homecare.co.uk XML Job Feed Schema

Table of Contents

- Schema Document Properties
- Global Declarations
    - Element: **feed**
- Global Definitions
    - Complex Type: **category_type**
    - Complex Type: **contact_coms**
    - Complex Type: **contact_type**
    - Complex Type: **JobElement**
    - Complex Type: **location_type**
    - Complex Type: **pay_descriptor**
    - Complex Type: **pay_type**
    - Complex Type: **setting_type**
    - Complex Type: **terms_type**
    - Complex Type: **type_type**
    - Simple Type: **integer_id**
    - Simple Type: **longString**
    - Simple Type: **money**
    - Simple Type: **pay_type_id_enum**
    - Simple Type: **pay_type_name_enum**
    - Simple Type: **token_emailaddress**
    - Simple Type: **token_postcode**
    - Simple Type: **token_url**
    - Simple Type: **token255**
    - Simple Type: **token60**
    - Simple Type: **category_id_enum**
    - Simple Type: **category_name_enum**
    - Simple Type: **setting_id_enum**
    - Simple Type: **setting_name_enum**
    - Simple Type: **type_id_enum**
    - Simple Type: **type_name_enum**
- Glossary

top

Schema Document Properties

Target Namespace	None
Element and Attribute Namespaces	• Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations have no namespace. • By default, local attribute declarations have no namespace.

Schema Composition	• This schema includes components from the following schema document(s):
	  ◦ https://api.homecare.co.uk/feed/job/xml/3/category.xsd
	  ◦ https://api.homecare.co.uk/feed/job/xml/3/setting.xsd
	  ◦ https://api.homecare.co.uk/feed/job/xml/3/type.xsd

Declared Namespaces

Prefix	Namespace
xml	http://www.w3.org/XML/1998/namespace
xs	http://www.w3.org/2001/XMLSchema

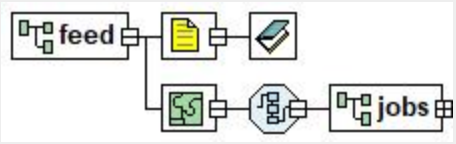
Schema Component Representation

```
<xs:schema>
   <xs:include
   schemaLocation="https://api.homecare.co.uk/feed/job/xml/3/category.xsd"/>
   <xs:include
   schemaLocation="https://api.homecare.co.uk/feed/job/xml/3/setting.xsd"/>
   <xs:include
   schemaLocation="https://api.homecare.co.uk/feed/job/xml/3/type.xsd"/>
   ...
</xs:schema>
```

top

Global Declarations

Element: feed

Name	feed
Type	Locally-defined complex type
Nillable	no
Abstract	no
Documentation	A list of open and active jobs
Diagram	

XML Instance Representation

```
<feed>
   <jobs> [1]
      <!--
      Uniqueness Constraint - reference_unique
      Selector - job/reference
      Field(s) - .
      -->

      <job> JobElement </job> [0..*]
   </jobs>
</feed>
```

Schema Component Representation

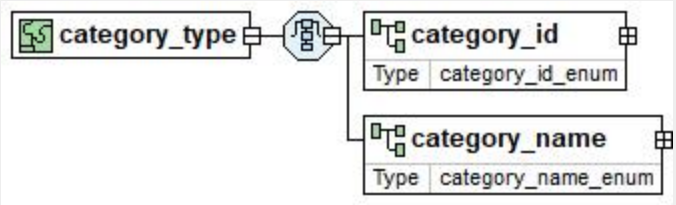
```
<xs:element name="feed">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="jobs">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="job" type="JobElement" minOccurs="0"
                  maxOccurs="unbounded"/>
               </xs:sequence>
            </xs:complexType>
            <xs:unique name="reference_unique">
               <xs:selector xpath="job/reference"/>
               <xs:field xpath="."/>
            </xs:unique>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

top

Global Definitions

Complex Type: category_type

Super-types:	None
Sub-types:	None

Name	category_type
Abstract	no
Documentation	Primary Job Category/role. supply either category_id or category_name
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <category_id> category_id_enum </category_id> [1]
      <category_name> category_name_enum </category_name> [1]
   End Choice
</...>
```

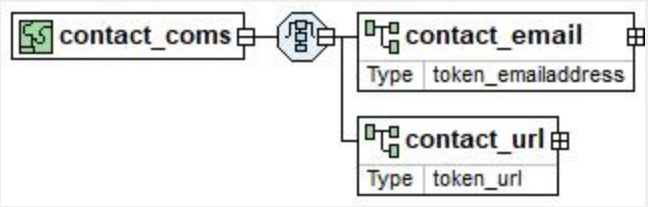
Schema Component Representation

```
<xs:complexType name="category_type">
   <xs:choice>
      <xs:element name="category_id" type="category_id_enum"/>
      <xs:element name="category_name" type="category_name_enum"/>
   </xs:choice>
</xs:complexType>
```

top

Complex Type: contact_coms

Super-types:	None
Sub-types:	None

Name	contact_coms
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <contact_email> token_emailaddress </contact_email> [1]
      <contact_url> token_url </contact_url> [1]
   End Choice
</...>
```

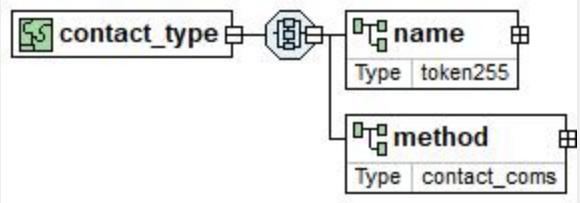
Schema Component Representation

```
<xs:complexType name="contact_coms">
   <xs:choice>
      <xs:element name="contact_email" type="token_emailaddress"/>
      <xs:element name="contact_url" type="token_url"/>
   </xs:choice>
</xs:complexType>
```

top

Complex Type: contact_type

Super-types:	None
Sub-types:	None

Name	contact_type
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start All [1]
      <name> token255 </name> [1]
      <method> contact_coms </method> [1]
   End All
</...>
```

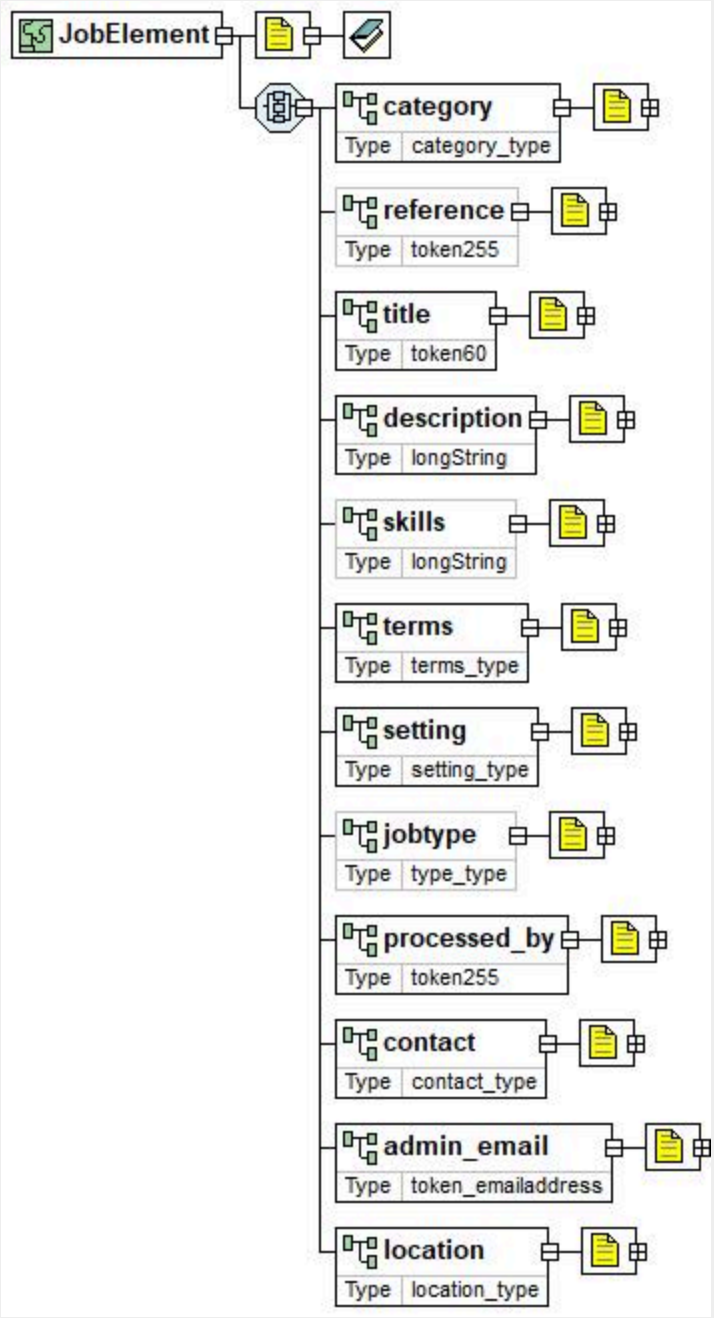
Schema Component Representation

```
<xs:complexType name="contact_type">
   <xs:all>
      <xs:element name="name" type="token255"/>
      <xs:element name="method" type="contact_coms"/>
   </xs:all>
</xs:complexType>
```

top

Complex Type: JobElement

Super-types:	None
Sub-types:	None

Name	JobElement
Abstract	no
Documentation	Definition of an individual Job Role
Diagram	

XML Instance Representation

```
<...>
   Start All [1]
      <category> category_type </category> [1]
      <!-- Primary Job Category/role. supply either category_id or category_name
      -->
      <reference> token255 </reference> [0..1]
      <!-- A unique reference for the Job -->
      <title> token60 </title> [1]
      <!-- Job title e.g. Head of Technical Support -->
      <description> longString </description> [1]
      <!-- Description of your job. HTML filter in place as follows
      p,i,b,u,strong,em,ol,ul,li,br allowed but attributes stripped (exception
      align on p tags) small,big,center,font,div,span will be removed but the
      content will stay e.g. \ -->
      <skills> longString </skills> [0..1]
      <!-- Comma separated list of Skills/Qualifications -->
      <terms> terms_type </terms> [1]
      <!-- should contain salary a free text description e.g. Negotiable or a
      pay descriptor -->
      <setting> setting_type </setting> [1]
      <!-- a setting_id or setting_name that describes where the job is to be
      undertaken -->
      <jobtype> type_type </jobtype> [0..1]
      <!-- a type_id or type_name describing the type of job e.g. Full Time or
      Part Time -->
      <processed_by> token255 </processed_by> [1]
      <!-- The name of the person entering the job - for logging purposes (Not
      displayed to user) -->
      <contact> contact_type </contact> [1]
      <!-- the name is Displayed with job details, email is not displayed but
      applications are sent to this email address, url on a click of \ -->
      <admin_email> token_emailaddress </admin_email> [1]
      <!-- A reminder will be sent to this email address before the job expires,
      to alert them if they wish to extend the job. (Email Not sent if the job
      is posted by a Partner ) -->
      <location> location_type </location> [1]
      <!-- The location of the job either the memberId, postcode or both must be
      provided if memberId or memberId and matching postcode then the members
      location is used if non matching postcode the postcode is used as the job
      location. -->
   End All
</...>
```

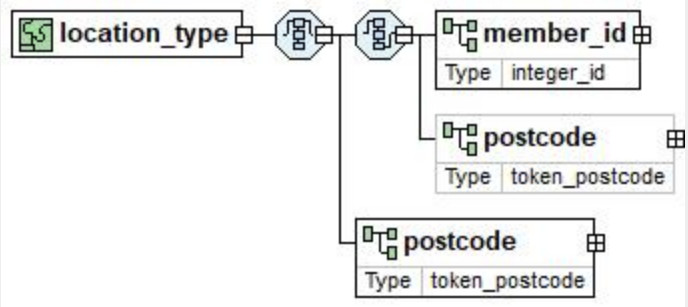
Schema Component Representation

```
<xs:complexType name="JobElement">
  <xs:all>
    <xs:element name="category" type="category_type"/>
    <xs:element name="reference" type="token255" minOccurs="0" maxOccurs="1"/>
    <xs:element name="title" type="token60"/>
    <xs:element name="description" type="longString"/>
    <xs:element name="skills" type="longString" minOccurs="0" maxOccurs="1"/>
    <xs:element name="terms" type="terms_type"/>
    <xs:element name="setting" type="setting_type"/>
    <xs:element name="jobtype" type="type_type" minOccurs="0" maxOccurs="1"/>
    <xs:element name="processed_by" type="token255"/>
    <xs:element name="contact" type="contact_type"/>
    <xs:element name="admin_email" type="token_emailaddress"/>
    <xs:element name="location" type="location_type"/>
  </xs:all>
</xs:complexType>
```

top

Complex Type: location_type

Super-types:	None
Sub-types:	None

Name	location_type
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <member_id> integer_id </member_id> [1]
      <postcode> token_postcode </postcode> [0..1]
      <postcode> token_postcode </postcode> [1]
   End Choice
</...>
```

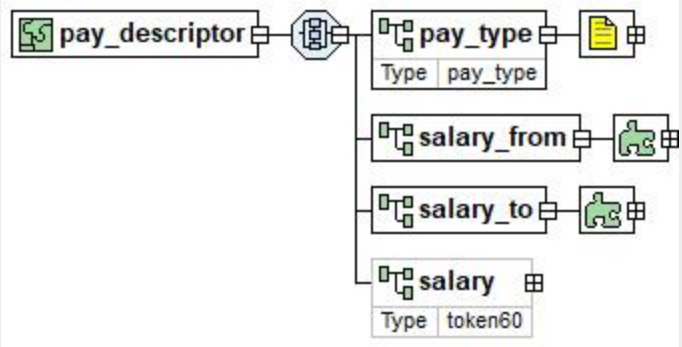
Schema Component Representation

```
<xs:complexType name="location_type">
   <xs:choice>
      <xs:sequence>
         <xs:element name="member_id" type="integer_id"/>
         <xs:element name="postcode" type="token_postcode" minOccurs="0"/>
      </xs:sequence>
      <xs:element name="postcode" type="token_postcode"/>
   </xs:choice>
</xs:complexType>
```

top

Complex Type: pay_descriptor

Super-types:	None
Sub-types:	None

Name	pay_descriptor
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start All [1]
      <pay_type> pay_type </pay_type> [1]
      <!-- either a pay_type_id or pay_type_name -->
      <salary_from> money (value > 0) </salary_from> [1]
      <salary_to> money (value > 0) </salary_to> [1]
      <salary> token60 </salary> [0..1]
   End All
</...>
```

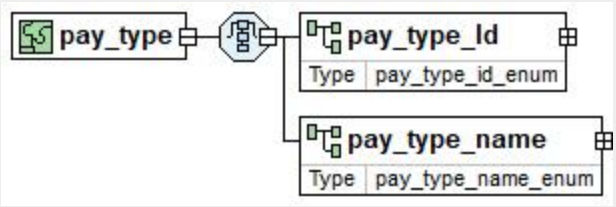
Schema Component Representation

```
<xs:complexType name="pay_descriptor">
   <xs:all>
      <xs:element name="pay_type" type="pay_type"/>
      <xs:element name="salary_from">
         <xs:simpleType>
            <xs:restriction base="money">
               <xs:minExclusive value="0"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:element>
      <xs:element name="salary_to">
         <xs:simpleType>
            <xs:restriction base="money">
               <xs:minExclusive value="0"/>
            </xs:restriction>
         </xs:simpleType>
      </xs:element>
      <xs:element name="salary" type="token60" minOccurs="0" maxOccurs="1"/>
   </xs:all>
</xs:complexType>
```

top

Complex Type: pay_type

Super-types:	None
Sub-types:	None

Name	pay_type
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <pay_type_Id> pay_type_id_enum </pay_type_Id> [1]
      <pay_type_name> pay_type_name_enum </pay_type_name> [1]
   End Choice
</...>
```

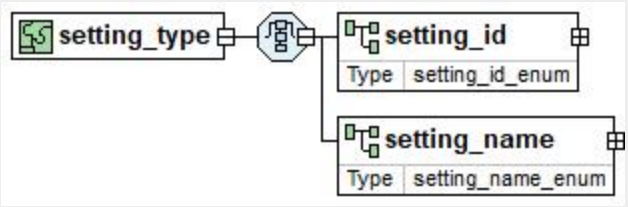
Schema Component Representation

```
<xs:complexType name="pay_type">
   <xs:choice>
      <xs:element name="pay_type_Id" type="pay_type_id_enum"/>
      <xs:element name="pay_type_name" type="pay_type_name_enum"/>
   </xs:choice>
</xs:complexType>
```

top

Complex Type: setting_type

Super-types:	None
Sub-types:	None

Name	setting_type
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <setting_id> setting_id_enum </setting_id> [1]
      <setting_name> setting_name_enum </setting_name> [1]
   End Choice
</...>
```

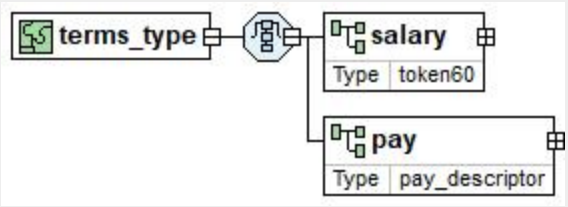
Schema Component Representation

```
<xs:complexType name="setting_type">
   <xs:choice>
      <xs:element name="setting_id" type="setting_id_enum"/>
      <xs:element name="setting_name" type="setting_name_enum"/>
   </xs:choice>
</xs:complexType>
```

top

Complex Type: terms_type

Super-types:	None
Sub-types:	None

Name	terms_type
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <salary> token60 </salary> [1]
      <pay> pay_descriptor </pay> [1]
   End Choice
</...>
```

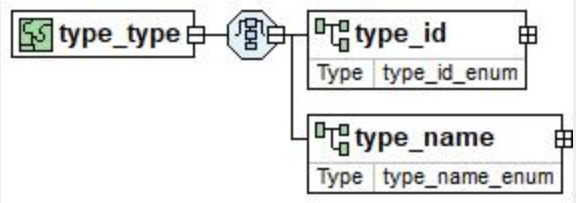
Schema Component Representation

```
<xs:complexType name="terms_type">
   <xs:choice>
      <xs:element name="salary" type="token60"/>
      <xs:element name="pay" type="pay_descriptor"/>
   </xs:choice>
</xs:complexType>
```

top

Complex Type: type_type

Super-types:	None
Sub-types:	None

Name	type_type
Abstract	no
Diagram	

XML Instance Representation

```
<...>
   Start Choice [1]
      <type_id> type_id_enum </type_id> [1]
      <type_name> type_name_enum </type_name> [1]
   End Choice
</...>
```

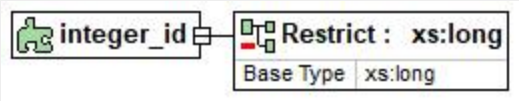
Schema Component Representation

```
<xs:complexType name="type_type">
   <xs:choice>
      <xs:element name="type_id" type="type_id_enum"/>
      <xs:element name="type_name" type="type_name_enum"/>
   </xs:choice>
</xs:complexType>
```

top

Simple Type: integer_id

Super-types:	xs:long < **integer_id** (by restriction)
Sub-types:	None

Name	integer_id
Content	• Base XSD Type: long • *value* > 0
Diagram	

Schema Component Representation

```
<xs:simpleType name="integer_id">
   <xs:restriction base="xs:long">
      <xs:minExclusive value="0"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: longString

Super-types:	xs:string < **longString** (by restriction)
Sub-types:	None

Name	longString
Content	• Base XSD Type: string • *length* >= 0
Diagram	

Schema Component Representation

```
<xs:simpleType name="longString">
   <xs:restriction base="xs:string">
      <xs:minLength value="0"/>
      <xs:maxLength value="8000"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: money

Super-types:	xs:decimal < **money** (by restriction)
Sub-types:	None

Name	money
Content	• Base XSD Type: decimal • *no. of fraction digits* = 2
Diagram	

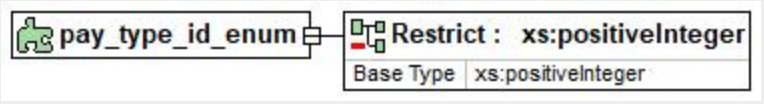
Schema Component Representation

```
<xs:simpleType name="money">
   <xs:restriction base="xs:decimal">
      <xs:fractionDigits value="2"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: pay_type_id_enum

Super-types:	xs:positiveInteger < **pay_type_id_enum** (by restriction)
Sub-types:	None

Name	pay_type_id_enum
Content	• Base XSD Type: positiveInteger • *value* comes from list: {'1'|'2'}
Diagram	

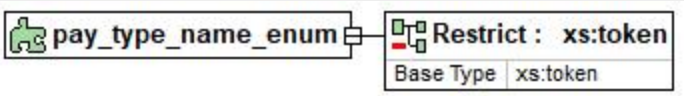
Schema Component Representation

```
<xs:simpleType name="pay_type_id_enum">
   <xs:restriction base="xs:positiveInteger">
      <xs:enumeration value="1"/>
      <xs:enumeration value="2"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: pay_type_name_enum

Super-types:	xs:token < **pay_type_name_enum** (by restriction)
Sub-types:	None

Name	pay_type_name_enum
Content	• Base XSD Type: token • *value* comes from list: {'per annum'|'per hour'}
Diagram	

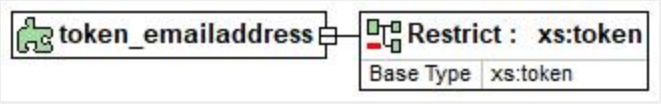
Schema Component Representation

```
<xs:simpleType name="pay_type_name_enum">
   <xs:restriction base="xs:token">
      <xs:enumeration value="per annum"/>
      <xs:enumeration value="per hour"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: token_emailaddress

Super-types:	xs:token < **token_emailaddress** (by restriction)
Sub-types:	None

Name	token_emailaddress
Content	• Base XSD Type: token • *pattern* = ([a-z0-9!#$%&'*+/=?^_`{\|}~\-]+(\.[a-z0-9!#$%&'*+/=?^_`{|}~\-]+)*|"([!#-Z\[\\\]^_a-~]|\\[!-~])*")@(([a-z0-9]([a-z0-9\-]*[a-z0-9])?\.)+[a-z0-9]([a-z0-9\-]*[a-z0-9])?|\[((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9\-]*[a-z0-9]:([!#-Z !#\-\[\]-~]|\\[!-~\-~]|\\[!-~])+)\]) • *length* <= 500
Diagram	

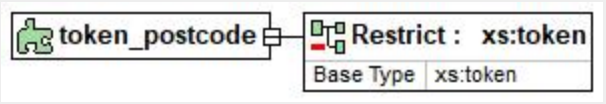
Schema Component Representation

```
<xs:simpleType name="token_emailaddress">
  <xs:restriction base="xs:token">
    <xs:maxLength value="500"/>
    <xs:pattern value="([a-z0-9!#$%&'*+/=?^_`{|}~\-]+(\.[a-z0-9!#$%&'*+/=?
    ^_`{|}~\-]+)*|"([ !#-Z\[\\\]^_a-~]|\\[!-~])*")@(([a-z0-9]([a-z0-9\-]*[a-
    z0-9])?\.)+[a-z0-9]([a-z0-9\-]*[a-z0-9])?|\[((25[0-5]|2[0-4][0-9]|[01]?[0-
    9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?|[a-z0-9\-]*[a-z0-9]:
    ([ !#-Z !#\-\[\]-~]|\\[!-~\-~]|\\[!-~])+)\])"/>
  </xs:restriction>
</xs:simpleType>
```

top

Simple Type: token_postcode

Super-types:	xs:token < **token_postcode** (by restriction)
Sub-types:	None

Name	token_postcode
Content	• Base XSD Type: token • *pattern* = (([A-Z][0-9]{1,2})|([A-Z][A-HJ-Y]?[0-9][A-HJ-Y])|([A-Z][A-HJ-Y][0-9]{1,2}))\s?[0-9][A-Z]{2} • *length* <= 10
Diagram	

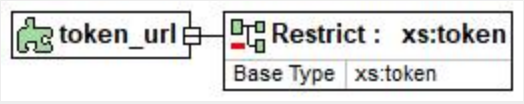
Schema Component Representation

```
<xs:simpleType name="token_postcode">
  <xs:restriction base="xs:token">
    <xs:maxLength value="10"/>
    <xs:pattern value="(([A-Z][0-9]{1,2})|([A-Z][A-HJ-Y]?[0-9][A-HJ-Y])|([A-Z]
    [A-HJ-Y][0-9]{1,2}))\s?[0-9][A-Z]{2}"/>
  </xs:restriction>
</xs:simpleType>
```

top

Simple Type: token_url

Super-types:	xs:token < **token_url** (by restriction)
Sub-types:	None

Name	token_url
Content	• Base XSD Type: token • *length* <= 1000
Diagram	

Schema Component Representation

```
<xs:simpleType name="token_url">
   <xs:restriction base="xs:token">
      <xs:maxLength value="1000"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: token255

Super-types:	xs:token < **token255** (by restriction)
Sub-types:	None

Name	token255
Content	• Base XSD Type: token • *length* >= 0
Diagram	

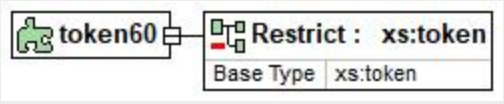
Schema Component Representation

```
<xs:simpleType name="token255">
   <xs:restriction base="xs:token">
      <xs:minLength value="0"/>
      <xs:maxLength value="255"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: token60

Super-types:	xs:token < **token60** (by restriction)
Sub-types:	None

Name	token60
Content	• Base XSD Type: token • *length* >= 0
Diagram	

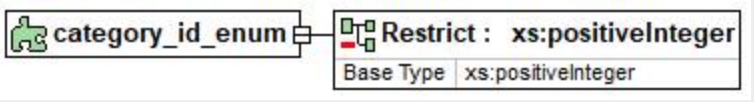
Schema Component Representation

```
<xs:simpleType name="token60">
   <xs:restriction base="xs:token">
      <xs:minLength value="0"/>
      <xs:maxLength value="255"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: category_id_enum

Super-types:	xs:positiveInteger < **category_id_enum** (by restriction)
Sub-types:	None

Name	category_id_enum
Content	• Base XSD Type: positiveInteger • *value* comes from list: {'22'|'47'|'48'|'24'|'49'|'23'|'29'|'75'|'67'|'76'|'26'|'63'|'50'|'51'|'52'|'27'}
Diagram	

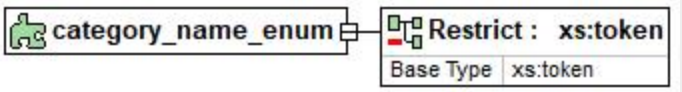
Schema Component Representation

```
<xs:simpleType name="category_id_enum">
  <xs:restriction base="xs:positiveInteger">
    <xs:enumeration value="22"/> <!-- Registered Manager / Service Manager -->
    <xs:enumeration value="47"/> <!-- Area / Regional Manager -->
    <xs:enumeration value="48"/> <!-- Clinical Lead Nurse -->
    <xs:enumeration value="24"/> <!-- Nurse -->
    <xs:enumeration value="49"/> <!-- Team Leader / Senior Carer -->
    <xs:enumeration value="23"/> <!-- Carer / Support Worker -->
    <xs:enumeration value="29"/> <!-- Live-in Carer -->
    <xs:enumeration value="75"/> <!-- Befriender / Companion -->
    <xs:enumeration value="67"/> <!-- Social Worker -->
    <xs:enumeration value="76"/> <!-- Training & Development -->
    <xs:enumeration value="26"/> <!-- Care Coordinator / Administrator -->
    <xs:enumeration value="63"/> <!-- Director / Divisional Director -->
    <xs:enumeration value="50"/> <!-- Finance / Accountant / Bookkeeper -->
    <xs:enumeration value="51"/> <!-- Human Resources / HR / Recruitment -->
    <xs:enumeration value="52"/> <!-- Marketing -->
    <xs:enumeration value="27"/> <!-- Other -->
  </xs:restriction>
</xs:simpleType>
```

top

Simple Type: category_name_enum

Super-types:	xs:token < **category_name_enum** (by restriction)
Sub-types:	None

Name	category_name_enum
Content	• Base XSD Type: token • *value* comes from list: {'Registered Manager / Service Manager'|'Area / Regional Manager'|'Clinical Lead Nurse'|'Nurse'|'Team Leader / Senior Carer'|'Carer / Support Worker'|'Live-in Carer'|'Befriender / Companion'|'Social Worker'|'Training & Development'|'Care Coordinator / Administrator'|'Director / Divisional Director'|'Finance / Accountant / Bookkeeper'|'Human Resources / HR / Recruitment'|'Marketing'|'Other'}
Diagram	

Schema Component Representation

```
<xs:simpleType name="category_name_enum">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Registered Manager / Service Manager"/>
    <xs:enumeration value="Area / Regional Manager"/>
    <xs:enumeration value="Clinical Lead Nurse"/>
    <xs:enumeration value="Nurse"/>
    <xs:enumeration value="Team Leader / Senior Carer"/>
    <xs:enumeration value="Carer / Support Worker"/>
    <xs:enumeration value="Live-in Carer"/>
    <xs:enumeration value="Befriender / Companion"/>
    <xs:enumeration value="Social Worker"/>
    <xs:enumeration value="Training & Development"/>
    <xs:enumeration value="Care Coordinator / Administrator"/>
    <xs:enumeration value="Director / Divisional Director"/>
    <xs:enumeration value="Finance / Accountant / Bookkeeper"/>
    <xs:enumeration value="Human Resources / HR / Recruitment"/>
    <xs:enumeration value="Marketing"/>
    <xs:enumeration value="Other"/>
  </xs:restriction>
</xs:simpleType>
```

top

Simple Type: setting_id_enum

Super-types:	xs:positiveInteger < **setting_id_enum** (by restriction)
Sub-types:	None

Name	setting_id_enum
Content	• Base XSD Type: positiveInteger • *value* comes from list: {'32'|'31'}
Diagram	

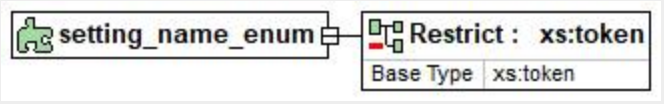
Schema Component Representation

```
<xs:simpleType name="setting_id_enum">
   <xs:restriction base="xs:positiveInteger">
      <xs:enumeration value="32"/> <!-- Clients Home -->
      <xs:enumeration value="31"/> <!-- Agency Office -->
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: setting_name_enum

Super-types:	xs:token < **setting_name_enum** (by restriction)
Sub-types:	None

Name	setting_name_enum
Content	• Base XSD Type: token • *value* comes from list: {'Clients Home'|'Agency Office'}
Diagram	

Schema Component Representation

```
<xs:simpleType name="setting_name_enum">
   <xs:restriction base="xs:token">
      <xs:enumeration value="Clients Home"/>
      <xs:enumeration value="Agency Office"/>
   </xs:restriction>
</xs:simpleType>
```

top

Simple Type: type_id_enum

Super-types:	xs:positiveInteger < **type_id_enum** (by restriction)
Sub-types:	None

Name	type_id_enum
Content	• Base XSD Type: positiveInteger • *value* comes from list: {'10'|'11'|'12'|'13'|'14'|'15'|'16'}
Diagram	

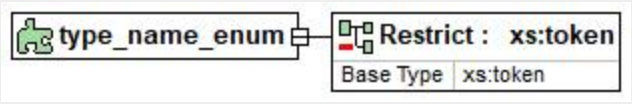
Schema Component Representation

```
<xs:simpleType name="type_id_enum">
  <xs:restriction base="xs:positiveInteger">
    <xs:enumeration value="10"/> <!-- Full Time -->
    <xs:enumeration value="11"/> <!-- Part Time -->
    <xs:enumeration value="12"/> <!-- Full Time or Part Time -->
    <xs:enumeration value="13"/> <!-- Temporary / Cover -->
    <xs:enumeration value="14"/> <!-- Contract -->
    <xs:enumeration value="15"/> <!-- Voluntary -->
    <xs:enumeration value="16"/> <!-- Apprenticeship -->
  </xs:restriction>
</xs:simpleType>
```

top

Simple Type: type_name_enum

Super-types:	xs:token < **type_name_enum** (by restriction)
Sub-types:	None

Name	type_name_enum
Content	• Base XSD Type: token • *value* comes from list: {'Full Time'|'Part Time'|'Full Time or Part Time'|'Temporary / Cover'|'Contract'|'Voluntary'|'Apprenticeship'}
Diagram	

Schema Component Representation

```
<xs:simpleType name="type_name_enum">
  <xs:restriction base="xs:token">
    <xs:enumeration value="Full Time"/>
    <xs:enumeration value="Part Time"/>
    <xs:enumeration value="Full Time or Part Time"/>
    <xs:enumeration value="Temporary / Cover"/>
    <xs:enumeration value="Contract"/>
    <xs:enumeration value="Voluntary"/>
    <xs:enumeration value="Apprenticeship"/>
  </xs:restriction>
</xs:simpleType>
```

top

Glossary

Abstract (Applies to complex type definitions and element declarations). An abstract element or complex type cannot used to validate an element instance. If there is a reference to an abstract element, only element declarations that can substitute the abstract element can be used to validate the instance. For references to abstract type definitions, only derived types can be used.

All Model Group Child elements can be provided *in any order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-all.

Choice Model Group *Only one* from the list of child elements and model groups can be provided in instances. See: http://www.w3.org/TR/xmlschema-1/#element-choice.

Collapse Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32). Then, collapse contiguous sequences of space characters into single space character, and remove leading and trailing space characters.

Disallowed Substitutions (Applies to element declarations). If *substitution* is specified, then substitution group members cannot be used in place of the given element declaration to validate element instances. If *derivation methods*, e.g. extension, restriction, are specified, then the given element declaration will not validate element instances that have types derived from the element declaration's type using the specified derivation methods. Normally, element instances can override their declaration's type by specifying an `xsi:type` attribute.

Key Constraint Like Uniqueness Constraint, but additionally requires that the specified value(s) must be provided. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Key Reference Constraint Ensures that the specified value(s) must match value(s) from a Key Constraint or Uniqueness Constraint. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

Model Group Groups together element content, specifying the order in which the element content can occur and the number of times the group of element content may be repeated. See: http://www.w3.org/TR/xmlschema-1/#Model_Groups.

Nillable (Applies to element declarations). If an element declaration is nillable, instances can use the `xsi:nil` attribute. The `xsi:nil` attribute is the boolean attribute, *nil*, from the *http://www.w3.org/2001/XMLSchema-instance* namespace. If an element instance has an `xsi:nil` attribute set to true, it can be left empty, even though its element declaration may have required content.

Notation A notation is used to identify the format of a piece of data. Values of elements and attributes that are of type, NOTATION, must come from the names of declared notations. See: http://www.w3.org/TR/xmlschema-1/#cNotation_Declarations.

Preserve Whitespace Policy Preserve whitespaces exactly as they appear in instances.

Prohibited Derivations (Applies to type definitions). Derivation methods that cannot be used to create sub-types from a given type definition.

Prohibited Substitutions (Applies to complex type definitions). Prevents sub-types that have been derived using the specified derivation methods from validating element instances in place of the given type definition.

Replace Whitespace Policy Replace tab, line feed, and carriage return characters with space character (Unicode character 32).

Sequence Model Group Child elements and model groups must be provided *in the specified order* in instances. See: http://www.w3.org/TR/xmlschema-1/#element-sequence.

Substitution Group Elements that are *members* of a substitution group can be used wherever the *head* element of the substitution group is referenced.

Substitution Group Exclusions (Applies to element declarations). Prohibits element declarations from nominating themselves as being able to substitute a given element declaration, if they have types that are derived from the original element's type using the specified derivation methods.

Target Namespace The target namespace identifies the namespace that components in this schema belongs to. If no target namespace is provided, then the schema components do not belong to any namespace.

Uniqueness Constraint Ensures uniqueness of an element/attribute value, or a combination of values, within a specified scope. See: http://www.w3.org/TR/xmlschema-1/#cIdentity-constraint_Definitions.

top