Behavioural Atom Protocol

Draft Specification Version 1.0

Monday, 27 June 2015

# Abstract

Coelition is committed to creating an open standard that gives control and transparency to individuals about how their behavioural data is used for their benefit and within the wider society. Established in the UK in 2013, Coelition is a member-led, not-for-profit organisation that supports businesses wanting to measure and understand everyday living in the digital age, whilst avoiding the complexities of big data and concerns about data privacy.

Behavioural Atoms represent distinct human behaviour events. They are designed to have a compromise granularity, so that they are small in terms of data volume but detailed enough to capture a single human behaviour. For example, eating egg based noodles or swimming laps of butterfly. This document describes the Behavioural Atom format and protocol for transmitting Atoms in this format to a Data Engine.

## Copyright Notice

Copyright © 2015, Coelition. All rights reserved.

Members may reproduce this document for their own use. All Coelition documents may be revised from time to time, the particular version and release date should be noted.

Some aspects of the Coelition specifications may be subject to patent rights or other licensing fees. Coelition makes no guaranties as to the existence of such rights nor is Coelition responsible to identify or disclose such rights. Coelition shall have no liability to any party, for failure to identify or disclose any such rights. Coelition shall have no liability to anyone using this document for costs or losses incurred if it is altered or withdrawn after publication.

## Table of Contents

[1 Abstract 1](#_Toc422230141)

[1.1 Copyright Notice 1](#_Toc422230142)

[1.2 Table of Contents 2](#_Toc422230143)

[2 Acknowledgements 2](#_Toc422230144)

[3 Summary 2](#_Toc422230145)

[4 Normative References 3](#_Toc422230146)

[5 Definitions 3](#_Toc422230147)

[6 HTTP Protocol 5](#_Toc422230148)

[6.1 Overview of HTTP Protocols 5](#_Toc422230149)

[6.2 Media Types for Messages 5](#_Toc422230150)

[6.3 Operations 5](#_Toc422230151)

[6.3.1 Data Engine Information Request 5](#_Toc422230152)

[6.3.2 Atom POST 6](#_Toc422230153)

[7 Atom Object Definition (JSON) 8](#_Toc422230154)

[7.1 Header 8](#_Toc422230155)

[7.2 Context 8](#_Toc422230156)

[7.3 When 9](#_Toc422230157)

[7.4 What 9](#_Toc422230158)

[7.5 How 10](#_Toc422230159)

[7.6 Where 10](#_Toc422230160)

[7.7 Who 11](#_Toc422230161)

[7.8 Extension 11](#_Toc422230162)

[8 Atom Format Definition (Compressed Binary) 12](#_Toc422230163)

[9 Security Management for Atom Post 12](#_Toc422230164)

# Acknowledgements

The following, in alphabetical order, have contributed to the creation of this document: Paul Bruton, Tessella; Joss Langford, Activinsights; Matt Reed, Unilever; David Snelling, Fujitsu.

# Summary

This document describes the Behavioural Atom format (in JSON) and the protocol for transmitting Atoms in this format to the Data Engine. A section describing a recommended security protocol is also provided.

# Normative References

Classification of Everyday Living V1.0, Coelition.

Coelition Identity Authority V1.0, Coelition.

IANA HTTP Header Registry, <http://www.iana.org/assignments/message-headers/perm-headers.html>.

IETF RFC2616, R. Fielding et al, Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.

IETF RFC3986, T.Berners-Lee et al, Uniform Resource Identifiers (URI): Generic Syntax, August 1998, <http://www.ietf.org/rfc/rfc3986.txt>.

IETF RFC4627, D. Crockford, The application/json Media Type for JavaScript Object Notation (JSON), July 2006, <http://www.ietf.org/rfc/rfc4627.txt>.

IETF RFC5246, T. Dierks and E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.2, <http://www.ietf.org/rfc/rfc5246.txt>.

ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards, <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

# Definitions

The terms "shall", "shall not," "should", "should not", "may," "need not", "can" and "cannot" in this document are to be interpreted as described [ISO/IEC Directives, Part 2](http://www.iec.ch/members_experts/refdocs/iec/isoiec-dir2%7Bed6.0%7Den.pdf).

**Atom**: A Behavioural Atom is a digital representation of an observable event in an individual’s life. It is a small block of self-describing, micro-structured data. Any type of life event can be coded into a Behavioural Atom using the Classification of Everyday Living (COEL), a hierarchical taxonomy of decreasing granularity. The individual’s identity is pseudo-anonymised with the directly identifying personal information (DIPI) segregated from the Behavioural Atoms in both storage and transmission. The Behavioural Atoms also code the time and duration of events, how they were observed and where they occurred. The Atom types, or the ‘What’ are described in the “COEL V01”. In this specification JSON encoding for Atoms are defined along with a compressed binary form.

**Data Engine**: Entity responsible for receiving, storing and processing Behavioural Data and Segment Data. Provides business-to-business services to Service Providers in the form of queries that create Report Data. The Data Engine shall not have directly identifying personal information, e.g. name, address, DoB, etc.

**Service Provider**: Entity acting as the primary link between the Data Engine and Consumer for data services. It will be able to use the Behavioural Data held by the Data Engine to develop personalised services for Consumers based on their everyday behaviours. Service Providers shall not collect or retain Atoms.

**Identity Authority**: Entity responsible for administering the unique identity keys that provide pseudonymous security and ensure the interoperability and universality of the ecosystem. The Identity Authority is specified in “IDAInterfaceSpecification”.

**Operator:** Entity acting as the primary link between the Service Provider and the Consumer. It will hold the DIPI required to engage with the Consumer. It will interact with the Consumer to set-up services and deliver reports from Service Provider. It may help transit Behavioural Data to Data Engine.

**Consumer:** The individual/data subject involved in the ecosystem. His/her behavioural data will be processed by certain actors in the ecosystem.

**Ecosystem:** The ecosystem is defined as ‘the extended set of corporate and individual actors who interact for their mutual benefit via the medium of the specification and under appropriate voluntarily entered into legal agreements’.

# HTTP Protocol

## Overview of HTTP Protocols

All Coelition interfaces are designed around the HTTP protocol stack [[HTTP](http://www.w3.org/Protocols/)] and in particular rely on the REST based operational model. Each message includes one of the HTTP verbs, in particular GET or POST only, and further information depending on the operation being performed. This later information is included in the message body and encoded in JSON format [[JSON](https://tools.ietf.org/html/rfc7159)].

In line with REST style protocol conventions, all accessible entities in the system will be identifiable and reachable through dereferencing a URL unique to that entity. Entry to the system as a whole is via a well-known initial URI, known as the Data Engine Home URI.

## Media Types for Messages

If the media type is present in the message, it shall be “application/json”. Atom server implementations shall accept message with this media type or none. However, they may reject malformed or oversized messages.

## Operations

Only two operations are supported by the Coelition Behavioural Atom protocol. The first is a GET operation directed at the Data Engine Home URI, which returns general information about the Data Engine and in particular the URI of the Atom POST operation URI.

### Data Engine Information Request

Every Data Engine will publish its Data Engine Home URI. Performing a GET on this URI shall return general information about the Data Engine as JSON object. The fields returned shall include the “atomsURI”, the “queryURI”, and the “managementURI” encoded as strings.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method** | **Request** | **Response**  **Status** | **Response Content-Type** | **Response Body** |
| GET | None | 200 (OK) | application/json | JSON object |
| GET | Any | 415 (Unsupported Media Type) | None | None |
| POST | Any | 405 (Method Not Allowed) | None | None |

The JSON object of the response may contain fields with information about the Data Engine. The fields returned must include the “atomsURI”, the “queryURI”, and the “managementURI”; These are the target URLs to be used for adding Atoms, querying Atoms and managing access to the data engine.

Example request message:

GET /home

Example response message:

HTTP/1.1 200 OK

{“atomsURI”: “https://www.dataengine.com/atoms”,

“queryURI”: “https://www.dataengine.com/query”,

“managementURI”: “https://www.dataengine.com/management”}

### Atom POST

To add an Atom to the Data Engine, a POST operation is sent to the Atom POST URI obtained by a preceding GET on the Data Engine Home URI. The POST shall include a non-empty body containing either a single JSON Atom Object or a JSON array containing one or more Atom Objects. The Content-Type of the message must be ‘application/json’.

The response returns HTTP status code 202 (Accepted) and an empty message body if the message format is accepted. One of the following HTTP status codes are returned if an error occurs:

* 400 (Bad Request) if the message does not contain valid JSON or mandatory fields are missing from one or more of the atoms.
* 405 (Method Not Allowed) if another operation (e.g. GET/PUT/DELETE) is used
* 415 (Unsupported Media Type) if the content type is not ‘application/json’
* 500 (Internal Server Error) if an internal error occurred

If the message was not accepted the response message may contain a JSON object with a description of the error, i.e. a list of error messages.

If one or more of the Atoms in a request is missing mandatory elements then the response will be 400 and none of the Atoms will be accepted by the Data Engine. The sender is advised to make a request to submit each atom individually in order that the well formed ones can be accepted.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Method** | **Request**  **Content-Type** | **Request Body** | **Response**  **Status** | **Response Content-Type** | **Response Body** |
| GET | Any | Any | 405 (Method not allowed) | None | None |
| POST | application/  json | Valid JSON Atom | 202 (Accepted) | None | None |
| POST | application/  json | Invalid JSON | 400 (Bad Request) | application/  json | None or JSON Object with a description of the error |
| POST | Any other |  | 415 (Unsupported Media Type) | None | None |

Example request message:

POST /atoms

Content-Type: application/json

Content-Length: nn

{ … }

Example response message:

HTTP/1.1 202 OK

Example request message with an incorrect content type:

POST /atoms

Content-Type: image/png

Content-Length: 2134

{ … }

Example response message:

HTTP/1.1 415 Unsupported Media Type

# Atom Object Definition (JSON)

An atom object has the following format. The top level JSON is an object with the elements described below:

## Header

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| Version | Integer | Version of message format and COEL model | Yes |

## Context

Context of the activity

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| Social | Integer, 0-6 | Indicates the social context of the activity | No |
| Weather | Integer, 0-999 | Indicates the general weather conditions at the time of the activity | No |
| ContextTag | Integer | Context provides the ability to encode “Why” information | No |
| ContextValue | Integer |  | Yes if Context Tag present |

The enumeration values for Social are:

0: Don’t Know

1: Family

2: Colleagues

3: Guests

4: Partner

5: Myself

6: Friends

The enumeration values for Weather are given by the Open Weather Map weather condition code scheme, see http://openweathermap.org/weather-conditions.

There are no ContextTags defined in this version of the specification.

## When

Time and duration of the activity

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| Time | Integer | Seconds since 1970/01/01 00:00Z (Unix timestamp in UTC) | Yes |
| UTCOffset | Integer | UTC Offset in seconds (e.g. UTC+1h = 3600, UTC-2h = -7200…) | No |
| Accuracy | Integer, 0-14 | Indicates accuracy of the time field | No |
| Duration | Integer | Duration of the activity in seconds | No |

The enumeration values for Accuracy are:

0: +/- 1 sec (exact)

1: +/- 1 min (default)

2: +/- 5 mins

3: +/- 15 mins

4: +/- 30 mins

5: +/- 1 hr

6: +/- 2 hrs

7: +/- 4 hrs

8: +/- 8 hrs

9: +/- 12 hrs

10: +/- 24 hrs (weekend)

11: +/- 72 hrs (week)

12: +/- 15 days (month)

13: +/- 91 days (season)

14: +/- 182 days (year)

This value refers to the accuracy reported and not necessarily the actual accuracy at which the measurement was obtained.

## What

Activity as defined by the COEL model

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| Cluster | Integer, 1-32 | COEL cluster, see “COEL V01”. | Yes |
| Class | Integer, 1-99 | COEL class, if available omit otherwise. | No |
| SubClass | Integer, 1-99 | COEL subclass, if available omit otherwise. | No |
| Element | Integer, 1-99 | COEL element, if available omit otherwise. | No |

## How

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| How | Integer, 0-11 | An enumerated value describing how the information was provided | No |
| Certainty | Integer, 0-100 | Percentage, certainty that this atom is associated with the individual indicated in the Who field | No |
| Reliability | Integer, 0-100 | Percentage, reliability of this atom as a whole. The default will be 50, with 100 only being used for correction atoms. | No |

The enumeration values for How are:

0: Don't Know

1: Observed

2: Objectively Measured: Public Infrastructure

3: Objectively Measured: Private Infrastructure

4: Objectively Measured: Fixed Computing Device

5: Objectively Measured: Portable Computer

6: Objectively Measured: Phones and Pocket Device

7: Objectively Measured: Wearables

8: Objectively Measured: Implants

9: Self-Reported

10: Remembered

11: Computationally derived from other Atoms

## Where

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| Exactness | Integer, 0-14 | Format and precision of where fields | No |
| Latitude | Double | GPS location | No |
| Longitude | Double | GPS location | No |
| MCC | Integer | Mobile country code | No |
| MNC | Integer | Mobile network code | No |
| LCA | Integer | Local Area Code | No |
| CID | Integer | Cell ID | No |
| Place | Integer, 0-2 | Profane location code | No |
| Postcode | String | Postcode | No |

The enumeration values for Exactness are:

0: Mobile phone mast connected to the device.

1: Postcode or Zip code very long form.

2: Postcode or Zip code long form.

3: Postcode of Zip code short form

4: Place

5: GPS with accuracy between 0m and 1m.

6: GPS with accuracy between 1m and 5m.

7: GPS with accuracy between 5m and 10m.

8: GPS with accuracy between 10m and 15m.

9: GPS with accuracy between 15m and 20m.

10: GPS with accuracy between 20m and 25m.

11: GPS with accuracy between 25m and 30m.

12: GPS with accuracy between 30m and 50m.

13: GPS with accuracy between 50m and 100m.

14: GPS with accuracy between worse than 100m.

The enumeration values for Place are:

0: Home

1: Work

2: School

## Who

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| DeviceID | String | ID of the device that must be registered with a Consumer ID | Yes if Consumer ID is not present |
| ConsumerID | String | User ID | Yes if Device ID is not present |

## Extension

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Type** | **Description** | **Required** |
| ExtIntTag | Integer | Extension tag for integer extension | No |
| ExtIntValue | Integer | Value of extension annotation | Yes, if ExtIntTag present |
| ExtFltTag | Integer | Extension tag for float extension | No |
| ExtFltValue | Float | Value of extension annotation | Yes if ExtFltTag present |
| ExtStrTag | Integer | Extension tag for string extension | No |
| ExtStrValue | String | Value of extension annotation | Yes if ExtStrTag present |

Some proposed tags and values are (can be either integer or float depending on the precision available/needed):

1001 Resting heart rate bpm

1002 Average heart rate bpm

1003 Maximum heart rate bpm

1004 Blood pressure Encoded (SSSDDD)

1005 Weight kg

1006 Respiratory rate bpm

1007 Lung capacity cl

1008 Temperature C

1009 Oxygen saturation %

1010 Calories ingested kcal

1011 Calories burned kcal

1012 Steps taken count

1013 Distance km

1014 Climb m

1015 Body fat %

1016 Metabolic equivalent MET

1017 Water intake cl

# Atom Format Definition (Compressed Binary)

A compressed binary format for Atoms to reduce transmission times for low bandwidth/low power settings has been conceived and prototyped. The binary format is sufficient small to be embodied in an SMS message and is suited to simple devices that cannot easily format a JSON message. The specification for the compressed binary format will be defined at a later time.

# Security Management for Atom Post

Atom POST uses anonymous TLS only. The Data Engine cannot authenticate the sender, since the Data Engine has no relationship with the consumer. Note that the ConsumerID or DeviceID must have been registered by an Operator for the Atom to be accepted.