referencing a different coordinate system:

```
<Hole>
  .....
  <holeLoc>
    <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
      <gml:pos>123 456</gml:pos>
    </gml:Point>
    <gml:Point srsName="http://www.localRefSystem.xml/#p2">
      <gml:pos>789 998</gml:pos>
    </gml:Point>
  </holeLoc>
</Hole>
```

AGSML files will be able to contain more than one coordinate system. For example a hole can have local and national coordinates.

In the above example "URN:National CRS" is the URI representing the National Grid Datum, and "URN:Local CRS" is another URI representing a different coordinate system.

### GML Implementation

The GML requirement of alternating Feature/Property/Feature will be followed. In general terms, the existing AGS groups are *Features* and AGS headings within groups are *Properties*. This means that Feature/Feature connections would not be allowed; such as currently exist between Project-Hole etc. Array properties will be used to limit the number of unnecessary tags eg:

```
<SiteInvestigation>
  <projects>
    <Proj>
      <holes>
        <Hole>
          ...
        </Hole>
        <Hole>
          <holeInformation>
            <Samp>
              ...
            </Samp>
            <Samp>
              ...
            </Samp>
          </ holeInformation >
        </Hole>
      </holes>
    </Proj>
  </projects>
</SiteInvestigation>
```

## Appendix B

### B.1 Extending AGSML with user defined types

Elements are ordered (i.e. elements would be required in a defined sequence) as this improves the efficiency of import of data. Another reason is that use of <sequence> allows the maximum number of occurrences of an element (maxOccurs) to be unbounded, while <all> is limited to a maxOccurs of 1. Since AGSML files will be produced by software packages, it was not felt this would cause difficulties for data producers.

The GML naming system would be adopted. Features (i.e. the AGS groups) would start with a capital letter (Proj, Hole). Properties would use camelcase (holes, samps, clsss).

Documentation annotation would be provided within the schema, thus avoiding the need for a separate document to define the tags. Description and Example properties would be added (using the AGS definitions). A style sheet will be developed to produce a formatted readable document listing the tags, with definitions and examples.

Numeric fields will be assigned either *float* or *integer* types. Text fields will be defined as *string* types. Date fields will use the *date* type which requires ISO standard dates (yyyy-mm-dd). Where a numeric field could also have a text entry (e.g. Non plastic or Dry) then a union of types will allow either a numeric value or an enumerated string list of allowable entries

### B.2 PROJ Group

A project is represented as a GML AbstractFeatureType. Multiple projects can be stored within one AGSML file by creating a collection of <Proj> elements within <projects> under <SiteInvestigation> elements, this is shown below:

```
<SiteInvestigation>
  <projects>
    <Proj>
      <projId>P567</projId>
      <projName>Sewerage Works</projName>
    </Proj>
    <Proj>
      <projId>P678</projId>
      <projName>Sewerage Works II</projName>
    </Proj>
  </projects>
</SiteInvestigation>
```

Software can add headers to the Proj group by extending the agsml type AbstractProjType. The example below adds two headers to the "Proj" table. The first called "elemOne" of type string, the second is called "elemTwo" of type float:

```
<xs:complexType name="ProjType">
  <xs:complexContent>
    <xs:extension base="agsml:AbstractProjType">
      <xs:sequence>
        <xs:element name="elemOne" type="xs:string"/>
        <xs:element name="elemTwo" type="xs:float"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The element can then be defined as follows:

```
<xs:element name="Proj" type="ProjType"
substitutionGroup="agsml:_ProjectsFeature"/>
```

The above code is placed in the users own schema, which imports the main AGSML schema.

To add a group under the projects element, software can define a type which extends AGSMLBase type, and is in the agsml:\_ProjectsFeature substitution group.

```

<complexType name="MyGroupType">
  <xs:complexContent>
    <xs:extension base="gml:AgSmlBaseType">
      <xs:sequence>
        <xs:element name="myElem1" type="xs:string"/>
        <xs:element name=" myElem2" type="xs:string"/>
        <xs:element name=" myElem3" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

The element is then defined as:

```
<xs:element name="MyGroup" type="MyGroupType" substitutionGroup="gml:_ProjectsFeature"/>
```

In an instance document would like:

```

<projects xmlns:myDict="http://MyWebsite/MyAGSMLTypes">
  <Proj>
    <projId>P1</projId>
    <projName>Trumpington Sewerage Works</projName>
    <myDict:elemOne>My first userdefined element in the PROJ group</myDict:elemOne>
    <myDict:elemTwo>1234</myDict:elemTwo>
  </Proj>
  <myDict:MyGroup>
    <myDict:myElem1>This is a user define group under the projects element</myDict:myElem1>
    <myDict:myElem2>These are elements allowed within the group </myDict:myElem2>
    <myDict:myElem3>Only these elements are allowed</myDict:myElem3>
  </myDict:MyGroup>
</projects>

```

### B.3 HOLE Group

Hole elements must be contained within a <holes> tag. The <holes> tag represents the relationship between the parent tag <Proj> and the child tag <Hole>.

Software can add headers to the Hole group by extending the AbstractHoleType. This is shown below:

```

<xs:complexType name="HoleType">
  <xs:complexContent>
    <xs:extension base="agsml:AbstractHoleType">
      <xs:sequence>
        <xs:element name="elemOne" type="xs:string"/>
        <xs:element name="elemTwo" type="xs:float"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

The element can then be defined as follows:

```
<xs:element name="Hole" type="HoleType" substitutionGroup="agsml:_HoleFeature"/>
```

The above code is placed in the users own schema, which imports the main AGSML schema.

To add a group under the Hole element, software can define a type which extends agsml:AgSmlBaseType, and is in the agsml:\_HoleFeature substitution group.

```

<complexType name="MyHoleGroupType">
  <xs:complexContent>
    <xs:extension base="agsml:AgsmIBaseType">
      <xs:sequence>
        <xs:element name="myElem1" type="xs:string"/>
        <xs:element name=" myElem2" type="xs:string"/>
        <xs:element name=" myElem3" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

The element is then defined as:

```

<xs:element name="MyHoleGroup" type="MyHoleGroupType"
  substitutionGroup="agsml:_ HoleFeature"/>

```

An instance document would look like:

```

<holes xmlns:myDict="http://MyWebsite/MyAGSMLTypes">
  <Hole>
    <holeId>P1</projId>
    <holeType>TP</holeType>
    <myDict:elemOne>My first userdefined element in the HOLE group</myDict:elemOne>
    <myDict:elemTwo>1234</myDict:elemTwo>
  </Proj>
  <myDict:MyGroup>
    <myDict:myElem1>This is a user define group under the holes element</myDict:myElem1>
    <myDict:myElem2>These are elements allowed within the group </myDict:myElem2>
    <myDict:myElem3>Only these elements are allowed</myDict:myElem3>
  </myDict:MyGroup>
</holes>

```

#### B.4 SAMP Group

<Samp> elements must be contained within a <holeInformation> tag. The <holeInformation> tag represents the relationship between the parent tag <Hole> and the child tag <Samp>.

Software can add headers to the Samp group by extending the AbstractSampType. This is shown below:

```

<xs:complexType name="SampType">
  <xs:complexContent>
    <xs:extension base="agsml:AbstractSampType">
      <xs:sequence>
        <xs:element name="elemOne" type="xs:string"/>
        <xs:element name="elemTwo" type="xs:float"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

The element can then be defined as follows:

```

<xs:element name="Samp" type="SampType" substitutionGroup="agsml:_HoleinformationFeature"/>

```

The above code is placed in the users own schema, which imports the main AGSML schema.

To add a group under the <Samp> element, software can define a type which extends GML AbstractFeatureType, and is in the <agsml:\_HoleinformationFeature> substitution group.

```

<xs:complexType name="MySampGroupType">
  <xs:complexContent>
    <xs:extension base="gml:AbstractFeatureType">
      <xs:sequence>
        <xs:element name="myElem1" type="xs:string"/>
        <xs:element name=" myElem2" type="xs:string"/>
        <xs:element name=" myElem3" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

The element is then defined as:

```

<xs:element name="MySampGroup" type="MySampGroupType"
  substitutionGroup="agsml:_HoleInformationFeature"/>

```

An instance document would look like:

```

<holeInformation xmlns:myDict="http://MyWebsite/MyAGSMLTypes">
  <Samp>
    <sampTop>0</ sampTop >
    <sampBase>0.5</SampBase>
    <myDict:elemOne>My first userdefined element in the SAMP group</myDict:elemOne>
    <myDict:elemTwo>1234</myDict:elemTwo>
  </Proj>
  <myDict:MyGroup>
    <myDict:myElem1>This is a user define group under the holeInformation element</myDict:myElem1>
    <myDict:myElem2>These are elements allowed within the group </myDict:myElem2>
    <myDict:myElem3>Only these elements are allowed</myDict:myElem3>
  </myDict:MyGroup>
</holes>

```

## B.5 CLSS Group

Software can add headers to the Clss group by extending the AbstractClssType. This is shown below:

```

<xs:complexType name="ClssType">
  <xs:complexContent>
    <xs:extension base="agsml:AbstractClssType">
      <xs:sequence>
        <xs:element name="elemOne" type="xs:string"/>
        <xs:element name="elemTwo" type="xs:float"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

```

The element can then be defined as follows:

```

<xs:element name="Clss" type="ClssType" substitutionGroup="agsml:_SampFeature"/>

```

The above code is placed in the users own schema, which imports the main AGSML schema.

To add a group under the Clss element, software can define a type which extends agsml:AbstractSampleType, and is in the agsml:\_LabTestingFeature substitution group.

```

<complexType name="MyClssGroupType">
  <xs:complexContent>
    <xs:extension base="agsml:SampleType">
      <xs:sequence>
        <xs:element name="myElem1" type="xs:string"/>

```

```

        <xs:element name=" myElem2" type="xs:string"/>
        <xs:element name=" myElem3" type="xs:string"/>
    </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

```

The element is then defined as:

```

<xs:element name="MyClssGroup" type="MyClssGroupType"
substitutionGroup="agsml:_SampFeature"/>

```

An instance document would look like:

```

<laboratoryTesting xmlns:myDict="http://MyWebsite/MyAGSMLTypes">
  <Clss>
    <clssNmc>0</ clssNmc >
    <clssLi>0.5</ clssLi >
    <myDict:elemOne>My first userdefined element in the laboratoryTesting group</myDict:elemOne>
    <myDict:elemTwo>1234</myDict:elemTwo>
  </Clss>
  <myDict:MyGroup>
    <myDict:myElem1>This is a user define group under the laboratoryTesting element</myDict:myElem1>
    <myDict:myElem2>These are elements allowed within the group </myDict:myElem2>
    <myDict:myElem3>Only these elements are allowed</myDict:myElem3>
  </myDict:MyGroup>
</holes>

```

## B.6 DICT Group

Below is a fragment of an AGSML file which adds headers to the Proj element and also adds a group:

```

<myDict:Proj>
  <ProjId>
    .....
    <!--Below are custom headers-->
    <myDict:elemOne> .... </myDict:elemOne>
    <myDict:elemTwo> .... </ myDict:elemTwo>
    <!--Below is a custom group-->
    <myDict:MyGroup>
      <myDict:myElem1> .....</myDict:myElem1>
      <myDict:myElem2> .....</myDict:myElem2>
      <myDict:myElem3> .....</myDict:myElem3>
    </myDict:MyGroup>
  </myDict:Proj>

```

In the example above, the "myDict" namespace is just an identifier to distinguish AGSML elements from the user defined elements. Any string value could be used for this, which is declared in the root element with other namespace declarations. This is shown below:

```

<SiteInvestigation xmlns:myDict=http://www.somewhere.com/myDict
  xmlns=http://www.ags.org.uk/agsml
  xmlns:gml=http://www.opengis.net/gml>

```

Users are able to add their own groups and headers by writing their own schema. The custom schema imports the AGSML schema. To add headers to an existing group, the schema must extend the abstract type defined in the AGSML schema, and add the elements needed by the user. To add a group, a new complex type can be

defined, and its element defined in the substitution group of its parent.

For example, if a group is to be added under the <Samp> element, it must be defined in the <\_samps> substitution group.

## B.7 ABBR and CODE Groups

The Abbr and Code tables are stored in separate files. Each file is referenced by an attribute in the root element. An example Code file is shown below:

```
<Codes xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="code.xsd">
  <Code>
    <CodeCode>MN</CodeCode>
    <CodeDesc>Manganese</CodeDesc>
  </Code>
  <Code>
    <CodeCode>ALCO</CodeCode>
    <CodeDesc>Alcohols</CodeDesc>
  </Code>
  <Code>
    <CodeCode>MOILS</CodeCode>
    <CodeDesc>Mineral Oils</CodeDesc>
  </Code>
</Codes>
```

The schema for this is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="Codes" type="CodesType"/>
  <xs:complexType name="CodeType">
    <xs:sequence>
      <xs:element name="CodeCode"/>
      <xs:element name="CodeDesc"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="CodesType">
    <xs:sequence>
      <xs:element name="Code" type="CodeType" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

The standard Code list will be available via the AGS website at <http://www.ags.org.uk/agsm/v1/codes.xml>. Users will specify that they are using the standard list by adding the attribute `codePickList="http://www.ags.org.uk/agsm/v1/codes.xml"` to the root element.

If a user wants to use their own pick list codes, they can download the standard file and edit the codes. They can then publish their own Code files on the Internet.

An example Abbr file is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Abbrs xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Abbr.xsd">
  <AbbrHdng hdng="InstType">
    <Abbr>
      <AbbrCode>IPCP</AbbrCode>
      <AbbrDesc>Interface pressure cell - pneumatic</AbbrDesc>
    </Abbr>
    <Abbr>
      <AbbrCode>IPCH</AbbrCode>
      <AbbrDesc>Interface pressure cell – hydraulic</AbbrDesc>
    </Abbr>
  </AbbrHdng>
</Abbrs>
```

```

</AbbrHdng>
<AbbrHdng hdng="IsptType">
  <Abbr>
    <AbbrCode>C</AbbrCode>
    <AbbrDesc>Cone</AbbrDesc>
  </Abbr>
  <Abbr>
    <AbbrCode>S</AbbrCode>
    <AbbrDesc>Split Spoon</AbbrDesc>
  </Abbr>
</AbbrHdng>
</Abbrs>

```

The <AbbrHdng> element has an attribute "hdng" specifying the group the abbreviations belong to. The schema for the Abbr file is shown below:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">
  <xs:element name="Abbrs">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="AbbrHdng"
          type="AbbrHdngType"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:complexType name="AbbrHdngType">
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="Abbr">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="AbbrCode"/>
            <xs:element name="AbbrDesc"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="hdng" type="xs:string" use="required"/>
  </xs:complexType>
</xs:schema>

```

The standard Abbr list will be available via the AGS website. Users will specify that they are using the standard list by adding the attribute abbrPickList="http://www.ags.org.uk/agsml/v1/abbrs.xml" to the root element.

If software needs to use their own Abbr pick list codes, it can download the standard file and edit the codes. They can then publish their own Abbr files on the Internet.

Storing the picklists in separate files means a schema will not be able to check that codes used in an AGSML are valid codes. However, specialist software or web services will be able to check that AGSML files have valid pick list codes.

### B.8 UNITS Group

Units will be implemented as an attribute (e.g. <geolTop uom="m">) that points to a separate *Units* dictionary. AGS will provide a standard *Units* dictionary on the website which will be referenced by URI as www.ags.org.uk/agsml/v1/units.xml. If non-standard units are adopted than an alternative dictionary would be provided with a different URI. If the standard AGS dictionary was referenced then the URI could be neglected i.e. <geolTop uom="m">.

Unit definitions would only need to be supplied once and could then apply to all enclosed tags, until the units were redefined. Here is an example of using units:

```
<sampTop uom="m">1.00</sampTop>
```

The next <sampTop> element will not have to specify the UoM attribute again, as all <sampTop> elements will now have 'm' as it units. If however, the units for sampTop should change half way though an AGSML file, the

attribute can be used again to specify a different unit, for example:

```
<sampTop uom="ft">4.00</sampTop>.
```

The unit 'ft' will now be used for all following <sampTop> elements.

User defined dictionaries must conform to the *Unit* schema, unit.xsd, provided on the AGSML website. The user defined dictionary can then be referenced from the AGSML file using an attribute on the element, for example:

```
<sampTop uom="http://www.myDomain.com/agsml/unit.xml#m">
```

## B.9 FILE Group

Below is an example xml file and schema showing how xlink can be used to attach multiple files to an element. It requires adding a File element for each file that is associated with an AGSML file.

attribute name	attribute value	Example
xlink:type	"locator"	"locator"
xlink:href	<i>Filename</i>	"Photo1.jpg"
xlink:label	<i>Label to refer to file</i>	"Photo"
xlink:title	<i>Description of File</i>	"Photo of Borehole"
ags:openWith	<i>Program to open file</i>	"PaintShop Pro"
ags:date	<i>Date</i>	"2004-11-24"
ags:type	<i>File Type</i>	"jpg"

```
<!--Project is root element-->
<Project xmlns:xlink=http://www.w3.org/1999/xlink
  xmlns:ags="http://www.ags.org.uk/ags">
  <Hole>
    <HoleId>TP501</HoleId>
    <HoleType>TP</HoleType>
    .....
    <files>
      <File xlink:type="locator" xlink:href="Photo.jpg"
        xlink:label="Photo1"
        xlink:title="Photo of Borehole TP501">
        <openWith>"PaintShop Pro"</openWith>
        <date>2004-11-24</date>
        <type>jpg</type>
      </File>
    </files>
  </Hole>
</Proj>
```

Using xlink in this way allows multiple files to be attached to an element, and information about the attached file is also stored.

The schema for the AGSML files will include the xlink schema:

```
<xs:import namespace=http://www.w3.org/1999/xlink
  schemaLocation="xlink.xsd"/>
```

All the attributes can be declared in complex type:

```
<complexType name="FileType">
  <annotation>
    <documentation>This type is used by all types needing to attach files</documentation>
  </annotation>
  <sequence>
    <element name="openWith" type="string"/>
  </sequence>
</complexType>
```

```
<element name="date" type="date"/>
  <element name="type" type="string"/>
</sequence>
<attribute ref="xlink:href"/>
<attribute ref="xlink:label"/>
<attribute ref="xlink:title"/>
</complexType>
```

This type can be used within a files element to attach multiple files:

```
<element name="files">
  <complexType>
    <sequence>
      <element name="File" type="agsml:FileType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

These can then be used on each element that can have a file attached:

```
<xsd:complexType name="HoleType">
  <xsd:sequence>
    <xsd:element name="HoleId" type="xsd:string"/>
    <xsd:element name="HoleType" type="xsd:string"/>
    .....etc...
    <xsd:element ref="agsml:files"/>
  </xsd:sequence>
</xsd:complexType>
```

## Appendix C AGSML Data Structure

```

<SiteInvestigation>
  <projects>
    <Proj>
      <holes>
        <Hole>
          <holeInformation>
            <Bkfl/>
            <Cdia/>
            <Chis/>
            <Drem/>
            <Flsh/>
            <Hdia/>
            <Hdph/>
            <Ptim/>
            <Samp>
              <laboratoryTesting>
                <Cbrg>
                  <cbrResults>
                    <Cbrt/>
                  </cbrResults>
                </Cbrg>
                <Chlk/>
                <Clss/>
                <Cmpg>
                  <comResults>
                    <Cmpt/>
                  </comResults>
                </Cmpg>
                <Cnmt/>
                <Cong>
                  <conResults>
                    <Cons/>
                  </conResults>
                </Cong>
                <Frst/>
                <Grad/>
                <Mcvg>
                  <mcvResults>
                    <Mcvt/>
                  </mcvResults>
                </Mcvg>
                <Ptst/>
                <Reld/>
                <Rock/>
                <Shbg>
                  <shbResults>
                    <Shbt/>
                  </shbResults>
                </Shbg>
                <Suct/>
                <Tnpc/>
                <Trig>
                  <triResults>
                    <Trix/>
                  </triResults>
                </Trig>
              </labTesting>
            </Samp>
          </holeInformation>
        <monitoring>
          <Hpgi>
            <hpgResults>
              <Hpgo/>
            </hpgResults>
          </Hpgi>
          <Inst>
            <insResults>

```

```

        <lobs/>
      </insResults>
    </Inst>
  <Monp>
    <monResults>
      <lcct/>
      <Monr/>
    </monResults>
  </Monp>
  <Pref>
    <preResults>
      <Pobs/>
    </preResults>
  </Pref>
  <Prof>
    <proResults>
      <Prob>
    </proResults>
  </Prof>
  </Trem>
</monitoring>
<fieldTesting>
  <Dprg>
    <dprResults>
      <Dprb/>
    </dprResults>
  </Dprg>
  <lcbr/>
  <ldcn/>
  <lfid/>
  <lpid/>
  <lprm/>
  <lrdx/>
  <lres/>
  <lvan/>
  <lspt/>
  <Prtg>
    <prtResults>
      <Prtb/>
      <Prtl/>
    </prtResults>
  </Prtg>
  <Pump/>
  <Stcn/>
</fieldTesting>
<groundInformation>
  <Core/>
  <Detl/>
  <Disc/>
  <Geol/>
  <Frac/>
  <Weth/>
  <Wstk/>
</groundInformation>
</Hole>
</holes>
</Proj>
</projects>
</SiteInvestigation>

```