Classification of Everyday Living Version 1.0

Committee Specification Draft 03 / Public Review Draft 02

23 November 2017

Specification URIs

This version:

http://docs.oasis-open.org/coel/COEL/v1.0/csprd02/COEL-v1.0-csprd02.docx (Authoritative)
http://docs.oasis-open.org/coel/COEL/v1.0/csprd02/COEL-v1.0-csprd02.html
http://docs.oasis-open.org/coel/COEL/v1.0/csprd02/COEL-v1.0-csprd02.pdf

Previous version:

http://docs.oasis-open.org/coel/COEL/v1.0/csprd01/COEL-v1.0-csprd01.docx (Authoritative)
http://docs.oasis-open.org/coel/COEL/v1.0/csprd01/COEL-v1.0-csprd01.html
http://docs.oasis-open.org/coel/COEL/v1.0/csprd01/COEL-v1.0-csprd01.pdf

Latest version:

http://docs.oasis-open.org/coel/COEL/v1.0/COEL-v1.0.docx (Authoritative)
http://docs.oasis-open.org/coel/COEL/v1.0/COEL-v1.0.html
http://docs.oasis-open.org/coel/COEL/v1.0/COEL-v1.0.pdf

Technical Committee:

OASIS Classification of Everyday Living (COEL) TC

Chairs:

Joss Langford (joss@activinsights.co.uk), Activinsights Ltd
David Snelling (David.Snelling@UK.Fujitsu.com), Fujitsu Limited

Editors:

Paul Bruton (Paul.Bruton@tessella.com), Tessella Ltd.
Joss Langford (joss@activinsights.co.uk), Activinsights Ltd
Matthew Reed (matt@coelition.org), Coelition
David Snelling (Dave.Snelling@UK.Fujitsu.com), Fujitsu Limited

Additional artifacts:

This prose specification is one component of a Work Product that also includes:

- COEL model v1.0: http://docs.oasis-open.org/coel/COEL/v1.0/csprd02/model/coel.json

Abstract:

This document defines the Classification of Everyday Living (COEL) version 1.0 specification for the complete implementation of a compliant system. Examples and non-normative material are also offered as guidance.

Status:

This document was last revised or approved by the OASIS Classification of Everyday Living (COEL) TC on the above date. The level of approval is also listed above. Check the “Latest version” location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=coel#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "Send A Comment button on the TC's web page at https://www.oasis-open.org/committees/coel/.

Citation format:

When referencing this specification the following citation format should be used:

[COEL-COEL-v1.0]

Classification of Everyday Living Version 1.0. Edited by Paul Bruton, Joss Langford, Matthew Reed, and David Snelling. 23 November 2017. OASIS Committee Specification Draft 03 / Public Review Draft 02. http://docs.oasis-open.org/coel/COEL/v1.0/csprd02/COEL-v1.0-csprd02.html. Latest version: http://docs.oasis-open.org/coel/COEL/v1.0/COEL-v1.0.html.

Notices

Table of Contents

Table of Figures

1 Introduction

1.0 IPR Policy

This Committee Specification Public Review Draft is provided under the RF on RAND Terms Mode of the OASIS IPR Policy, the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (https://www.oasis-open.org/committees/coel/ipr.php).

1.1 Objective

The COEL Specification provides a clear and robust framework for implementing a distributed system capable of capturing data relating to an individual as discrete events. It facilitates a privacy-by-design approach for personalised digital services, IoT applications where devices are collecting information about identifiable individuals and the coding of behavioural attributes in identity solutions.

The COEL Specification contains an extensive and detailed taxonomy of human behaviour. The taxonomy allows data from different systems to be encoded in a common format, preserving the meaning of the data across different applications. This ability to integrate universally at the data level, rather than just the technology level, is known as semantic harmonisation and provides full data portability. The communication protocols needed to support system interoperability across a wide range of implementations are also included.

Central to the specification is the separation of static and dynamic personal data. Static data are those pieces of information about an individual that do not change or change very slowly or infrequently which are often used as direct identifiers. Dynamic data are those that describe the sequence of behaviours of an individual over time. This separation of data types provides many advantages for both privacy and security; it is known as pseudonymisation. The COEL Specification provides the means to achieve this separation of data as it is collected rather than as a later operation (pseudonymisation at source).

1.2 Summary of key COEL concepts

The overall approach is motivated by a desire to create **an international standard for collecting and handling personal data** that provides both privacy for consumers and opportunities for enterprise. This aspiration was born from a recognition that the digitisation of commercial and social transactions is dissolving the boundary between the physical and virtual worlds and the digital data trail of choices that people leave behind them not only **enables personalised services, but also poses a potential privacy challenge**. For further background on these concepts see **[Data to Life]**.

The COEL Specification is built on the systematic application of the idea that the events of everyday life can be treated as **Behavioural Atoms**. Although an uncountable number of events happen in the lives of billions of people around the world each day, only a very limited number of types of elemental behaviour make up everyday life. What makes an individual human's life unique is not a huge range of types of Behavioural Atom, but the infinitely diverse ways humans string these Atoms together into the rituals and habits of daily life. On the whole, quality of life is not determined primarily by exceptional, one-off events such as marriage or births, but instead by the fine-textured fabric of everyday life.

Based on insights that the authors of the specification have gained through practically implementing a Behavioural Atom approach, the specification is built around a sharply definition of an Atom: **'a transient event, relating to an individual, that can be objectively recorded by a person or device'**. In a digitised world, these Atoms are normally digitised at source and recorded by networked devices and user interfaces.

A useful structure for recording a Behavioural Atom makes use of the following six pieces of information about the event:

- **What** type of event was recorded?
- **When** did the event begin and what was its duration?
- **How** was the event recorded?
- **Why** was the event recorded, or which event preceded it?
- **Who** was the event associated with?
- **Where** did the event happen?

Using this method, it is possible to create a practical implementation that can effectively guarantee that each and **every Behavioural Atom ever generated is unique**. A large collection of Behavioural Atoms simultaneously shows some of the advantages of unstructured data (it can be queried in an open-ended manner) and the advantages of highly structured data. It is an example of a **micro-structured** form of data.

The ambition of recording daily life in detail seems like an impossibly complex task, but the Behavioural Atom approach described here can be used to build a useful picture very quickly. It deliberately ignores the issue of **why people do things** (psychology) in order to build up a simple picture of **what people do** (observational science) to provide a new way of looking at human activities.

The best insight & knowledge about human behaviour patterns comes when **multiple information sources are brought together**, which is one reason why the **COEL Specification is designed for interoperability and data portability**. With a tool as powerful as this, providers of services need to ensure that people are protected and feel confident to use it in an **open, transparent and auditable manner**.

1.3 Implementations

A wide range of different services and processes can qualify as implementations of the COEL Specification. Examples include, but are not limited to:

- A **Data Engine** could use the COEL Specification to structure and manage the dynamic personal data it is set up to receive and process;
- A **Personal Data Store** could use the COEL Specification to structure the dynamic personal data it is designed to manage;
- A **Personal Electronic Device** could use the COEL Specification to communicate the personal event data that it can output;
- An **Internet of Things (IoT) Device** which interacts with identifiable individuals could use the COEL Specification to communicate the personal event data that it can output;
- An **Identity Authority** could use the COEL Specification to deliver and check unique pseudonymised keys;
- A **Data Portability Exchange** could use the COEL Specification to translate personal data stored in another format into compliant Behavioural Atom data;
- A **User Interface** could use the COEL Specification to code and interpret interactions with an individual;
- A **Customer Relationship Manager (CRM)** could use the COEL Specification to code, store and analyse interactions with an individual.

1.4 Terminology

A set of key normative definitions are presented in the Glossary in section 1.8. Several novel terms are used to describe activities that are associated with use of the COEL Specification.

Sections marked "non-normative" in the section title are informative only and not subject to conformance clauses. When a section has been marked as informative, all subsections of that section are also informative and not subject to conformance clauses. All examples, figures and introduction sections are informative only.

1.5 Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]

1.6 Normative References

[KI-CR-v1.0.0]	*Kantara CISWG Consent Receipt*. https://kantarainitiative.org/confluence/display/infosharing/Consent+Receipt+Specification
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. http://www.ietf.org/rfc/rfc2119.txt
[RFC2616]	Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999. http://www.rfc-editor.org/info/rfc2616
[ISO3166]	*ISO 3166 Country codes.* http://www.iso.org/iso/country_codes
[ISO/IEC 5218]	Codes for the representation of human sexes, December 2004. http://www.iso.org/iso/catalogue_detail.htm?csnumber=36266
[RFC3339]	Klyne, G., Newman, C., “Date and Time on the Internet: Timestamps”, RFC 3339, July 2002. http://www.ietf.org/rfc/rfc3339.txt
[RFC3986]	Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005. http://www.rfc-editor.org/info/rfc3986
[RFC4122]	Leach, P., Mealling, M., Salz, R., “A Universally Unique Identifier (UUID) URN Namespace”, RFC 4122, July 2005. http://www.ietf.org/html/rfc4122
[RFC4627]	D. Crockford, The application/json Media Type for JavaScript Object Notation (JSON), July 2006. http://www.ietf.org/rfc/rfc4627.txt
[RFC5246]	Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008. http://www.ietf.org/rfc/rfc5246.txt
[RFC7617]	J. Reschke, Ed., "The 'Basic' HTTP Authentication Scheme", RFC 7617, September 2015. http://www.ietf.org/rfc/rfc7617.txt

1.7 Non-Normative References

[Coelition]	http://www.coelition.org
[Data to Life]	Reed, M. & Langford, J. (2013). Data to Life. Coelition, London. ISBN 978-0957609402
[App-CR-V.9.3]	*Kantara CISWG Consent Receipt Example Purpose Categories*. Latest version: http://kantarainitiative.org/confluence/display/infosharing/Appendix+CR+-+V.9.3+-+Example+Purpose+Categories
[Weather]	*OpenWeatherMap, Weather Condition Codes.* Latest version: http://openweathermap.org/weather-conditions
[what3words]	http://what3words.com/about/

1.8 Glossary

The following terms are used throughout this specification and have the following definitions when used in context of this document.

Term	***Definition***
Architecture	*Synonymous with COEL Architecture in this document.*
Atom	*Synonymous with COEL Behavioural Atom in this document.*
Aggregated and Anonymised Summary Data	*Non-personal data developed, by suitable techniques, from the analysis of Behavioural Data for the purposes of providing services.*
Associated Service Provider	*Associated Service Providers have agreed access to data within a Data Engine that has been specified and collected by another Service Provider. They can provide services back to the originating Service Provider or link to an Operator.*
BasicAuth	*The underlying connection is protected by transport level security (TLS) [RFC5246] and the client uses HTTP Basic Authentication [RFC7617] for authentication and authorisation.*
Behavioural Data	*Behavioural Data is dynamic personal data describing an individual's activities, i.e. what they have been observed to do or recorded themselves.*
Class	*The second layer of the COEL Model taxonomy.*
Cluster	*The highest and least granular level of the COEL Model taxonomy.*
Consumer	*The generic reference to any individual whose personal data is processed within the Architecture, often referred to as the data subject in regulatory documents and contexts. They might be patients in a healthcare system, citizens in a state setting, users of data management platforms as well as consumers of a commercial digital service.*
ConsumerID	*An IDA unique Pseudonymous Key assigned to a single Consumer. A Consumer can have multiple ConsumerIDs from different Service Providers and multiple profiles with the same Service Provider.*
COEL Architecture	*The complete embodiment of all roles and interactions described by the COEL Specification, also referred to as just 'Architecture' in this document.*
COEL Behavioural Atom	*A transient event, relating to an individual, that can be objectively recorded by a person or device. This is the fundamental data type defined and used extensively throughout the COEL Specification. Any type of life event can be coded into a COEL Behavioural Atom using, as a minimum, a COEL Model code, a unique ConsumerID (or DeviceID) and a DateTime. Also referred to as just 'Atom' and 'Behavioural Atom' in this document.*
COEL Model	*The hierarchical taxonomy of decreasing granularity capable of describing all human events. It includes the data structure, content and version control.*
COEL Specification	*This document and the specifications described within it.*
Data Engine	*The role of a Data Engine is to receive, store and process Behavioural Atoms. The Data Engine provides data services to Service Providers.*
DateTime	*A string formatted as a date-time according to [RFC_3339]. Used to represent the time of an event within the Architecture.*
Device	*Any digital system capable of authoring information about an individual.*
DeviceID	*An IDA unique Pseudonymous Key for a particular Consumer Device.*
Directly Identifying Personal Information (DIPI)	*Static or slow-changing data needed to deliver services to a Consumer including, for example: name, date of birth, contact information, medical/insurance numbers and payment details. DIPI is information that would be generally known as PII (Personally Identifying Information) in some regulatory contexts.*
Ecosystem	*The Ecosystem is defined as the extended set of organisations and individual who interact for their mutual benefit via the medium of the COEL Specification and under appropriate voluntarily entered into legal agreements.*

Element	*The fourth and most granular layer of the COEL Model taxonomy.*
Hardware Developer	*Hardware Developers design and manufacture hardware (such as Internet of Things devices) which are compliant with the COEL Specification for use by Service Providers and Operators.*
Identical Atoms	*Two Atoms are said to be identical if they contain the same fields and values of those fields are exactly the same. In particular, values that differ in case or numeric syntax and not the same. For example, "Home" and "home" are different, and "42.0" and "42" are different.*
Identity Authority (IDA)	*The role of an Identity Authority is to issue and check the unique Pseudonymous Keys that ensure interoperability, universality and security of the Architecture.*
NoAuth	*The underlying connection is protected by server side authenticated transport level security (TLS) [RFC5246], but the TLS connection is anonymous from the client side and therefore no authentication nor authorisation is needed.*
Operator	*The role of an Operator is to manage and administer the relationship with the Consumer. The Operator holds the Directly Identifying Personal Information (DIPI) needed to engage with the Consumer and represents the Consumer within the Architecture.*
OperatorID	*An IDA unique Pseudonymous Key for a particular Operator.*
Pseudonymous Key	*A string formatted as a UUID as defined in [RFC_4122, Section 3] that uniquely identifies pseudonymously, an entity or profile in the Architecture. Unique Pseudonymous Keys are generated by the Identity Authority for use within the Architecture to provide unique codes for the data and transaction of Consumers, Devices, Operators and Service Providers.*
Report Data	*Personal data developed from the querying or analysis of Behavioural Data for the purposes of providing services.*
Segment Data	*Year of birth, gender, home time zone (GMT+/-x) and home latitude to single degree resolution.*
Service Embodiment	*A Service Embodiment is an instance of a specific service that uses the Architecture as defined by the Service Provider.*
Service Provider	*The role of a Service Provider is to specify the purposes and types of data to be processed in Service Embodiment. The Service Provider is the link between the Operator and the Data Engine.*
ServiceProviderID	*An IDA unique Pseudonymous Key for a particular Service Provider.*
SubClass	*The third layer of the COEL Model taxonomy.*
Technical Service Developer	*Technical Service Developers create tools, infrastructure and software for managing data or services within the Architecture. They do not directly manage services or personal data. They include: app developers for Service Providers, development agencies that create Service Provider or Data Engine or other infrastructure.*

2 The COEL Architecture (non-normative)

2.1 Introduction

The COEL Specification is structured around a number of well-defined **roles** and defines the **interfaces** between these roles.

This approach is supported by a strict definition of, and separation of, **data types**. The most significant separation is between the static personal data needed to establish a service and the dynamic personal data (Behavioural Data) that allows a service to be created, managed and personalised.

2.2 Data Types

2.2.1 Behavioural Data

Behavioural Data is dynamic personal data describing an individual person's activities, i.e. what they have been observed to do, or recorded themselves. Any type of activity or life event can be coded using the hierarchical taxonomy of the COEL Model. A single instance or event is known as a COEL Behavioural Atom (Atom) that codes a specific human event relating to one individual in time. These Atoms are small blocks of self-describing, micro-structured data that can also code the duration of events, how they were observed, where they occurred, the context and the purposes for which they can be used.

2.2.2 Directly Identifying Personal Data (DIPI)

Directly Identifying Personal Data (DIPI) is the static or slow-changing data needed to establish services for a Consumer including, for example: name, date of birth, contact information, medical/insurance numbers and payment details. DIPI specifically excludes all event-based information (Behavioural Data / Atoms). DIPI is information that would be generally known as PII (Personally Identifying Information) in some regulatory contexts.

2.2.3 Segment Data

The Segment Data is the only static personal data that can be accessed throughout the Architecture, it comprises 4 items: year of birth, gender, home time zone (GMT +/- x) and home latitude to single degree resolution.

2.2.4 Report Data

Report Data is personal data developed from the querying or analysis of Behavioural Data for the purposes of providing services.

2.2.5 Aggregated and Anonymised Summary Data

Aggregated and Anonymised Summary Data is non-personal data developed, by suitable techniques, from the analysis of Behavioural Data for the purposes of providing services.

2.3 Roles

2.3.1 Identity Authority

The role of an Identity Authority is to issue and check the unique Pseudonymous Keys that ensure interoperability, universality and security of the Architecture. It oversees the effective, open running of the Architecture and administers the operation of the Identity Authority service.

In a privacy-by-design implementation of the COEL Specification, the Identity Authority does not take on any other role in the Ecosystem and cannot gain profit or commercial advantage through its role.

2.3.2 Date Engine

The role of a Data Engine is to receive, store and process Behavioural Atoms. A Data Engine provides data services to Service Providers. These data services can be in the form of queries that create Report Data or Aggregated and Anonymised Summary Data.

A Data Engine can exist in a centralised form (as an organisation providing consumer services), a distributed private form (personal data store) or a distributed public form (ledger).

The Segment Data is the maximal static personal data that a Data Engine can request for the purposes of categorising and anonymising Behavioural Atoms.

In a privacy-by-design implementation, a Data Engine will not take on the roles of Service Provider or Operator. This ensures static personal data and the dynamic personal data are held by different actors, delivering increased data security and a separation of powers. In most jurisdictions and implementations, the Data Engine role is a data processing role that is conducted on behalf of the Service Provider.

2.3.3 Service Provider

The role of a Service Provider is to specify the purposes and types of data to be processed in a Service Embodiment. A Service Provider is the link between an Operator and a Data Engine.

Service Providers can query the Behavioural Data held by a Data Engine to develop personalised services for Consumers based on their individual behavioural preference. These services are then delivered via the Operator. Service Providers will often be consumer-facing brands.

An Associated Service Provider is a Service Provider that has access to data collected by another Service Provider to provide a service to a Consumer or Service Provider.

In a privacy-by-design implementation, the Service Provider role and Operator role can be conducted by the same organisation but neither will hold the dynamic personal data (Behavioural Atoms). In most jurisdictions and implementations, the Service Provider role will be a data controller role.

2.3.4 Operator

The role of an Operator is to manage and administer the relationship with the Consumer. The Operator holds the Directly Identifying Personal Information (DIPI) needed to engage with the Consumer and represents the Consumer within the Architecture.

An Operator might be an independent app, exist within a Service Provider or be an independent organisation. Operators only receive information from their Consumers and their Service Provider.

In most jurisdictions and implementations, the Operator role will be a data controller role.

2.3.5 Consumer

The Consumer is any individual human being, whose personal data is processed within the Architecture, often referred to as the data subject in regulatory texts. They might be patients in a healthcare system, citizens in a state setting, users of data management platforms as well as consumers of a commercial digital service. All Devices are associated with one or more Consumers.

2.4 Interfaces

There are four interfaces described in the COEL Specification. One is hosted by the Identity Authority:

- The Identity Authority Interface (IDA) allows the allocation of Pseudonymous Keys.

The remaining three interfaces are hosted by a Data Engine:

- The Minimal Management Interface (MMI) allows Service Providers and Operators to manage Consumers and Devices within a Service Embodiment;

- The Behavioural Atom Protocol Interface (BAP) is the mechanism to send Behavioural Data to the Data Engine;
- The Public Query Interface (PQI) returns Report Data from queries of Behavioural Data and Segment Data held by the Data Engine.

The relationships between the interfaces, roles and data types are shown in Figure 1.

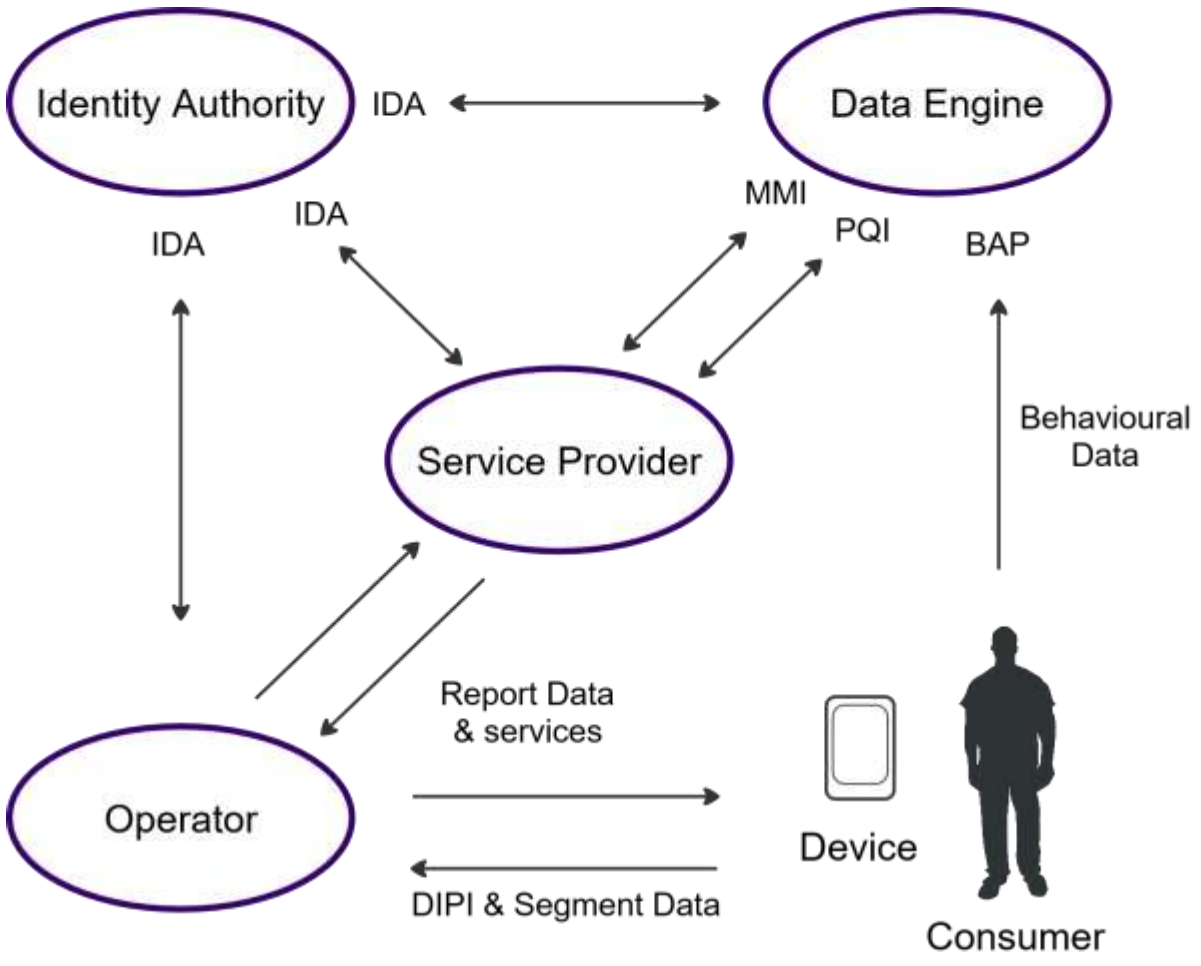


Figure 1 : A representation of the Ecosystem interfaces, roles and data types

2.5 General Operation and Data Flows

The Identity Authority issues a unique Pseudonymous Key to the Operator when the Consumer is enrolled. Once this has been registered with the Data Engine it becomes the ConsumerID and replaces the DIPI in transactions other than those between the Operator and Consumer. In IoT implementations, a unique Pseudonymous Key is assigned to the Device (DeviceID) and registered to a Consumer in the Data Engine.

In normal operation the Behavioural Data will stay with the Data Engine unless the Service Provider needs to provide non-standard services or the Consumer makes a specific data request.

The Segment Data is sent directly to a Data Engine by the Operator when the Consumer is first registered and can be recalled by the Service Provider.

The Service Provider can also use the Identity Authority service for management purposes and to request keys.

3 COEL by Example (non-normative)

3.1 Introduction

This section shows a sequence of operations to demonstrate the technical embodiment of the Architecture that has been described.

3.2 Basic Operations

3.2.1 Service Provider registered with Data Engine

a) Check the status of the IDA

```
GET      <IdentityAuthorityURI>/home
```

Response from IDA:

```
{
  "IdentityAuthorityURI": "<IdentityAuthorityURI>",
  "ServerTime": 1507789656,
  "IdentityAuthorityStatus": "Up",
  "CoelSpecificationVersion": [1,0]
}
```

b) IDA creates new Data Engine (Validator)

(Out of bounds of this specification.)

```
Header   Authorization: Basic base64'IDAadmin_ID':'password'
POST     <IdentityAuthorityURI>/users
{
  "Name": "Data Engine 1",
  "Username": "DE1@example.com",
  "Role": "Validator"
}
```

Response from IDA:

```
{
  "Id": "99a23518-0ca7-4e7a-9fec-2c0180a2dee0",
  "Name": "Data Engine 1",
  "Username": "DE1@example.com",
  "Role": "Validator",
  "Password":
"am5HdEFtdnBPSHdLNTJMcTRKUllub2NESTVRUG9JdjhTQko0NFZmMEtGSTFxMXc3Qk91dUl3YlJFa
G9qc3R0RHZnS2FnNzFiVFdkcXRuWGZYamhSTE9qN3dTZGJtdEM2T2FxV1pRcndUM05TUEVWd0lJUFg
xVVNjTkpKem4yZ2Q=",
  "Enabled": true
}
```

Data Engine now has a 'DE_IDA_ID' & 'password' for IDA access.

c) Data Engine creates credentials for validation

(Out of bounds of this specification.)

```
Header   Authorization: Basic base64'DE_IDA_ID':'password'
POST     <IdentityAuthorityURI>/users/99a23518-0ca7-4e7a-9fec-
2c0180a2dee0/api-credentials
```

Response from IDA:

```
{
  "Id": "95cb43f9-9b49-42ac-bd0b-f99c828f2d52",
  "Role": "Validator",
  "Password":
"cE1uandDT1lRYmhYM1JTZ2dydUpVM3BGbUVlUTAyaDJXeWp2TzI5S3VBS1NBWmZiTjB6ZmdXWWo2V
21rTWs5YTNSVVN0UDRwTXU4Y1FuVUlWblVrVkwwQUQydmN1TEFRbnF5MG52ZllRWkR2OG9zMzBCWll
5RDRwVm55MzlTOGc=",
  "Enabled": true
}
```

Data Engine now has a 'DE_IDA_cred' & 'password' for validating Pseudonymous Keys.

d) IDA creates new Service Provider (B2B Generator)

(Out of bounds of this specification.)

```
Header   Authorization: Basic base64'IDAadmin_ID':'password'
POST     <IdentityAuthorityURI>/users
{
  "Name": "Service Provider 1",
  "Username": "SP1@example.com",
  "Role": "B2BGenerator"
}
```

Response from IDA:

```
{
  "Id": "c1fcde19-5d6d-4580-983a-5918c64103a9",
  "Name": "Service Provider 1",
  "Username": "SP1@example.com",
  "Role": "B2BGenerator",
  "Password":
"NURpejBiYVhRWThycllUNFpka2ltTEhiQUxjemt4SnFHTGxMY1U5MmNNcjhKWFVCaVZaODNPRGpyV
GJyZlJlVndyYlhRNDJpQmFLeE9PSmdxVEwzVTJXN25UdHpoZ3I0c0FRNFZrUjZzUnpneGtkZXAyajJ
PTlZaVlZLWkZoeTk=",
  "Enabled": true
}
```

Service Provider now has a 'SP_IDA_ID' & 'password' for IDA access. The ID returned here is also the Service Provider's unique identifier in the Service Embodiment.

e) Data Engine registers Service Provider

(Out of bounds of this specification.)

Service Provider gains 'SP_DE_ID' & 'password' to access Data Engine.

3.2.2 Operator registered with Data Engine

a) Service Provider creates Operator with IDA

(Out of bounds of this specification.)

```
Header   Authorization: Basic base64'SP_IDA_ID':'password'
POST     <IdentityAuthorityURI>/users/c1fcde19-5d6d-4580-983a-
5918c64103a9/operator
```

Response from IDA:

```
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27",
  "TimeStamp": "2017-10-12T08:12:14.5328496Z",
  "Signature":
"kl3u/y4C2ENrYqFfytIFPn1+O4vaDMbNeuUhFRdwzobMv4tDuQekvZrN28LBAs5ihMrFKXPWLcafE
Z1WyrZs1bRoYz97PlqmuU8RLInuubNBu28O/xQvWhkG4FxUCkRPUe2URwEpDgC+1URypBR0UtZjw9I
3YwrjFI4X+MRm240=",
  "Password":
"SGlCMG9lQm04RjFLYUdoVXhESEhkVkx4a1YwWW8wYnBPWEs4Sm5zY1poanlwOWNOaDRucjk5T2l2e
mNNSk12WEhJUks2YTRyUm42aWczNzFtTTdDUUM1WlNFY09yVndKa2Vwc2NRZ2VzRFJua2hOaW56WjZ
wZ3NSQlkyRDR2eDk="
}
```

b) Service Provider registers Operator with Data Engine

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/operator
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27",
  "TimeStamp": "2017-10-12T08:12:14.5328496Z",
  "Signature":
"kl3u/y4C2ENrYqFfytIFPn1+O4vaDMbNeuUhFRdwzobMv4tDuQekvZrN28LBAs5ihMrFKXPWLcafE
Z1WyrZs1bRoYz97PlqmuU8RLInuubNBu28O/xQvWhkG4FxUCkRPUe2URwEpDgC+1URypBR0UtZjw9I
3YwrjFI4X+MRm240="
}
```

c) Data Engine validates the Operator Pseudonymous Key with IDA

```
Header   Authorization: Basic base64'DE_IDA_cred':'password'
POST     <IdentityAuthorityURI>/validation
{
  "PseudonymousKey": "63a91646-f160-4578-a89a-2bdb9e821e27",
  "TimeStamp": "2017-10-12T08:12:14.5328496Z",
  "Signature":
"kl3u/y4C2ENrYqFfytIFPn1+O4vaDMbNeuUhFRdwzobMv4tDuQekvZrN28LBAs5ihMrFKXPWLcafE
Z1WyrZs1bRoYz97PlqmuU8RLInuubNBu28O/xQvWhkG4FxUCkRPUe2URwEpDgC+1URypBR0UtZjw9I
3YwrjFI4X+MRm240="
}
```

Operator now has a 'OP_IDA_cred' & 'password' for generating Pseudonymous Keys. The ID returned here is also the Operator's unique identifier in the Service Embodiment.

3.2.3 Consumer registered with Operator

a) Operator requests Consumer Pseudonymous Key from IDA

```
Header   Authorization: Basic base64'OP_IDA_cred':'password'
POST     <IdentityAuthorityURI>/pseudonymouskey
```

Response from IDA:

```
{
  "PseudonymousKey": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5",
  "TimeStamp": "2017-10-12T10:33:43.8601264Z",
  "Signature":
"o0nMAAme2J1h3vjB2a1Qif04who43R3W06kEOK3jyygm85+3MssGzey+I/by3aFujTAFDQNmTt8aI
FYUosG32hnrmOPiNqBFeqEJM8LSOS5uOTbRGu+g1N1vmIbJpknC47nSfF2OFW3ujD8G+1+tNCJg11i
aLdsemFBdsnmgM2w="
}
```

b) Operator registers Operator with Data Engine

```
POST     <ManagementURI>/operator/consumer
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27",
  "ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5",
  "TimeStamp": "2017-10-12T10:33:43.8601264Z",
  "Signature":
"o0nMAAme2J1h3vjB2a1Qif04who43R3W06kEOK3jyygm85+3MssGzey+I/by3aFujTAFDQNmTt8aI
FYUosG32hnrmOPiNqBFeqEJM8LSOS5uOTbRGu+g1N1vmIbJpknC47nSfF2OFW3ujD8G+1+tNCJg11i
aLdsemFBdsnmgM2w=",
  "SegmentData": {
    "ResidentTimeZone": "+03:00",
    "ResidentLatitude": 51,
    "Gender": 2,
    "YearOfBirth": 1993
  }
}
```

c) Data Engine validates the Consumer Pseudonymous Key with IDA

```
Header   Authorization: Basic base64'DE_IDA_cred':'password'
POST     <IdentityAuthorityURI>/validation
{
  "PseudonymousKey": " f7b0ce76-30a8-4544-aa2e-9667f6228ae5",
  "TimeStamp": "2017-10-12T10:33:43.8601264Z",
  "Signature":
"o0nMAAme2J1h3vjB2a1Qif04who43R3W06kEOK3jyygm85+3MssGzey+I/by3aFujTAFDQNmTt8aI
FYUosG32hnrmOPiNqBFeqEJM8LSOS5uOTbRGu+g1N1vmIbJpknC47nSfF2OFW3ujD8G+1+tNCJg11i
aLdsemFBdsnmgM2w="
}
```

3.2.4 Device registered with Data Engine

a) Service Provider creates credentials for generating Pseudonymous Keys

(Out of bounds of this specification.)

```
Header   Authorization: Basic base64'SP_IDA_ID':'password'
POST     <IdentityAuthorityURI>/users/c1fcde19-5d6d-4580-983a-
5918c64103a9/api-credentials
```

Response from IDA:

```
{
  "Id": "13d654ff-2423-4d0e-9a91-388cc28a0d0e",
  "Role": "Generator",
  "Password":
"OVpkZDdkaFowaVg0YzFZZTlJM0hka2VoQ1J0QVRYTkhTWmdaeHlQc2k0cnhORkRSSFdZR2lyRzJKd
3Vwa0lGOThKM0NuNWc2cVJ0NE81YWNsUURObUJiaEI3RkJOb1JEQ1RyYnZwQTU0U0M4MG5WZzBXZ1o
0ZnZoV1h1WXBsZmk=",
  "Enabled": true
}
```

Service Provider now has a 'SP_IDA_cred' & 'password' for generating Pseudonymous Keys.

b) Service Provider requests Pseudonymous Key Batch from IDA

```
Header   Authorization: Basic base64'SP_IDA_cred':'password'
POST     <IdentityAuthorityURI>/pseudonymouskeybatch
{
  "Size": 3
}
```

Response from IDA:

```
{
  "PseudonymousKeys": [
    "a01216b7-9a6f-4eb2-ad68-7b6f968f897d",
    "e77bcf3f-d37e-40df-9096-79f2c3b3266f",
    "0b7e6a10-9407-4feb-b143-7b84ad2bb2c6"
  ],
  "TimeStamp": "2017-10-12T16:46:26.2080914Z",
  "Signature":
"f6vXD7MfuddvNHm/5+gAgkHOO0fAnjExOGbMlT7hD/GKABeDzoH3p7CLls8gaS5ukSfnMx5IyW0ix
IBgl9hl4eqTK0pOVQ2abO4rcFcz8TaXnA2sLhDSP1l14okMc6/z3BEclZ4u3sAZbvHXOMzOSG9LkS5
aO33hBguAyqINn0Q="
}
```

c) Service Provider registers Devices with Data Engine

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/registerDevices
{
  "DeviceIDs": [
    "a01216b7-9a6f-4eb2-ad68-7b6f968f897d",
    "e77bcf3f-d37e-40df-9096-79f2c3b3266f",
    "0b7e6a10-9407-4feb-b143-7b84ad2bb2c6"
  ],
  "TimeStamp": "2017-10-12T16:46:26.2080914Z",
  "Signature":
"f6vXD7MfuddvNHm/5+gAgkHOO0fAnjExOGbMlT7hD/GKABeDzoH3p7CLls8gaS5ukSfnMx5IyW0ix
IBgl9hl4eqTK0pOVQ2abO4rcFcz8TaXnA2sLhDSP1l14okMc6/z3BEclZ4u3sAZbvHXOMzOSG9LkS5
aO33hBguAyqINn0Q=",
  "DeviceType": "Personal"
}
```

d) Data Engine validates the Devices with IDA

```
Header   Authorization: Basic base64'DE_IDA_cred':'password'
POST     <IdentityAuthorityURI>/validation
{
  "PseudonymousKeys": [
    "a01216b7-9a6f-4eb2-ad68-7b6f968f897d",
    "e77bcf3f-d37e-40df-9096-79f2c3b3266f",
    "0b7e6a10-9407-4feb-b143-7b84ad2bb2c6"
  ],
  "TimeStamp": "2017-10-12T16:46:26.2080914Z",
  "Signature":
"f6vXD7MfuddvNHm/5+gAgkHOO0fAnjExOGbMlT7hD/GKABeDzoH3p7CLls8gaS5ukSfnMx5IyW0ix
IBgl9hl4eqTK0pOVQ2abO4rcFcz8TaXnA2sLhDSP1l14okMc6/z3BEclZ4u3sAZbvHXOMzOSG9LkS5
aO33hBguAyqINn0Q="
}
```

3.2.5 Device assigned to Consumer

a) Operator assigns Device to Consumer with Data Engine

```
POST     <ManagementURI>/operator/device
{
  "DeviceID": "a01216b7-9a6f-4eb2-ad68-7b6f968f897d",
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27",
  "ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5"
}
```

3.2.6 Send Behavioural Data

a) Atom sent with ConsumerID

```
POST     <AtomsURI>/
[{
    "Header": {"Version": [1,0,1,0]},
    "Who": {"ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5"},
    "What": {"Cluster": 4,"Class": 4,"SubClass": 1,"Element": 4},
    "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1507864341,"Duration":
600},
    "Reliability": 70,
    "Where": {"Exactness": 2, "Postcode": "UB4 8FE"},
    "How": {"How": 9},
    "Context": 4
}]
```

b) Atom sent with DeviceID

```
POST     <AtomsURI>/
[{
    "Header": { "Version": [1,0,1,0] },
    "Who": {"DeviceID": "a01216b7-9a6f-4eb2-ad68-7b6f968f897d"},
    "What": {"Cluster": 22,"Class": 1,"SubClass": 1,"Element": 2},
    "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1507875158,"Duration": 
3903},
    "Where": {"Exactness": 6,"Latitude": 51.53118159161092,"Longitude": -
0.4319647327069491},
    "How": {"How": 9},
    "Extension": {"ExtFltTag": 10003,"ExtFltValue": 26.2 }
}]
```

3.2.7 Assure Consumer & Operator

a) Service Provider checks whether an Operator has registered a Consumer

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/assure
{
  "ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5",
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27"
}
```

Response from Data Engine:

```
{
"Assured": true
}
```

3.2.8 Report Data created from query

a) Service Provider queries all data for a Consumer in a time window

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <QueryURI>/query
{
  "ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5",
  "Timewindow": {
    "StartTime": 1507334400,
    "EndTime": 1507939200
  },
  "Query": {}
}
```

Response from Data Engine:

```
[{
    "Header": {"Version": [1,0,1,0]},
    "Who": {"ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5"},
    "What": {"Cluster": 4,"Class": 4,"SubClass": 1,"Element": 4},
    "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1507864341,"Duration": 
600},
    "Reliability": 70,
    "Where": {"Exactness": 2, "Postcode": "UB4 8FE"},
    "How": {"How": 9},
    "Context": 4
},
{
    "Header": { "Version": [1,0,1,0] },
    "Who": {"ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5"},
    "What": {"Cluster": 22,"Class": 1,"SubClass": 1,"Element": 2},
    "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1507875158,"Duration": 
3903},
    "Where": {"Exactness": 6,"Latitude": 51.53118159161092,"Longitude": -
0.4319647327069491},
    "How": {"How": 9},
    "Extension": {"ExtFltTag": 10003,"ExtFltValue": 26.2 }
}]
```

(The communication of results of queries from Service Provider to Operator and then Consumer are out of bounds of this specification.)

3.2.9 Retrieve Operator list, suspend & resume Operator

a) Service Provider retrieves Operator list

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/operators
{
  "ServiceProviderID": "c1fcde19-5d6d-4580-983a-5918c64103a9"
}
```

Response from Data Engine:

```
{
  "Operators": [{
      "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27",
      "Suspended": false
    }]
}
```

b) Service Provider suspends Operator

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/suspendOperator
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27"
}
```

c) Service Provider retrieves Operator list

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/operators
{
  "ServiceProviderID": "c1fcde19-5d6d-4580-983a-5918c64103a9"
}
```

Response from Data Engine:

```
{
  "Operators": [{
      "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27",
      "Suspended": true
    }]
}
```

d) Service Provider resumes Operator

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/resumeOperator
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27"
}
```

3.2.10 Retrieve Device list, unassign Device & retrieve Device list

a) Service Provider retrieves Device list

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/devices
{
  "ServiceProviderID": "c1fcde19-5d6d-4580-983a-5918c64103a9"
}
```

Response from Data Engine:

```
{
  "Devices": [
    {
      "DeviceID": "a01216b7-9a6f-4eb2-ad68-7b6f968f897d",
      "DeviceType": "Personal",
      "ConsumerIDs": ["f7b0ce76-30a8-4544-aa2e-9667f6228ae5"]
    },
    {
      "DeviceID": "e77bcf3f-d37e-40df-9096-79f2c3b3266f",
      "DeviceType": "Personal",
      "ConsumerIDs": []
    },
    {
      "DeviceID": "0b7e6a10-9407-4feb-b143-7b84ad2bb2c6",
      "DeviceType": "Personal",
      "ConsumerIDs": []
  }]
}
```

b) Service Provider unassigns a Device

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/unassignDevice
{
  "DeviceID": "a01216b7-9a6f-4eb2-ad68-7b6f968f897d"
}
```

c) Service Provider retrieves Device list

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
POST     <ManagementURI>/service-provider/devices
{
  "ServiceProviderID": "c1fcde19-5d6d-4580-983a-5918c64103a9"
}
```

Response from Data Engine:

```
{
  "Devices": [
    {
      "DeviceID": "a01216b7-9a6f-4eb2-ad68-7b6f968f897d",
      "DeviceType": "Personal",
      "ConsumerIDs": []
    },
    {
      "DeviceID": "e77bcf3f-d37e-40df-9096-79f2c3b3266f",
      "DeviceType": "Personal",
      "ConsumerIDs": []
    },
    {
      "DeviceID": "0b7e6a10-9407-4feb-b143-7b84ad2bb2c6",
      "DeviceType": "Personal",
      "ConsumerIDs": []
  }]
}
```

3.2.11 Retrieve Consumer list, request Segment Data, forget Consumer & retrieve Consumer list

a) Service Provider retrieves Consumer list

```
Header   Authorization: Basic base64'SP_DE_ID':'password'
```

```
POST      <ManagementURI>/service-provider/consumers
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27"
}
```

Response from Data Engine:

```
{
  "ConsumerIDs": ["f7b0ce76-30a8-4544-aa2e-9667f6228ae5"]
}
```

b) Service Provider requests Segment Data

```
Header    Authorization: Basic base64'SP_DE_ID':'password'
POST      <QueryURI>/segment
{
  "ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5",
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27"
}
```

Response from Data Engine:

```
{
  "SegmentData": {
    "ResidentTimeZone": "+03:00",
    "ResidentLatitude": 51,
    "Gender": 2,
    "YearOfBirth": 1993
  }
}
```

c) Operator requests Consumer to be forgotten

```
POST      <ManagementURI>/operator/forgetConsumer
{
  "ConsumerID": " f7b0ce76-30a8-4544-aa2e-9667f6228ae5"
}
```

d) Service Provider confirms forget request with Data Engine

(Outside the bounds of this specification.)

e) Service Provider retrieves Consumer list

```
Header    Authorization: Basic base64'SP_DE_ID':'password'
POST      <ManagementURI>/service-provider/consumers
{
  "OperatorID": "63a91646-f160-4578-a89a-2bdb9e821e27"
}
```

Response from Data Engine:

```
{
  "ConsumerIDs": []
}
```

4 The COEL Model

4.1 Introduction

The COEL Model is a hierarchical taxonomy of everyday human life events: it has both a nomenclature (a way of naming things) and a classification (a way to discriminate between different types of thing based on their features or attributes). The COEL Model is very compact by design. Nevertheless, it's high level **structure** and **content** represents a significant knowledge base which is held within the COEL Specification as a **JSON object**.

This first version of the COEL Model already provides codes for more than 5,000 distinct human behaviours and events. This is sufficient to describe most events in most people's lives. The taxonomy structure allows any type of activity or life event to be coded to some level of granularity. The approach to increasing both the granularity and range of events represented in future **versions** of the COEL Model is described.

The COEL Model provides the **semantic** basis for the deep interoperability and data portability of the COEL Specification. It is a comprehensive, and unambiguous tool, for referencing human life events across languages and cultures. As such it is a global asset. A COEL Model code represents a meaning; the reference approach to both **language** and **style** is described.

4.2 COEL Model Specification

4.2.1 Structure

A COEL Model MUST be constructed as a four-level hierarchy for both interoperable machine readability and ease of human understanding. The entities in the lower levels of the structure MUST be sub-types of an entity at the next higher level. Thus, the lower levels represent progressively more detailed views of life events.

The most logical way to describe the structure of the full taxonomy is from the top down. However, the fine-grained (and often most interesting) detail is at the bottom of the hierarchy, at the level of the most basic events.

At the top level of the tree there are about thirty Clusters of event categories that go together. The name of each Cluster has been chosen to be intuitive for users of the classification. Some of these Clusters inevitably have a much richer structure than others, since certain aspects of daily life contain more variation than others.

Below the level of the Clusters come three further levels: Class, SubClass and Element. This structure is shown schematically below.

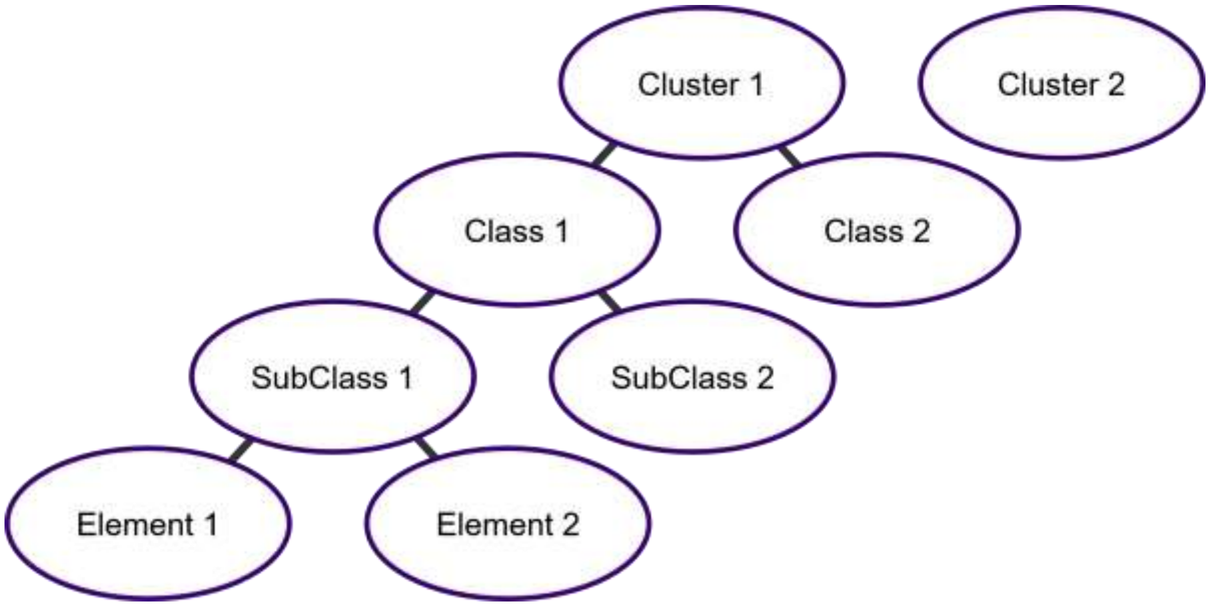


Figure 2 : The structure of the COEL Model hierarchical taxonomy

In addition to the hierarchical taxonomy, a COEL Model MUST contain a version field of two numbers (major and minor).

Applications that refer to or use a COEL Model either as a knowledge base of daily human events or as a data model that embodies that knowledge base MUST reference to this document and additional artefacts.

4.2.2 Content

The content of the COEL Model is constructed according to a set of design principles which support the classification and naming of the events that form our everyday lives.

- **Granular**: beneath the important surface of individual and cultural differences, everyday human behaviour is surprisingly similar. Our daily lives are made up of a finite number of behaviours which have a natural granularity. A COEL Model SHOULD work at this granularity of events.
- **Complete**: a COEL Model SHOULD aim to classify, name and code all the observable daily behaviours that can make up an individual's life, that is, it SHOULD be Collectively Exhaustive.
- **Single-category**: category errors SHOULD be avoided and a COEL Model SHOULD only define observable human behaviours. Personal emotions / thoughts become a type of observable event only when an individual reports those emotions / thoughts, for example in conversation or a digital diary.
- **Hierarchical**: a discrete behavioural event SHOULD sit at the bottom of the logically clustered hierarchy and events that have certain similarities SHOULD be kept together.
- **Distinctive**: events at any single level (Class, Subclass, Element) within a Cluster SHOULD be clearly distinct, that is, they SHOULD be Mutually Exclusive.

The requirement for a COEL Model to be both Mutually Exclusive AND Collectively Exhaustive (MECE) is particularly demanding but valuable.

4.2.3 Semantics and Language

A COEL Model code represents a meaning and the reference descriptions for a COEL Model MUST be in the English language. The COEL Model SHALL NOT be translated by OASIS COEL-TC. Other entities

MAY create translations; however, the original English language version SHALL remain the authoritative version.

Some cultural practices do not have natural English language translations. In these circumstances, the entity SHALL be described in English but the original language word MAY be added after the description.

4.2.4 Style Guide

The name of each Cluster SHOULD be chosen to be intuitive for users of the classification.

The string descriptions MUST be formatted with only the first word capitalised with no punctuation, abbreviations or trailing spaces. The Clusters MUST be single words with no spaces.

4.2.5 Version Control

The COEL Model MUST have a major and a minor version number. Any changes or addition to the COEL Model MUST be recorded with a change in the version numbers. Additions that retain backward compatibility MAY increment just the minor version number. Changes and additions that are not backwardly compatible MUST increment the major version number and reset the minor version number to zero.

When a non-backwards compatible change is made to the COEL Model, this MUST run through the full OASIS process and a new version of the COEL Specification will be released. Backwardly compatible changes MUST be agreed by the OASIS Committee.

4.2.6 JSON Object

The COEL Model exists concretely as a single digital artefact – a JSON object containing five elements that define the Version, Clusters, Classes, SubClasses and Elements.

Key	***Type***	***Description***	***Required***
Version	*Array of Number*	*The integer version numbers [major, minor] of this instance of the COEL Model.*	*Yes*
Clusters	*Array of Object*	*The model Clusters. See below for details.*	*Yes*
Classes	*Array of Object*	*The model Classes. See below for details.*	*Yes*
SubClasses	*Array of Object*	*The model SubClasses. See below for details.*	*Yes*
Elements	*Array of Object*	*The model Elements. See below for details.*	*Yes*

The objects in the Clusters, Classes, SubClasses and Elements arrays SHALL all have the same structure. Each activity is described fully by its Cluster, Class, SubClass, and Element code numbers. When an activity is being used as a general term (or the detail is not sufficient to describe at all four levels) the upper levels MAY be used in place of the more specialized descriptions (by providing a zero value for the lower level code numbers). See example below.

Key	***Type***	***Description***	***Required***
Name	*String*	*The name of the everyday living activity.*	*Yes*
Cluster	*Number*	*The Cluster code number of the activity (integer).*	*Yes*
Class	*Number*	*The Class code number of the activity (integer).*	*Yes*
SubClass	*Number*	*The SubClass code number of the activity (integer).*	*Yes*
Element	*Number*	*The Element code number of the activity (integer).*	*Yes*

Example

A valid subset of the COEL Model showing two example Elements and their SubClass, Class and Cluster.

```
    {
          "Version": [1, 0],
          "Clusters": [{
                 "Cluster": 22,
                 "Name": "Travel",
                 "Class": 0,
                 "SubClass": 0,
                 "Element": 0
          }],
          "Classes": [{
                 "Cluster": 22,
                 "Name": "Non powered",
                 "Class": 1,
                 "SubClass": 0,
                 "Element": 0
          }],
          "SubClasses": [{
                 "Cluster": 22,
                 "Name": "Travel by bike",
                 "Class": 1,
                 "SubClass": 1,
                 "Element": 0
          }],
          "Elements": [{
                        "Cluster": 22,
                        "Name": "Mountain bike",
                        "Class": 1,
                        "SubClass": 1,
                        "Element": 1
                 },
                 {
                        "Cluster": 22,
                        "Name": "Racing bike",
                        "Class": 1,
                        "SubClass": 1,
                        "Element": 2
                 }
          ]}
```

4.3 Permanent location of COEL Model JSON artefacts

The authoritative version of the additional artefact that accompanies this specification (COEL Model V1.0) is located by OASIS as part of the COEL Specification. As new versions of the COEL Model are agreed and new versions of the JSON artefact are formally released by OASIS, they will be added to the URI.

4.4 COEL Model Overview (non-normative)

To provide a human readable top level description of the COEL Model, the following table provides the names and longer form descriptions of the Clusters. Note that any apparent logical ambiguities that can be suggested by these top level cluster names can be resolved by moving down in the hierarchy, where the actual coherence is guaranteed by the full set of elements.

Cluster Name	***Long Form Description***
Personalcare	*All self performed activities related to looking after yourself*
Childcare	*Activities related to looking after children*
Adultcare	*Activities related to looking after adults*
Housework	*Cleaning and day to day running of your dwelling*

Maintenance	*Functional upkeep of your dwelling and possessions*
Animalcare	*Activities related to looking after animals*
Health	*Activities related to your own health*
Medicine	*The diagnosis & treatment of ailments*
Symptoms	*Specific events related to symptoms of illness*
Eating	*The consumption of food items*
Drinking	*The consumption of liquid items*
Cooking	*The preparation of food and drink*
Sleep	*Activities related to preparing for sleep and the timecourse of sleep itself*
Sports	*Sports and predominantly physically active hobbies & pastimes*
Hobbies	*Sports and hobbies using vehicles / equipment*
Spectator	*Activities related to watching sports*
Pastimes	*Participatory pastimes (non-physically active)*
Observer	*Spectator pastimes (non-physically active)*
Media	*All activities involving the use of media*
Shopping	*Activities involved in shopping for physical goods*
Service	*Activities involved in shopping for services*
Travel	*Moving from one place to another for a specific purpose*
Communication	*All methods of socially interacting via communicating face to face, non face to face and to groups & audiences*
Device	*Using electronic devices*
Trials	*Unplanned events which cause irritation or shock*
Education	*Activities involved with the process of acquiring knowledge*
Accident	*Accidents and injuries related to people*
Lifestage	*Life defining events*
Lifestyle	*Events related to lifestyle and type of person*
Task	*Generic work tasks*
Work	*Different types of work*
Mind	*Observable manifestations of emotion*

4.5 Visualising the COEL Model (non-normative)

As a helpful service to users of the COEL Specification, a dynamic visual representation of the latest version of the full COEL Model is provided at **[Coelition]**.

5 The COEL Behavioural Atom

5.1 Introduction

The COEL Behavioural Atom is a small block of self-describing, micro-structured data that codes a specific human event relating to one individual in time. It is defined as a JSON object which can also code the duration of events, how they were observed, where they occurred, the context and the purposes for which they can be used.

5.2 COEL Behavioural Atom Specification

A COEL Behavioural Atom (Atom) is a JSON object containing four REQUIRED elements and an additional five OPTIONAL elements. Each element is itself an object. The following JSON Schema defines the structure, spelling and basic type of each element and sub-element. An Atom MUST comply with this schema, and with the additional constraints specified in the remainder of this section.

5.2.1 Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "required": ["Header", "When", "What", "Who" ],

  "additionalProperties":false,
  "properties":{
    "Header"   : {
        "type": "object",
        "additionalProperties":false,
        "properties":{"Version":{"type":"array"}}},

    "When"     : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "Time":{"type":"integer"},
            "Duration":{"type":"integer"},
            "UTCOffset":{"type":"integer"},
            "Accuracy":{"type":"integer"}}},

    "What"     : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "Cluster": {"type": "integer"},
            "Class": {"type": "integer"},
            "SubClass": {"type": "integer"},
            "Element": {"type": "integer"}}},

    "Who"      : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "ConsumerID":{"type":"string"},
            "DeviceID":{"type":"string"}}},

    "How"      : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "How":{"type":"integer"},
            "Certainty":{"type":"integer"},
            "Reliability":{"type":"integer"}}},

    "Where"    : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "Exactness":{"type":"integer"},
            "Latitude":{"type":"number"},
            "Longitude":{"type":"number"},
            "W3W":{"type":"string"},
            "Place":{"type":"integer"},
            "Postcode":{"type":"string"}}},

    "Context"  : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "Social":{"type":"integer"},
            "Weather":{"type":"integer"},
            "ContextTag":{"type":"integer"},
            "ContextValue":{"type":"integer"}}},

    "Consent"  : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "Jurisdiction":{"type":"string"},
            "Date":{"type":"integer"},
            "RetentionPeriod":{"type":"integer"},
            "Purpose":{"type":"integer"},
            "PolicyURL":{"type":"string"},
            "RecordID":{"type":"string"},
            "RecordService":{"type":"string"}}},

    "Extension": {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "ExtIntTag":{"type":"integer"},
            "ExtIntValue":{"type":"integer"},
            "ExtFltTag":{"type":"integer"},
            "ExtFltValue":{"type":"number"},
            "ExtStrTag":{"type":"integer"},
            "ExtStrValue":{"type":"string"}}}}}
```

5.2.2 Constraints

1. The following elements in an Atom are REQUIRED:
    a. Header;
    b. When;
    c. What;
    d. Who;
    e. Version (in Header);
    f. Time (in When);
    g. Cluster (in What).
2. An atom MUST contain either a DeviceID or a ConsumerID, but not both a DeviceID and a ConsumerID.
3. The following elements MUST appear in pairs. If an Atom contains one, it MUST also contain the other:

    a. ContextTag, ContextValue;
    b. ExtIntTag, ExtIntValue;
    c. ExtFltTag, ExtFltValue;
    d. ExtStrTag, ExtStrValue;
    e. RecordID, RecordService.
4. The following constraints apply within a What element:
    a. If Element present, SubClass MUST also be present;
    b. If SubClass is present, Class MUST also be present.
5. If a Consent element is present, the following elements MUST also be present: Date, RetentionPeriod.

5.2.3 Header

Key	*Type*	*Description*	*Required*
Version	*Array of Number*	*Array of integers [0,1,2,3] indicating the COEL Specification and COEL Model versions used to define this Atom.* *0 - COEL Specification major version number* *1 - COEL Specification minor version number* *2 - COEL Model major version number* *3 - COEL Model minor version number*	*Yes*

The detailed meanings and usage scenarios for the COEL Model version numbers are provided in section 4.2.5.

The COEL Specification MUST have a major and a minor version number. Any changes or addition to the COEL Specification MUST be recorded with a change in the version numbers. Additions that retain backward compatibility MAY increment just the minor version number. Changes and additions that are not backwardly compatible MUST increment the major version number and reset the minor version number to zero.

5.2.4 When

Time and duration of the Atom:

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

The enumeration values for Accuracy SHALL be those defined in Appendix A.

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

Atoms with duration of zero MAY be used and indicate an instantaneous event (or one where the duration is less than a second). A zero duration Atom MAY also be a marker for the end of a sequence of Atoms such as in a running route, see section 5.2.8 Where.

5.2.5 What

Activity recorded by the atom (as defined by the COEL Model in Section 4**)**:

Key	*Type*	*Description*	*Required*
Cluster	*Number*	*Cluster (integer 1-99).*	*Yes*
Class	*Number*	*Class, if available omit otherwise (integer 1-99).*	

SubClass	*Number*	*SubClass, if available omit otherwise (integer 1-99).*	*See 5.2.2 Constraints*
Element	*Number*	*Element, if available omit otherwise (integer 1-99).*	

When appropriate event descriptions are not available in the latest version of the COEL Model, development codes MAY be used for new applications. These codes SHALL use the format 1xxxx (i.e. integers in the range 10000 to 19999). These codes MAY be used at any level of the COEL Model.

5.2.6 Who

Who the Atom relates to:

Key	***Type***	***Description***	***Required***
DeviceID	*String*	*Pseudonymous Key of a Device that is registered with a Consumer.*	*See 5.2.2 Constraints*
ConsumerID	*String*	*Pseudonymous Key for the Consumer (subject, user or patient).*	

5.2.7 How

How the Atom was measured:

Key	***Type***	***Description***	***Required***
How	*Number*	*An enumerated value describing how the information was provided (integer 0-11).*	*No*
Certainty	*Number*	*Percentage, certainty that this Atom is associated with the individual indicated in the Who field (integer 0-100).*	*No*
Reliability	*Number*	*Percentage, reliability of this Atom as a whole. The default SHALL be 50, with 100 only being used for correction Atoms (integer 0-100).*	*No*

The enumeration values for How SHALL be those defined in Appendix A.

If when an Atom was posted from a Device (i.e. the DeviceID was present and the ConsumerID was not), the Certainty value (or 100 if the Certainty value was missing) MUST be divided by the number of ConsumerIDs associated with the DeviceID at the time the Atom was posted. Thus Certainty represents the probability that the Atom is associated with this Consumer.

5.2.8 Where

Where the Atom occurred:

Key	***Type***	***Description***	***Required***
Exactness	*Number*	*Format and precision of where fields (integer 0-14).*	*No*
Latitude	*Number*	*GPS location (double format).*	*No*
Longitude	*Number*	*GPS location (double format).*	*No*
W3W	*String*	*what3words code (word.word.word) see **[what3words]**.*	*No*
Place	*Number*	*Profane location code (integer 0-2).*	*No*
Postcode	*String*	*Postcode.*	*No*

The enumeration values for Exactness and Place SHALL be those defined in Appendix A.

When appropriate enumerated values for Place are not available in the COEL Behavioural Atom Specification, development codes MAY be used for new applications. These codes SHALL use the format 1xxxx (i.e. integers in the range 10000 to 19999).

Where journeys are being recorded the location in this field SHALL be the starting location. The displacement of the journey can be recorded in an extension field and/or the final location MAY be recorded in a subsequent Atom.

5.2.9 Context

Context of the Atom:

Key	*Type*	*Description*	*Required*
Social	*Number*	*Indicates the social context of the activity (integer 0-6).*	*No*
Weather	*Number*	*Indicates the general weather conditions at the time of the activity (integer 0-999).*	*No*
ContextTag	*Number*	*Context provides the ability to encode "Why" information (integer).*	*See 5.2.2 Constraints*
ContextValue	*Number*	*Value of Context annotation (integer).*	

The enumeration values for Social and Weather SHALL be those defined in Appendix A. The enumeration values for Weather are derived from those of **[Weather]**, however the values in Appendix A are normative for this specification.

There are no ContextTags defined in this version of the COEL Behavioural Atom Specification, but these MAY include references to previous Atoms to indicate causality or question / answer pairs to sequence interactions.

5.2.10 Consent and Notice

A summary of the notice given to or consent given by the Consumer for management purposes:

Key	*Type*	*Description*	*Required*
Jurisdiction	*String*	*The jurisdiction in which consent or notice was given. Two letter country code: Alpha-2 representation as defined in **[ISO3166]**.*	*No*
Date	*Number*	*The date of the consent or notice. Seconds since 1970/01/01 00:00Z (integer Unix time stamp in UTC).*	*See 5.2.2 Constraints*
RetentionPeriod	*Number*	*The number of integer seconds stated in the consent or notice for retention or review of retention.*	
Purpose	*Number*	*Single purpose category for which consent or notice was given. Integer enumerated field defined in **[App-CR-V.9.3]**. Multiple Atoms MAY be used to present multiple purposes.*	*No*
PolicyURL	*String*	*The privacy policy and/or notice that applies to the record (HTTP URL).*	*No*
RecordID	*String*	*The unique identifier that represents the record. MAY be a JSON Web Token and MAY be another form of identifier. Any data subject identifiers MUST be encrypted.*	*See 5.2.2 Constraints*
RecordService	*String*	*The URL of the processing service providing the record or the WebTokenID (HTTP URL).*	

The object formats are defined to be compatible with **[KI-CR-v1.0.0]** where possible. The use of a consent receipt as defined by **[KI-CR-v1.0.0]** is possible by generating a "Service/Legal/Consent/Granting consent" Atom at the point of original consent agreement and including the RecordID and RecordService fields. Records of notice can be generated with a "Service/Legal/Notice" Atom in a similar way.

The enumeration values for Purpose SHALL be those defined in Appendix A. The enumeration values for Purpose are those of **[App-CR-V.9.3]**, however Appendix A is normative.

5.2.11 Extension

Additional information about the Atom:

Key	*Type*	*Description*	*Required*
ExtIntTag	*Number*	*Extension tag for integer extension (integer).*	

ExtIntValue	*Number*	*Value of extension annotation (integer).*	*See 5.2.2 Constraints*
ExtFltTag	*Number*	*Extension tag for float extension (integer).*	
ExtFltValue	*Number*	*Value of extension annotation (float).*	
ExtStrTag	*Number*	*Extension tag for string extension (integer).*	
ExtStrValue	*String*	*Value of extension annotation.*	

The tags and values SHALL be those defined in Appendix A (values can be either integer or float depending on the precision available/needed).

When appropriate Extension tags are not available in the COEL Behavioural Atom Specification, development codes MAY be used for new applications. These codes SHALL use the format 1xxxx (i.e. integers in the range 10000 to 19999).

5.3 COEL Behavioural Atom Examples (non-normative)

The following is an example Behavioural Atom for the activity: ‘Housework’, ‘Dishes’, ‘Loading and unloading the dishwasher’, ‘Load the dishwasher’; the time is accurate to +/- 1 minute; it took place at a given postcode, it was reported by the user with a 100% certainty of the ‘Who’ field and a general ‘Reliability’ of 70%, the social context was with a partner

```
{
   "Header":{ "Version":[1,0,1,0]},
   "Who":{ "ConsumerID": "5a702670-ff63-4d1d-ba9d-077dd345ab62"},
   "What":{ "Cluster":4, "Class":4, "SubClass":1, "Element":4},
   "When":{ "UTCOffset":-3600,"Accuracy":0, "Time":1433397180,
"Duration":600},
    "Reliability": 70,
   "Where":{"Exactness":2, "Postcode": "UB4 8FE"},
   "How":{"How":9},
   "Context": 4
}
```

The following is an example COEL Behavioural Atom for the activity: ‘Travel’, ‘Non Powered’, ‘Travelling by bicycle’, ‘Racing bike’; the time is exact; it started at the given latitude and longitude, it was reported by the user, and an application specific extension indicated that 26.2 km had been travelled.

```
{
   "Header": {"Version": [1, 0, 1, 0]},
   "Who": {"ConsumerID": "5a702670-ff63-4d1d-ba9d-077dd345ab62"},
   "What": {"Cluster": 22,"Class": 1,"SubClass": 1,"Element": 2},
   "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1433397180,"Duration":
3903},
   "Where": {"Exactness": 6,"Latitude": 51.53118159161092,"Longitude": -
0.4319647327069491},
   "How": {"How": 9},
   "Extension": {"ExtFltTag": 10003,"ExtFltValue": 26.2}
}
```

6 Security

6.1 General Technical Principles

6.1.1 Internet

SSL/TLS **[RFC5246]** SHALL be used for all internet communications within the Architecture. This creates an encrypted channel for the data (Behavioural Atoms, Report Data, Segment Data and Pseudonymous Keys) and prevents a third party from reading it in transit. It means that servers like the IDA, Data Engine and any Service Provider and Operator systems MUST use SSL/TLS certificates.

6.1.2 Pseudonymous Keys

IDA generated Pseudonymous Keys SHALL be used as the userids for the roles and actors in the Architecture. These are devoid of DIPI and unique across the Architecture. Pseudonymous Keys used as ConsumerIDs need to be handled securely and carefully since they could be mis-used to pollute the Atom collection in a Data Engine, or to retrieve data about a Consumer if a Service Provider's credentials are divulged.

6.1.3 Userids and passwords

Different userids MAY be used and different passwords SHALL be used for each service layer (e.g. for Operator with Identity Authority, Operator with Data Engine). These SHALL be encrypted when stored. Separate credentials SHOULD be used to access the Management Interface (MMI) and Query Interface (PQI), reducing the likelihood of getting access to both and retrieving Atoms for all of a Service Provider’s Consumers.

Where the Operator is a separate entity from the Service Provider, it SHOULD use BasicAuth, as a minimum, to request/return reports from its Service Provider. These reports SHALL be pseudonymised and contain no DIPI.

7 Minimal Management Interface

7.1 Introduction

This section defines the Minimal Management Interface (MMI) between a Data Engine and other roles in the Architecture. It provides an information request operation through which other actors in the Architecture discover the URLs for operations on the Data Engine. It provides operation definitions for Service Providers and Operators as follows:

- Service Provider Operations:
  - Register a new Operator;
  - Retrieve a list of existing Operators;
  - Retrieve a list of Consumers associated with a given Operator;
  - Suspend an Operator;
  - Resume an Operator;
  - Register Devices;
  - Unassign Devices; and
  - Assure a Consumer is registered with a given Operator.
- Operator Operations:
  - Register a Consumer;
  - Forget a Consumer; and
  - Associate a Device with a Consumer.

There are two important aspects of managing personal data that impact on implementations, namely the 'right to be forgotten' and the requirement for data to be accurate and up-to-date. The former is addressed by the Forget Consumer operation. For the later, it is suggested that Atoms be stored with a Reliability of less than 100%, thus allowing for later updates and corrections. However, in the event of erroneous 100% Reliable Atoms and erroneous / incomplete Segment Data, the appropriate approach is to download all data for the Consumer, forget the old Consumer, register a new Consumer with a new ConsumerID and upload the corrected data and Atoms.

7.2 COEL Minimal Management Interface Specification (MMI)

7.2.1 Authorization Protocol

To access all Service Provider operations on the Data Engine MMI API, Service Providers MUST use the BasicAuth Protocol.

To assess all Operator operations on the Data Engine MMI API, Operators MUST use the NoAuth Protocol.

7.2.2 Information Request

Every Data Engine SHALL publish its Data Engine Home URI. Performing a GET on this URI SHALL return general information about the Data Engine as a JSON object.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
GET /home	*None*	*200 (OK)*	*application/json*	*JSON object*

Elements in the response body JSON object:

Key	***Type***	***Description***	***Required***

AtomsURI	*String*	*The URI of the Atoms service encoded as a string.*	*If service implemented.*
QueryURI	*String*	*The URI of the Query service encoded as a string.*	*If service implemented.*
ManagementURI	*String*	*The URI of the Management service encoded as a string.*	*If service implemented.*
ServerTime	*Number*	*Current server time in UTC as an integer Unix time stamp.*	*Yes*
AtomsStatus	*String*	*The current status of the Atoms service encoded as a string. It MUST be one of "Up", "Down", "Not implemented" or "Unknown".*	*Yes*
QueryStatus	*String*	*The current status of the Query service encoded as a string. It MUST be one of "Up", "Down", "Not implemented" or "Unknown".*	*Yes*
ManagementStatus	*String*	*The current status of the Management service encoded as a string. It MUST be one of "Up", "Down", "Not implemented" or "Unknown".*	*Yes*
CoelSpecificationVersion	*Array of Number*	*The specification version that this data engine complies with (e.g. [1,0]).*	*No*
CoelModelVersion	*Array of Number*	*The version of the COEL Model that this data engine complies with (e.g. [1,0]).*	*No*

Example

Example request message:

```
GET /home
```

Example response message:

```
HTTP/1.1 200 OK

{"AtomsURI": "https://www.example.com/atoms",
 "QueryURI": "https://www.example.com/query",
 "ManagementURI": "https://www.example.com/management",
 "AtomsStatus": "Up",
 "QueryStatus": "Up",
 "ManagementStatus": "Up",
 "ServerTime": 1470822001,
 "CoelSpecificationVersion": [1,0],
 "CoelModelVersion": [1,0] }
```

7.2.3 Service Provider: Create New Operator

Create a new Operator within the Data Engine and associate it with the requesting Service Provider. Completion of this operation allows the Operator to register new Consumers.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

If validation of the OperatorID fails, with a 410 (Gone) error from the IDA, an error 410 (Gone) SHOULD be returned.

If the OperatorID is already in use for another Service Provider, Operator, Consumer or Device, an error 410 (Gone) SHOULD be returned.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***

POST <ManagementURI>/ service-provider/ operator	*JSON Object*	*200 (OK)*	*None*	*None*
		410 (Gone)	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
OperatorID	*String*	*A Pseudonymous Key generated by an IDA and associated with the Operator being registered.*	*Yes*
TimeStamp	*String*	*Time stamp, in DateTime format, of the OperatorID indicating when the IDA created this Pseudonymous Key.*	*Yes*
Signature	*String*	*Signature proving that an IDA created this OperatorID.*	*Yes*

Content of the response body JSON object:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the registration failed.*	*No*

Example

Example request message:

```
POST service-provider/operator

{"OperatorID": "00000000-0000-0000-0000-000000000000",
 "TimeStamp": "2011-02-14T00:00:00",
 "Signature":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA="}
```

Example response message:

```
HTTP/1.1 410 Gone

{"Reason":"Operator was not valid."}
```

7.2.4 Service Provider: Retrieve Operator List

A Service Provider uses this operation to retrieve a list of all registered Operators registered to the requesting Service Provider.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ service-provider/operators	*JSON Object Service Provider ID*	*200 (OK)*	*application/json*	*JSON Object*
		Error code	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	*Type*	*Description*	*Required*
ServiceProviderID	*String*	*A Pseudonymous Key, generated by an IDA, of the requesting Service Provider.*	*Yes*

Content of the response body JSON object for a successful request:

Key	*Type*	*Description*	*Required*
Operators	*Array of Object*	*An array of Operator objects, one for each of the Operators associated with the requesting Service Provider.*	*Yes*

Content of the JSON Operator object:

Key	*Type*	*Description*	*Required*
OperatorID	*Number*	*A Pseudonymous Key, generated by an IDA, for the requesting Service Provider.*	*Yes*
Suspended	*Boolean*	*True if the associated operator is suspended.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	*Type*	*Description*	*Required*
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/operators

{"ServiceProviderID": "00000000-0000-0000-0000-000000000000"}
```

Example response message:

```
HTTP/1.1 200 OK

{"Operators": [
   {"OperatorID": "00000000-0000-0000-0000-000000000000", "Suspended": false},
   {"OperatorID": "00000000-0000-0000-0000-000000000001", "Suspended": true},
   {"OperatorID": "00000000-0000-0000-0000-000000000002", "Suspended": true}]}
```

7.2.5 Service Provider: Retrieve Consumer List

A Service Provider uses this operation to retrieve a list of all Consumers registered to a given Operator, which is in turn registered to the requesting Service Provider.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	*Request Body*	*Response Status*	*Response Content-Type*	*Response Body*
POST <ManagementURI>/	*JSON Object*	*200 (OK)*	*application/json*	*JSON Object*
		Error code	*application/json*	*JSON Object*

service-provider/consumers

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
OperatorID	*String*	*A Pseudonymous Key generated by an IDA and associated with an Operator registered with the requesting Service Provider.*	*Yes*

Content of the response body JSON object for a successful request:

Key	***Type***	***Description***	***Required***
ConsumerIDs	*Array of String*	*An array of Pseudonymous Keys one for each of the Consumers associated with the given Operator.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/consumers

{"OperatorID": "00000000-0000-0000-0000-000000000000"}
```

Example response message:

```
HTTP/1.1 200 OK

{"ConsumerIDs": [
         "00000000-0000-0000-0000-000000000000",
         "00000000-0000-0000-0000-000000000001",
         "00000000-0000-0000-0000-000000000002"]}
```

7.2.6 Service Provider: Suspend Operator

Suspend the given Operator’s ability to create new Consumers and assign Devices. This operation has no effect on data stored for existing Consumers. The Operator SHALL still be permitted to execute a Forget Consumer operation. Operators will be shown as assured or not independent of their suspended state. Query operations will be performed independent of the Operator's suspended state.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ service-provider/suspendOperator	*JSON Object*	*200 (OK)*	*None*	*None*
		Error code	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	*Type*	*Description*	*Required*
OperatorID	*String*	*A Pseudonymous Key generated by an IDA and associated with the Operator to be suspended.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	*Type*	*Description*	*Required*
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/suspendOperator

{"OperatorID": "00000000-0000-0000-0000-000000000000"}
```

Example response message:

```
HTTP/1.1 200 OK
```

7.2.7 Service Provider: Resume Operator

Resume the given Operator’s ability to create new Consumers and assign Devices.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	*Request Body*	*Response Status*	*Response Content-Type*	*Response Body*
POST <ManagementURI>/ service-provider/resumeOperator	*JSON Object*	*200 (OK)*	*None*	*None*
		Error code	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	*Type*	*Description*	*Required*
OperatorID	*String*	*A Pseudonymous Key generated by an IDA and associated with the Operator to be resumed.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	*Type*	*Description*	*Required*
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/resumeOperator

{"OperatorID": "00000000-0000-0000-0000-000000000000"}
```

Example response message:

```
HTTP/1.1 200 OK
```

7.2.8 Service Provider: Register Devices

All Devices associated with a Service Provider are registered in advance of being assigned to a Consumer. Register Devices associates one or more Devices with Service Provider, assigns each a device type (Personal or IoT), and validates the Pseudonymous Key of the Device. A Device SHALL be registered only once. Only Operators associated with the registering Service Provider MAY assign the Device to a Consumer.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

If validation of the DeviceIDs fails, with a 410 (Gone) error from the IDA, an error 410 (Gone) SHOULD be returned.

If any of the DeviceIDs is already in use for another Service Provider, Operator, Consumer or Device, an error 410 (Gone) SHOULD be returned.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ service-provider/registerDevices	*JSON Object*	*200 (OK)* *Error code*	*None* *application/json*	*None* *JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
DeviceIDs	*Array of String*	*An array of Pseudonymous Keys associated with the Devices and generated by an IDA.*	*Yes*
TimeStamp	*String*	*Time stamp, in DateTime format, indicating when the IDA created these Pseudonymous Keys.*	*Yes*
Signature	*String*	*Signature proving that an IDA created these Pseudonymous Keys.*	*Yes*
DeviceType	*String*	*A string ("Personal" or "IoT") indicating that the Devices are personal Devices that MAY be assigned to exactly one Consumer each or IoT Devices that MAY be assigned to multiple Consumers.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/registerDevices

{"DeviceIDs": ["00000000-0000-0000-0000-000000000001",
               "00000000-0000-0000-0000-000000000002",
               "00000000-0000-0000-0000-000000000003"],
 "TimeStamp": "2011-02-14T00:00:00",
 "Signature":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA=",
"DeviceType": "Personal"}
```

Example response message:

```
HTTP/1.1 200 OK
```

7.2.9 Service Provider: Retrieve Device List

A Service Provider uses this operation to retrieve a list of all Devices registered to the requesting Service Provider.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ *service-provider/devices*	*JSON Object Service Provider ID*	*200 (OK)* *Error code*	*application/json* *application/json*	*JSON Object* *JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
ServiceProviderID	*String*	*A Pseudonymous Key, generated by an IDA, of the requesting Service Provider.*	*Yes*

Content of the response body JSON object for a successful request:

Key	***Type***	***Description***	***Required***
Devices	*Array of Object*	*An array of Device objects, one for each of the Devices registered by the requesting Service Provider.*	*Yes*

Content of the JSON Device object:

Key	***Type***	***Description***	***Required***
DeviceID	*String*	*A Pseudonymous Key generated by an IDA for the requesting Service Provider representing a Device.*	*Yes*
DeviceType	*String*	*A string, either "IoT" or "Personal" indicating the type of Device.*	*Yes*
ConsumerIDs	*Array of String*	*An array of Pseudonymous Keys containing the ConsumerID(s) of the Consumer(s) assigned to the Device. Only "IoT" Devices SHALL list multiple Consumers. The array MAY be empty, indicating that the Device has not been assigned.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***

Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/devices

{"ServiceProviderID": "00000000-0000-0000-0000-000000000000"}
```

Example response message:

```
HTTP/1.1 200 OK

{"Devices": [
   {"DeviceID": "00000000-0000-0000-0000-000000000000",
    "DeviceType": "IoT",
    "ConsumerIDs": []},
   {"DeviceID": "00000000-0000-0000-0000-000000000001",
    "DeviceType": "Personal",
    "ConsumerIDs": ["00000000-0000-0000-0000-000000000007"]},
   {"DeviceID": "00000000-0000-0000-0000-000000000002",
    "DeviceType": "IoT",
    "ConsumerIDs": ["00000000-0000-0000-0000-000000000008",
                    "00000000-0000-0000-0000-000000000009"]}]
}
```

7.2.10 Service Provider: Unassign Device

Remove all the assignments of the Device from Consumers to which it has been assigned. Note: for IoT Devices all assigned Consumers SHALL be unassigned and the Operator might need to reassign some Consumers, if for example the Operator wished to remove only one Consumer.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ *service-provider/unassignDevice*	*JSON Object*	*200 (OK)* *Error code*	*None* *application/json*	*None* *JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
DeviceID	*String*	*A Pseudonymous Key associated with the Device and generated by an IDA.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/unassignDevice

{"DeviceID": "00000000-0000-0000-0000-000000000001"}
```

Example response message:

```
HTTP/1.1 200 OK
```

7.2.11 Service Provider: Assure

This operation provides assurance that a given Consumer is associated to a given Operator and that both are associated with the requesting Service Provider.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ service-provider/assure	*JSON Object*	*200 (OK)*	*application/json*	*JSON Object*
		Error code	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
ConsumerID	*String*	*A Pseudonymous Key associated with the Consumer and generated by an IDA.*	*Yes*
OperatorID	*String*	*A Pseudonymous Key associated with the Operator and generated by an IDA.*	*Yes*

Content of the response body JSON object for a successful request:

Key	***Type***	***Description***	***Required***
Assured	*Boolean*	*A Boolean value that is true if the given Consumer and Operator are associated with each other and with the requesting Service Provider and false otherwise.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST service-provider/assure

{"ConsumerID": "00000000-0000-0000-0000-000000000001",
 "OperatorID": "00000000-0000-0000-0000-000000000002"}
```

Example response message:

```
HTTP/1.1 200 OK

{"Assured": true}
```

7.2.12 Operator: Forget Consumer

Request that all data for a Consumer associated with this Operator be forgotten by the Data Engine. This operation uses the NoAuth protocol. The Data Engine MUST confirm requests with the Service Provider associated with the requesting Operator individually before proceeding as the initial request does not require authorisation. The mechanism for confirmation is out of scope of the MMI, e.g. email confirmation. The Data Engine MAY either delete all data associated with the Consumer or render that data non-personal. The Data Engine SHOULD keep a record of which ConsumerIDs have been forgotten (for audit purposes).

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ operator/forgetConsumer	*JSON Object*	*200 (OK)*	*None*	*None*
		Error code	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
ConsumerID	*String*	*A Pseudonymous Key associated with the Consumer and generated by an IDA.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST operator/forgetConsumer

{"ConsumerID": "00000000-0000-0000-0000-000000000001"}
```

Example response message:

```
HTTP/1.1 200 OK
```

7.2.13 Operator: Create New Consumer

Create a new Consumer within the Data Engine and associate it with the given Operator. Completion of this operation allows Behavioural Atoms to be posted anonymously to the Data Engine and be associated with the given Consumer. This operation uses the NoAuth protocol. This operation is not permitted when an Operator is suspended.

The Segment Data can only be added when a new Consumer is created. If the Segment Data for the Consumer changes (e.g. permanent move to a new time zone) then the appropriate approach is to create a new profile with a new ConsumerID while retaining the old profile.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

If validation of the ConsumerID fails, with a 410 (Gone) error from the IDA, an error 410 (Gone) SHOULD be returned.

If the ConsumerID is already in use for another Operator, Consumer or Device, an error 410 (Gone) SHOULD be returned.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ operator/consumer	*JSON Object*	*200 (OK)*	*None*	*None*
		410 (Gone)	*application/json*	*JSON Object*
		Error code	*application/json*	*JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
OperatorID	*String*	*A Pseudonymous Key associated with the Operator and generated by an IDA.*	*Yes*
ConsumerID	*String*	*A Pseudonymous Key associated with the Consumer and generated by an IDA.*	*Yes*
TimeStamp	*String*	*Time stamp, in DateTime format, of the ConsumerID indicating when the IDA created this Pseudonymous Key.*	*Yes*
Signature	*String*	*Signature proving that an IDA created this ConsumerID.*	*Yes*
SegmentData	*Object*	*An OPTIONAL object containing (OPTIONALLY) residential time zone and latitude, gender, and year of birth. The field names are: ResidentTimeZone; ResidentLatitude; Gender; and YearOfBirth.*	*No*
ResidentTimeZone	*String*	*The time zone in which the Consumer generally resides, as a string indicating +/-hh:mm from UTC (e.g "-05:00").*	*No*
ResidentLatitude	*Number*	*The latitude (rounded to an integer) at which the Consumer generally resides.*	*No*
Gender	*Number*	*Integer representing the gender of the Consumer, where: 0 = not known; 1 = male; 2 = female;* *9 = not applicable.*	*No*
YearOfBirth	*Number*	*Year in which the Consumer was born as an integer.*	*No*

The Gender parameter SHALL have enumerated fields reserved for compliance with **[ISO/IEC 5218]**.

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***

Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST operator/consumer

{"OperatorID": "00000000-0000-0000-0000-000000000000",
 "ConsumerID": "00000000-0000-0000-0000-000000000000",
 "TimeStamp": "2011-02-14T00:00:00",
 "Signature":
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAA=",
"SegmentData":
   {"ResidentTimeZone": "+03:00",
    "ResidentLatitude": 51,
    "Gender": 2,
    "YearOfBirth": 1993
   }
}
```

Example response message:

```
HTTP/1.1 200 OK
```

7.2.14 Operator: Assign a Device to a Consumer

Assign a Pseudonymous Key representing a Device to a Consumer associated with the requesting Operator. All Atoms posted with this Pseudonymous Key SHALL be associated with the corresponding Consumer. Once assigned to a Consumer, a Personal Device MUST not be reassigned to another Consumer, without first being Unassigned from all Consumers. An Operator MAY assign an IoT Device to multiple Consumers. This operation uses the NoAuth protocol. This operation is not permitted when an Operator is suspended. The Device, the Operator, and the Consumer MUST already be registered with the Data Engine and associated with the same Service Provider.

If successful, an HTTP status code of 200 *OK* MUST be returned. If unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <ManagementURI>/ operator/device	*JSON Object*	*200 (OK)* *Error code*	*None* *application/json*	*None* *JSON Object*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
DeviceID	*String*	*A Pseudonymous Key associated with the Device and generated by an IDA.*	*Yes*
ConsumerID	*String*	*A Pseudonymous Key of the Operator to which the Consumer is associated.*	*Yes*

OperatorID	*String*	*A Pseudonymous Key of the user to which the Device is to be associated. The user MUST already be associated with the requesting Operator.*	*Yes*

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request message:

```
POST operator/device

{"DeviceID": "00000000-0000-0000-0000-000000000000",
 "OperatorID": "00000000-0000-0000-0000-000000000001",
 "ConsumerID": "00000000-0000-0000-0000-000000000002"
}
```

Example response message:

```
HTTP/1.1 200 OK
```

8 COEL Behavioural Atom Protocol Interface

8.1 Introduction

This section defines the Behavioural Atom Protocol Interface (BAP) of a Data Engine. It provides operation definitions on a Data Engine for the submission of COEL Behavioural Atoms for storage.

8.2 COEL Behavioural Atom Protocol Interface Specification (BAP)

8.2.1 Authorization Protocol

The Data Engine cannot authenticate the sender, since the Data Engine has no relationship with the Consumer. Therefore, the authorization protocol is NoAuth.

8.2.2 Atom POST

To add a COEL Behavioural Atom to the Data Engine, a POST operation SHALL be sent to the *AtomsURI* obtained from the Data Engine Information Request (7.2.2). 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’.

If the media type is present in the message, it SHALL be “application/json”. Atom server implementations SHALL accept message with this media type. However, they MAY reject malformed or oversized messages.

Note that the ConsumerID or DeviceID MUST have been registered by an Operator for the Atom to be stored.

The operation MUST return a HTTP status code as outlined below:

- 202 (Accepted) and an empty response body if all of the Atoms in the request body are correctly formed. If the ConsumerID or DeviceID is not registered with the Data Engine, then the Data Engine MAY discard correctly formed Atoms while still returning a code 202.
- 400 (Bad Request) if the request body does not contain valid JSON, or if one or more of the Atoms is missing REQUIRED elements or if REQUIRED fields are missing from one or more of the Atoms.
- 500 (Internal Server Error) if an internal error occurred.

If the status is not 202 (Accepted), the response message MAY contain a JSON object containing a “Reason” field encoded as a string.

If the status is not 202 (Accepted), none of the Atoms SHALL be accepted by the Data Engine. In this case, the sender MAY submit requests for each Atom individually in order that the well-formed ones can be accepted.

<u>Handling Identical Atoms</u>

The Data Engine MUST NOT store multiple copies of Identical Atoms.

<u>Handling of Certainty when DeviceID is present:</u>

When an Atom is posted from a Device (i.e. the DeviceID is present and the ConsumerID is not), a copy of the Atom is stored for each ConsumerID associated with that DeviceID. Before being stored, the Certainty value (or 100 if the Certainty value is missing) is divided by the number of ConsumerIDs associated with the DeviceID. Thus Certainty, in a stored Atom, represents the probability that the Atom is associated with this Consumer.

Method	*Request Body*	*Response Status*	*Response Content-Type*	*Response Body*
POST <AtomsURI>/	*JSON Atom or array of JSON Atoms*	*202 (Accepted)*	*None*	*None*
		400 (Bad Request)	*application/json*	*JSON Object (Reason)*
		500 (Internal Error)	*application/json*	*JSON Object (Reason)*

The content of the request body JSON object is EITHER a single COEL Behavioural Atom OR a JSON array of COEL Behavioural Atoms

Content of the response body JSON object:

Key	*Type*	*Description*	*Required*
Reason	*String*	*An OPTIONAL description of why the operation failed.*	*No*

Example

Example request message:

```
POST     <AtomsURI>/
[{
    "Header": {"Version": [1,0,1,0]},
    "Who": {"ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5"},
    "What": {"Cluster": 4,"Class": 4,"SubClass": 1,"Element": 4},
    "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1507864341,"Duration":
600},
    "Reliability": 70,
    "Where": {"Exactness": 2, "Postcode": "UB4 8FE"},
    "How": {"How": 9},
    "Context": 4
}]
```

Example response message:

```
HTTP/1.1 202 Accepted
```

Example request message with an incorrect content type:

```
POST     <AtomsURI>/
[{
    "Header": {"Version": [1,0,1,0]},
    "Who": {"ConsumerID": "f7b0ce76-30a8-4544-aa2e-9667f6228ae5"},
    "What": {"Cluster": 4,"Class": 4,"SubClass": 1,"Element": 4},
    "When": {"UTCOffset": -3600,"Accuracy": 0,"Time": 1507864341,"Duration":
600},
    "Reliability": "Seventy Percent",
    "Where": {"Exactness": 2, "Postcode": "UB4 8FE"},
    "How": {"How": 9},
    "Context": 4
}]
```

Example response message:

```
HTTP/1.1 400 Bad Request

{"Reason": "Incorrect content type"}
```

9 Public Query Interface

9.1 Introduction

This section defines the Public Query Interface (PQI) of a Data Engine. It provides operations to retrieve the Segment Data stored when a Consumer was initially registered and a general query protocol for retrieving Atoms, including support for aggregation operations (Report Data).

9.2 COEL Public Query Interface Specification (PQI)

9.2.1 Authentication and Authorisation

To access both operations on the Data Engine PQI API, Service Providers MUST use the BasicAuth Protocol.

Separate credentials SHOULD be used to access the Minimal Management Interface (section 7) and the Public Query Interface, reducing the likelihood of getting access to both and retrieving Atoms for all of a Service Provider's Consumers.

9.2.2 Query Operation

Initiate the query contained in the body of the request and return the result of the query.

There are three possible responses to a query.

1) If successful and the Data Engine choses to return the query result immediately, an HTTP status code of 200 *OK* MUST be returned and the QueryResult element included in the body of the response. If the query includes an Aggregate element, the QueryResult SHALL contain a Table element, otherwise it SHALL contain an Atoms element.
2) The Data Engine MAY chose to create a separate resource where the client can obtain the query result, if for example the query response is very large. In this case the Data Engine MUST return an HTTP status code 201 *Created* and set the "Location:" header to the URL where the QueryResult can be obtained with (a possibly paged) GET request. In this case the response MAY include the ResultCreated element. The Response to issuing a GET on the Location field, MUST be one of these three same options (200, 201 or Error). This allows the data engine to further defer the result if necessary by issuing revised Availability times.
3) Lastly, if unsuccessful, an HTTP error code SHOULD be returned and a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <QueryURI>*/query*	*JSON Object*	*200 (OK)*	*application/json*	*JSON Object (QueryResult)*
		201 (Created)	*application/json*	*JSON Object (ResultCreated)*
		Error code	*application/json*	*JSON Object (Reason)*

9.2.2.1 Request

The request body is a JSON object containing two REQUIRED elements (ConsumerID and OperatorID) and an additional two OPTIONAL elements (TimeWindow and Query). The following JSON Schema defines the structure, spelling and basic type of each element and sub-element. The request body MUST comply with this schema, and with the additional constraints specified in the remainder of this section.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "required": ["ConsumerID", "OperatorID"],

  "properties":{
    "additionalProperties":false,
    "ConsumerID"   : {"type": "string"},
    "OperatorID"   : {"type": "string"},
    "TimeWindow"     : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "StartTime":{"type":"integer"},
            "EndTime":  {"type":"integer"}}},

    "Query"     : {
        "type": "object",
        "additionalProperties":false,
        "properties":{
            "Filter":{"type":"object",
                "required": ["ColName", "Comparator", "Value"],
                "additionalProperties": false,
                "properties": {
                    "ColName":    {"type":"string"},
                    "Comparator": {"type":"string"},
                    "Value":      {"type":"string"}}},
            "AND":{"type":"array"},
            "OR": {"type":"array"},
            "NOT":{"type":"object"},

            "Aggregate":{
                "type": "object",
                "additionalProperties":false,
                "properties":{
                    "Columns": {"type":"array",
                                "items": {
                                    "type": "object",
                                    "additionalProperties":false,
                                    "required": ["ColName", "Aggregator"],
                                    "properties":{
                                        "ColName": {"type":"string"},
                                        "Aggregator": {"type":"string"}}}},
                    "GroupBy": {"type":"array"}}}}
}}
```

Constraints:

1. ConsumerID and the OperatorID are REQUIRED.
2. If the Consumer is NOT for that Operator then no data is returned.
3. The Query element contains no more than one Filter, AND, OR, or NOT element
4. An AND element contains an array of Filter, AND, OR, or NOT elements
5. An OR element contains an array of Filter, AND, OR, or NOT elements
6. A NOT element contains a exactly one Filter, AND, OR, or NOT element

Element Descriptions:

Key	***Type***	***Description***	***Required***

ConsumerID	*String*	*A Pseudonymous Key representing the requesting Consumer who is the subject of the query.*	*Yes*
OperatorID	*String*	*A Pseudonymous Key representing the Consumer's Operator.*	*Yes*
TimeWindow	*Object*	*Indicates start and end time for Atom selection.*	*No*
StartTime	*Number*	*Start of time interval to be included in the query. Time in seconds since 1/1/1970 UTC. If absent, 1/1/1970 is assumed. Atoms SHALL be included if their time stamp is between the StartTime and the EndTime.*	*No*
EndTime	*Number*	*End of time interval to be included in the query. Time in seconds since 1/1/1970 UTC. If absent, infinity is assumed. Atoms SHALL be included if their time stamp is between the StartTime and the EndTime.*	*No*
Query	*Object*	*The element describing the overall structure of the query.*	*No*
Filter	*Object*	*The element describing which Atoms to use in the query.*	*No*
ColName	*String*	*Column name.*	*No*
Comparator	*String*	*One of "=", ">", ">=", "<", "<=", "!=".*	*No*
Value	*String*	*Comparison value.*	*No*
Aggregate	*Object*	*The element describing how to aggregate values selected in the query.*	*No*
Columns	*Array of Object*	*An array indicating which columns to aggregate and how to aggregate them.*	*No*
Aggregator	*String*	*One of AVG, SUM, COUNT, MIN, MAX, STDDEV.*	*No*
ColName	*String*	*Column name.*	*No*
GroupBy	*Array of String*	*One or more Column Names to group by. Each unique combination of the values included in the GroupBy element SHALL result in a separate row in the Table element.*	*No*

9.2.2.2 Response (200)

Content of the response body JSON object when a 200 (OK) status code is returned is either:

1) An array of Atoms; or
2) A table of data (resulting from an aggregate query).

Note:

- Atoms MAY be returned either in the version in which they were originally stored or in the currently supported version.

The following JSON Schema defines the structure, spelling and basic type of each element and sub-element. The response body MUST comply with this schema, and with the additional constraints specified in the remainder of this section.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "additionalProperties":false,
  "properties":{
    "QueryResult" : {
      "type": "object",
      "additionalProperties":false,
      "properties":{
        "Atoms" : {"type": "array"},
        "Table" : {"type": "array",
                   "Row": {"type":"array",
                             "items": {
                                 "type":"object",
                                 "additionalProperties":false,
                                 "properties": {
                                     "ColName":    {"type":"string"},
                                     "Aggregator": {"type":"string"},
                                     "Value":      {"type":"number"}}}}}}}}}
```

Key	***Type***	***Description***	***Required***
QueryResult	*Object*	*If the aggregate element is not present in a query, the result contains a single Atoms element. For aggregates and grouped queries, the QueryResult contains a single Table element.*	*No*
Atoms	*Array of Object*	*Array of Atoms as described in section 5.2. If a projection is specified only requested fields of the matching Atoms are included.*	*No*
Table	*Array of Object*	*Array of Rows.*	*No*
Row	*Array of Object*	*Each element in the array represents a cell of the result table, including a ColName, Value, and (if appropriate) Aggregator property.*	*No*
ColName	*String*	*The column name used in the query.*	*No*
Aggregator	*String*	*The aggregation function used in the query.*	*No*
Value	*Number*	*The resulting value.*	*No*

9.2.2.3 Response (201)

Content of the response body JSON object when a 201 (Created) status code is returned MUST conform to the following schema:

```
{ "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "additionalProperties":false,
  "properties":{
    "ResultCreated" : {
      "type": "object",
      "additionalProperties":false,
      "properties":{
        "Size"   : {"type": "integer"},
        "Location"   : {"type": "string"},
        "AvailableFrom"   : {"type": "integer"},
        "AvailableUntil"   : {"type": "integer"}}}}}
```

Example:

```
{"ResultCreated": {
         "Size": 1000,
         "Location": "http://...",
         "AvailableFrom": 1234,
         "AvailableUntil": 2345}}
```

Key	***Type***	***Description***	***Required***

ResultCreated	*Object*	*This element describes the query result's size, availability and location. The fields are Size, Location, AvailableFrom and AvailableUntil.*	*No*
Size	*Number*	*The expected size in bytes of the QueryResult object as an integer.*	*No*
Location	*String*	*The location (MUST be the same as in the Location: header) where the QueryResult can be obtained.*	*No*
AvailableFrom	*Number*	*Time from which the QueryResult can be obtained, presented in integer seconds since 1970/01/01 00:00Z (Unix time stamp in UTC).*	*No*
AvailableUntil	*Number*	*Time until which the QueryResult can be obtained, presented in integer seconds since 1970/01/01 00:00Z (Unix time stamp in UTC).*	*No*

9.2.2.4 Response (Error)

Content of the response body JSON object in the case of an error:

Example

```
{"Reason": "Wrong Operator for this Consumer"}
```

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

9.2.2.5 Column Names

The following table contains the column names to be used in queries. These correspond to the field values of the Atoms posted to the Data Engine.

Name	***Data Type***
HEADER_VERSION	*[Number, Number, Number, Number]*
WHEN_UTCOFFSET	*Number*
WHEN_ACCURACY	*Number*
WHEN_DURATION	*Number*
WHAT_CLUSTER	*Number*
WHAT_CLASS	*Number*
WHAT_SUBCLASS	*Number*
WHAT_ELEMENT	*Number*
HOW_HOW	*Number*
HOW_CERTAINTY	*Number*
HOW_RELIABILITY	*Number*
CONTEXT_SOCIAL	*Number*
CONTEXT_WEATHER	*Number*
CONTEXT_CONTEXTTAG	*Number*
CONTEXT_CONTEXTVALUE	*Number*
WHERE_EXACTNESS	*Number*
WHERE_LATITUDE	*Number*
WHERE_LONGITUDE	*Number*

WHERE_W3W	*String*
WHERE_PLACE	*Number*
WHERE_POSTCODE	*String*
CONSENT_ JURISDICTION	*String*
CONSENT_DATE	*Number*
CONSENT_RETENTIONPERIOD	*Number*
CONSENT_PURPOSE	*Number*
CONSENT_POLICYURL	*String*
CONSENT_RECORDID	*String*
CONSENT_RECORDSERVICE	*String*
EXTENSION_INTTAG	*Number*
EXTENSION_INTVALUE	*Number*
EXTENSION_FLTTAG	*Number*
EXTENSION_FLTVALUE	*Number*
EXTENSION_STRTAG	*Number*
EXTENSION_STRVALUE	*String*

Note that inclusion of HEADER_VERSION in a query filter clause affects the selection of data but does not affect the version of any Atoms returned by the query. The version of the result MAY be that of the current version supported by the Data Engine or that of the Atoms as originally stored. Except for WHERE_LATITUDE, WHERE_LONGITUDE, and EXTENSION_FLTVALUE, which MAY be decimals, all Number values are assumed to be integers.

Examples

Example request Query for an Atoms query.

```
POST query

{"ConsumerID" : "ed58fc40-a866-11e4-bcd8-0800200c9a66",
 "Timewindow" : {
    "StartTime" : 1415145600,
    "EndTime" : 1415232000
   }
}
```

Corresponding response message:

```
HTTP 1.1 200 OK

{"QueryResult": {
  "Atoms": {
    [{"Header":{ "Version":[1,0,1,0]},
      "Who":{ "ConsumerID": "5a702670-ff63-4d1d-ba9d-077dd345ab62"},
      "What":{ "Cluster":22, "Class":1, "SubClass":1, "Element":2},
      "When":{  "Time":1433397180, "Duration":3903}},
     {"Header":{ "Version":[1,0,1,0]},
      "Who":{ "ConsumerID": "5a702670-ff63-4d1d-ba9d-077dd345ab62"},
      "What":{ "Cluster":22, "Class":1, "SubClass":1, "Element":2},
      "When":{  "Time":1433397240, "Duration":2705}}
    ]}}
```

Example request Query for an aggregate/group-by query.

```
POST query
```

```
{"ConsumerID" : "ed58fc40-a866-11e4-bcd8-0800200c9a66",
 "Timewindow" : {
    "StartTime" : 1415145600,
    "EndTime" : 1415232000
   }
 "Query": {
    "Aggregate": {
        "Columns": [
            {"ColName": "WHEN_DURATION",
             "Aggregator": "SUM"},
            {"ColName": "HOW_RELIABILITY",
             "Aggregator": "AVG"}],
        "GroupBy": ["WHAT_ELEMENT","WHAT_SUBCLASS"]}
    "Project" : {
        "Include": ["WHAT_CLUSTER", "WHAT_CLASS"]
    }
}
```

Corresponding response message:

```
HTTP 1.1 200 OK

{"QueryResult": {
    "Table": [
        [ {"ColName": "WHEN_DURATION",  "Aggregator": "SUM","Value": 127.3},
          {"ColName": "HOW_RELIABILITY","Aggregator": "AVG","Value": 83},
          {"ColName": "WHAT_CLUSTER",   "Value": 12},
          {"ColName": "WHAT_CLASS",     "Value": 23},
          {"ColName": "WHAT_ELEMENT",   "Value": 2031},
          {"ColName": "WHAT_SUBCLASS",  "Value": 1256}
        ],
        [ {"ColName": "WHEN_DURATION",  "Aggregator": "SUM","Value": 993},
          {"ColName": "HOW_RELIABILITY","Aggregator": "AVG","Value": 12},
          {"ColName": "WHAT_CLUSTER",   "Value": 12},
          {"ColName": "WHAT_CLASS",     "Value": 23},
          {"ColName": "WHAT_ELEMENT",   "Value": 2037},
          {"ColName": "WHAT_SUBCLASS",  "Value": 1334}
        ]
      ] }}
```

9.2.3 Segment Data

Request Segment Data for a Consumer.

If successful, an HTTP status code of 200 *OK* MUST be returned along with the Segment Data. If unsuccessful, an HTTP error code SHOULD be returned, in which case a JSON object MAY be returned providing some explanation of the failure.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <QueryURI>*/segment*	*JSON Object*	*200 (OK)*	*application/json*	*JSON Object (SegmentData)*
		Error code	*application/json*	*JSON Object (Reason)*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***

ConsumerID	*String*	*A Pseudonymous Key representing the requesting Consumer who is the subject of the segment data request.*	*Yes*
OperatorID	*String*	*A Pseudonymous Key representing the Consumer's Operator.*	*Yes*

Content of the response body JSON object when a 200 (OK) status code is returned:

Key	***Type***	***Description***	***Required***
SegmentData	*Object*	*An object containing residential time zone and latitude, gender, and year of birth. The field names are ResidentTimeZone, ResidentLatitude, Gender and YearOfBirth.*	*No*
ResidentTimeZone	*String*	*The time zone in which the Consumer generally resides, as a string indicating +/-hh:mm from UTC (e.g "-05:00").*	*No*
ResidentLatitude	*Number*	*The latitude (rounded to an integer) at which the Consumer generally resides.*	*No*
Gender	*Number*	*Integer representing the gender of the Consumer, where:* *0 = not known; 1 = male; 2 = female;* *9 = not applicable.*	*No*
YearOfBirth	*Number*	*Year in which the Consumer was born as an integer.*	*No*

The Gender parameter SHALL have enumerated fields reserved for compliance with **[ISO/IEC 5218]**.

Content of the response body JSON object in the case of an error:

Key	***Type***	***Description***	***Required***
Reason	*String*	*An OPTIONAL description of why the request failed.*	*No*

Example

Example request messages:

```
POST segment

{"ConsumerID" : "ed58fc40-a866-11e4-bcd8-0800200c9a66",
 "OperatorID" : "fd58dc41-a856-31d4-5558-6534237776ac"}
```

Example response message:

```
HTTP/1.1 200 OK

{"SegmentData":
   {"ResidentTimeZone": "+03:00",
    "ResidentLatitude": 51,
    "Gender": 2,
    "YearOfBirth": 1993
   }
}
```

10 Identity Authority Interface

10.1 Introduction

This section defines how an Identity Authority Interface (IDA) is used to generate and subsequently validate digitally signed unique Pseudonymous Keys.

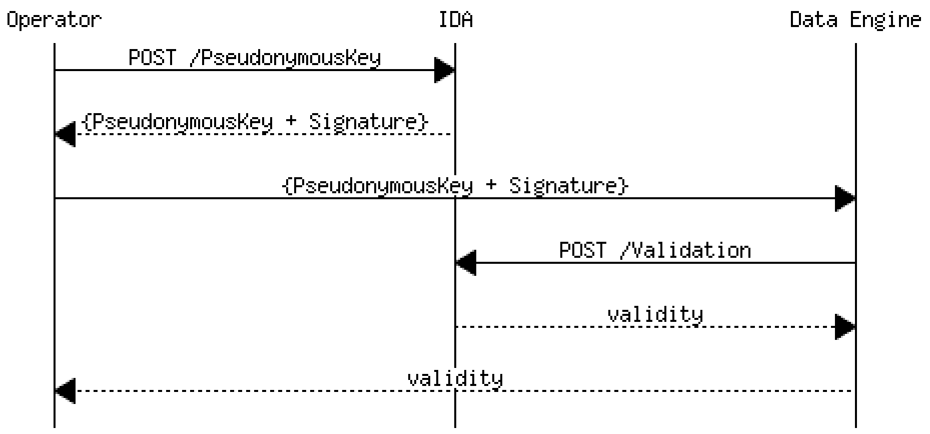


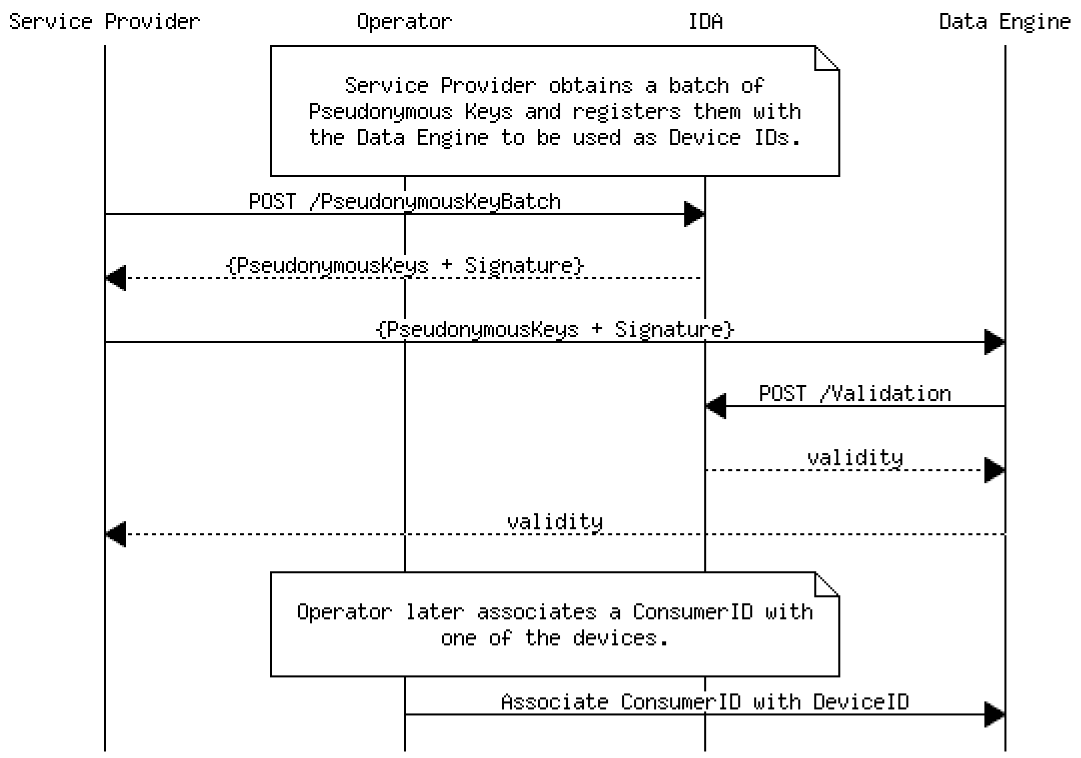
Figure 3 : IDA / Data Engine signup sequence

Figure 3 shows the sequence an Operator follows in signing up a new Consumer: obtain a Pseudonymous Key from IDA and then use it to signup with the Data Engine.

The signature is used so that the Data Engine can be assured that the Pseudonymous Key is genuine. Rather than using asymmetric key-pairs and distributing a public key and signing algorithm, the IDA provides the means for a receiver of a signed Pseudonymous Key to validate its signature.

It is assumed that this transaction is short – Operators only request Pseudonymous Keys when they are needed and register them shortly afterwards (ideally within minutes). The Identity Authority needs to be free to alter the means of signature (if for example it believes the mechanism used internally has been compromised). If this change happens during a transaction then validation SHALL fail. This is an unlikely event, but parties in the transaction need to be able to manage it:

- Data Engines receiving a failed validation code from the IDA pass the failure back to the Operator (see MMI: create new Operator).
- Operators receiving a failed validation code from the Data Engine discard the Pseudonymous Key and request a new one from the IDA.
- If the second attempt also fails, the Operator SHOULD try once more after a short delay (1-2 seconds) before aborting the attempt to register.



Service Provider
Operator
IDA
Data Engine
Service Provider obtains a batch of Pseudonymous Keys and registers them with the Data Engine to be used as Device IDs.
POST /PseudonymousKeyBatch
{PseudonymousKeys + Signature}
{PseudonymousKeys + Signature}
POST /Validation
validity
validity
Operator later associates a ConsumerID with one of the devices.
Associate ConsumerID with DeviceID

Figure 4 : Service Provider registering a batch of DeviceIDs

The IDA also provides a means to generate a batch of up to 1000 Pseudonymous Keys in one request as shown in Figure 4. The batch contains a single signature and the same protocol is followed for validation: The Service Provider passes the batch to the Data Engine which validates the batch with the IDA. It is expected that the Service Provider then provides this batch of IDs to a Hardware Developer to be used in Devices (section 7.2.8). Pseudonymous Keys are primarily intended to represent a Consumer in the Architecture. However, Operators and Service Providers also require keys to identify themselves in their machine to machine interactions. IDA generated Pseudonymous Keys can be used for this purpose since they are devoid of DIPI and unique across the Architecture.

The IDA cannot guarantee that a Pseudonymous Key is unique for all time. Although extremely unlikely that duplicate Pseudonymous Keys will be generated, the situation is handled within the Ecosystem. The signup sequence above allows a Data Engine to reject a duplicate identifier and request a new one. If duplicate identifiers are registered in separate Data Engines, this clash will only become evident if the data for one of the Consumers is transferred into the other Data Engine. In this case the clash will be dealt with by the Service Provider as part of the data migration, by allocating a new ConsumerID.

10.2 COEL Identity Authority Interface Specification (IDA)

The Identity Authority (IDA) API provides a means for Operators and/or Service Providers to generate unique Pseudonymous Keys for Consumers or Devices. A Pseudonymous Key is REQUIRED when an Operator or Service Provider registers a Consumer or Device with a Data Engine. Pseudonymous Keys are digitally signed so that Data Engines can validate them to ensure they were generated by the Identity Authority and have not been altered.

10.2.1 Authentication and Authorisation

With the exception of the IDA Information Request, callers MUST use the BasicAuth Protocol for all interactions with the IDA.

The IDA information request requires only the NoAuth Protocol.

Where BasicAuth is being used, each userid MUST be assigned to one of the following two roles in the IDA:

- *Generator:* Allowing the caller to generate Pseudonymous Keys
- *Validator*: Allowing the caller to validate Pseudonymous Keys

If the userid is unknown to the IDA, or the wrong password is supplied a HTTP status code *401 Invalid username or password* SHALL be returned**.**

If a request is made with a userid that is assigned a role that is not authorized to perform that action then the HTTP status code *403 Unauthorised* SHALL be returned.

An Identity Authority MUST NOT retain any DIPI relating to Consumers or Operators.

10.2.2 Information Request

An Identity Authority SHALL publish its Home URI. Performing a GET on this URI SHALL return general information about the Identity Authority as JSON object.

Method	***Request***	***Response Status***	***Response Content-Type***	***Response Body***
GET /home	*None*	*200 (OK)*	*application/json*	*JSON object*

Content of the returned JSON Object:

Key	***Type***	***Description***	***Required***
IdentityAuthorityURI	*String*	*The URI of the Identity Authority service encoded as a string.*	*Yes*
ServerTime	*Number*	*Current server time in UTC as an integer Unix time stamp.*	*Yes*
IdentityAuthorityStatus	*String*	*The current status of the Identity Authority service encoded as a string. It MUST be one of "Up", "Down", or "Unknown".*	*Yes*
CoelSpecificationVersion	*Array of Number*	*The specification version that this IDA complies with (e.g. [1,0]).*	*No*

Example

Example request message:

```
GET /home
```

Example response message:

```
HTTP/1.1 200 OK

{"IdentityAuthorityURI": "https://www.ida.com/api",
 "IdentityAuthorityStatus": "Up",
 "ServerTime": 1470822001,
 "CoelSpecificationVersion": [1,0] }
```

10.2.3 PseudonymousKey endpoint

The IDA SHALL provide a PseudonymousKey end-point which provides the means to generate Pseudonymous Keys for users whose API Credentials have the Generator role.

The Identity Authority MAY periodically change the mechanism used to sign the response, so the response SHOULD be passed to the Data Engine shortly after generation or validation can fail.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <IdentityAuthorityURI> */PseudonymousKey*	*None*	*200 (OK)*	*application/json*	*JSON*
		Any other status	*None*	*None*

Content of the response body JSON object:

Name	***Value***	***Description***	***REQUIRED***
PseudonymousKey	*String*	*A Pseudonymous Key generated by the IDA.*	*Yes*
TimeStamp	*String*	*Time stamp, in DateTime format, indicating when the IDA created this response.*	*Yes*
Signature	*String*	*Signature proving that the IDA created this response.*	*Yes*

Status:

200: The operation was successful

401/403: The operation failed due to authentication or authorization failure. The caller SHOULD confirm its credentials and retry.

500: Internal error, the caller SHOULD retry

Example

Example request message:

```
POST /PseudonymousKey
```

Example response message

```
HTTP/1.1 200 OK

{
  "PseudonymousKey": "00000000-0000-0000-0000-000000000000",
  "TimeStamp": "2011-02-14T00:00:00",
  "Signature": "SGFDXCTVIVVIFUJUVUYBKYKJHBK=="
}
```

10.2.4 PseudonymousKeyBatch endpoint

The Identity Authority SHALL provide a PseudonymousKeyBatch end-point which provides the means to generate a batch of Pseudonymous Keys in one response packet for users whose API Credentials have the Generator role. The request body SHALL contain one parameter: Size. The response SHALL contain three parameters: An array of Pseudonymous Keys; the time stamp at which the response was generated; and a signature that can be used for validation. The IDA SHALL be capable of generating batches of up to 1000 Pseudonymous Keys in each request.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <IdentityAuthorityURI> /PseudonymousKeyBatch	*JSON*	*200 (OK)*	*application/json*	*JSON*
		Any other status	*None*	*None*

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
Size	*Number*	*The number of Pseudonymous Keys to return in the batch (integer 1<=N<=1000).*	*Yes*

Content of the response body JSON object:

Key	***Type***	***Description***	***Required***
PseudonymousKeys	*Array of String*	*Array of Pseudonymous Keys that can be used to represent Devices in the Architecture.*	*Yes*
TimeStamp	*String*	*Time stamp, in DateTime format, indicating when the IDA created this response.*	*Yes*
Signature	*String*	*Signature proving that an IDA created this response.*	*Yes*

Status codes:

200: The operation was successful

400: The operation failed due to the request body being malformed or the size being out of range [1..1000]

401/403: The operation failed due to authentication or authorization failure. The caller SHOULD confirm its credentials and retry.

500: Internal error, the caller SHOULD retry

Example

Example request message

```
POST /PseudonymousKeyBatch

{"Size": 3}
```

Corresponding example response message:

```
HTTP/1.1 200 OK

{
  "PseudonymousKeys": [
      "00000000-0000-0000-0000-000000000000",
      "00000000-0000-0000-0000-000000000001",
      "00000000-0000-0000-0000-000000000002"]
  "TimeStamp": "2011-02-14T00:00:00",
  "Signature": "SGFDXCTVIVVIFUJUVUYBKYKJHBK=="
}
```

Example request message

```
POST /PseudonymousKeyBatch

{"Size": -1}
```

Corresponding example response message:

```
HTTP/1.1 400 Bad Request
```

10.2.5 Validation endpoint

The Identity Authority SHALL provide a validation end-point which provides the means to validate a signed Pseudonymous Key or a signed batch of Pseudonymous Keys for users whose API Credentials have the Validator role.

Method	***Request Body***	***Response Status***	***Response Content-Type***	***Response Body***
POST <IdentityAuthorityURI> /Validation	*JSON*	*200 (OK)*	*None*	*None*
		Any other status	*application/json*	*JSON*

The request body is formatted as either a /PseudonymousKey response packet or a /PseudonymousKeyBatch response packet. The fields from the IDA response MUST not be modified or validation SHALL fail.

Content of the request body JSON object:

Key	***Type***	***Description***	***Required***
PseudonymousKey	*String*	*A Pseudonymous Key generated by the IDA.*	*Exactly one of these is REQUIRED*
PseudonymousKeys	*Array of String*	*Array of Pseudonymous Keys generate by the IDA*	
TimeStamp	*String*	*Time stamp, in DateTime format, from the original IDA response.*	*Yes*
Signature	*String*	*Signature from the original IDA response.*	*Yes*

Content of the response body JSON object:

Key	***Type***	***Description***	***Required***
Reason	*String*	*OPTIONAL description of why the operation failed*	*No*

Status:

200: The operation was successful. The Pseudonymous Key or batch of Pseudonymous Keys is valid.

410: The operation was successful (the request was properly formed and authorized) but the Pseudonymous Key or batch of Pseudonymous Keys is no longer valid.

400: The operation failed due to the request body being malformed.

401/403: The operation failed due to authentication or authorization failure. The caller SHOULD confirm its credentials and retry.

500: Internal error, the caller SHOULD retry

Example

Example request message

```
POST /Validation

{
  "PseudonymousKey": "00000000-0000-0000-0000-000000000000",
  "TimeStamp": "2011-02-14T00:00:00",
  "Signature": "SGFDXCTVIVVIFUJUVUYBKYKJHBK=="
}
```

Corresponding example response:

```
HTTP/1.1 200 OK
```

Example request message

```
POST /Validation

{
  "PseudonymousKeys": [
      "00000000-0000-0000-0000-000000000000",
      "00000000-0000-0000-0000-000000000001",
      "00000000-0000-0000-0000-000000000002"],
  "TimeStamp": "2011-02-14T00:00:00",
  "Signature": "SGFDXCTVIVVIFUJUVUYBKYKJHBK=="
}
```

Corresponding example response:

```
HTTP/1.1 410 Gone

{"Reason": "IDA could not validate these keys"}
```

Example request message

```
POST /Validation

{
  "PseudonymousKeys": [
      "00000000-0000-0000-0000-000000000000",
      "00000000-0000-0000-0000-000000000001",
      "00000000-0000-0000-0000-000000000002"],
}
```

Corresponding example response:

```
HTTP/1.1 400 OK

{"Reason": "The input was missing mandatory elements"}
```

11 Privacy-by-Design Implementations (non-normative)

11.1 Introduction

This section describes how the normative elements of the COEL Specification can be configured to achieve a privacy-by-design implementation within an Ecosystem managed by a single IDA. It sets out principles of operation, controls the roles that actors can perform and gives detailed requirements of the responsibilities of actors.

11.2 Principles

11.2.1 Data Separation Principle (P1)

The COEL Specification implements a separation of data types for specific roles and this principle extend this to the actors. Data Engines keep data on *what* Consumers do (COEL Behavioural Atoms) and the Operator keeps data on *who* Consumers are (DIPI). No single organisation holds both sets of data together. This means that it would need a double accidental or malicious disclosure for connected information to be released.

11.2.2 Data Atomisation Principle (P2)

Data is deliberately broken down into small chunks of information by the Operator and coded with the Consumer's ConsumerID, thus each separate COEL Behavioural Atom has a very low privacy risk.

11.2.3 Atomised Consent Principle (P3)

Consumers give informed consent to the Operator or are provided appropriate notice under the requirements set out in this section. This allows the Operator to sign up the Consumer with a ConsumerID. This ConsumerID is the indicator to Identity Authority and other Ecosystem actors that appropriate consent or notice is in place. The requirement for every COEL Behavioural Atom to have a ConsumerID (or associated DeviceID) in combination with the detailed consent fields, ensures each Atom has that Consumer's consent / notice written into the structure of the data. The time stamp uniquely associated with each COEL Behavioural Atom allows full auditing of this principle.

11.2.4 Separation of Competence Principle (P4)

Data Engines are expert data handlers. They know how to run robust, secure and always on cloud based data services; they handle COEL Behavioural Atoms not Consumers. Service Providers are expert at manipulating behavioural data to deliver services and service content; they handle COEL Behavioural Atoms not Consumers. Operators are experts at Consumer facing services and handling DIPI; they handle Consumers not COEL Behavioural Atoms. The Identity Authority is expert at overseeing the Ecosystem.

11.2.5 No Conflict of Interest Principle (P5)

Consumers need to see that there are no conflicts around their data. To ensure this, the Identity Authority acts on behalf of the Consumer in partnership with Operator, Service Provider, Data Engine and regulators.

11.2.6 Active Support Principle (P6)

All actors will actively promote these principles, safeguard the structure of the Ecosystem and support good data practice for both individuals and enterprises.

11.2.7 Transparency Principle (P7)

The roles and identities of all the actors in the Ecosystem who are working together on behalf of a Consumer will be clear and visible to that Consumer.

11.3 Actors' Responsibilities

The roles will be performed by a number of actors that create the Ecosystem. Actors can have multiple roles but certain combinations are not permissible as described in the requirements and the table below. The table shows all the possible roles an actor can have (✓ = role that an actor can take on; ✗ = role that an actor cannot take on).

		Role						
		Identity Authority	*Data Engine*	*Service Provider*	*Operator*	*Consumer*	*Technical Service Developer*	*Hardware Developer*
Actor	*Identity Authority*	✓	✗	✗	✗	✗	✗	✗
	Data Engine	✗	✓	✗	✗	✗	✓	✓
	Service Provider	✗	✗	✓	✓	✗	✓	✓
	Operator	✗	✗	✓	✓	✗	✓	✓
	Consumer	✗	✓*	✗	✗	✓	✓	✓
	Technical Service Developer	✗	✗	✗	✗	✗	✓	✓
	Hardware Developer	✗	✗	✗	✗	✗	✓	✓

* In the specific circumstance where the Data Engine role is fulfilled by a personal data store, the Consumer will also be the Data Engine.

11.3.1 Identity Authority

Requirement	***Guiding principles & notes***

will	*Maintain an IDA service with good availability and timeliness*	*P4*
	Provide its services on a fair, reasonable and non-discriminatory basis	*P5*
	Provide Consumers with information about the operation of the Ecosystem free of charge	*P5 & P7*
will not	*Take on any other role in the Ecosystem (other than for the purposes of providing a limited 'sandbox' test environment)*	*P4 & P5*
	Gain profit or commercial advantage through its role in the Ecosystem	*P5*
	Store COEL Behavioural Atoms	*P4 & P5*
	Hold any Consumer's Directly Identifying Personal Information (DIPI)	*P5*
can	*Request Data Engine support to deliver population-level insights for public information and the purposes of marketing the COEL Specification*	*P6*
	Make a query on Data Engines to ensure a specific ConsumerID has been forgotten	*P7 This allows the Identity Authority to audit the forgetting process*
	Provide Consumers with information about their status within the Ecosystem, i.e. 'known' or 'forgotten' and only by ConsumerID and not DIPI	*P5 & P7*
	Provide audit services to Data Engine, Service Provider, Operator and regulators	*P6*

11.3.2 Data Engine

Requirement		***Guiding principles & notes***
will	*Provide secure storage of COEL Behavioural Atoms for a period to be agreed with the Service Provider in line with the Consumer consent or notice*	*P2 & P3*
	Provide minimal interface services for Service Providers to process joiners, movers, and leavers (e.g. Operator & Consumer & Device trees, registration, unique Pseudonymous Key re-allocation, forgetting)	*P4*
	Provide minimal interface services for querying COEL Behavioural Atoms by registered Service Providers	*P1*
	Maintain an entry point for uploading COEL Behavioural Atoms to the Data Engine with good availability & timeliness	*P4*
	Provide information to the Service Provider about the location and security of the infrastructure used in the delivery of services	*P7*
will not	*Link COEL Behavioural Atom data to Directly Identifying Personal Information (DIPI) from external sources*	*P1*
	Link COEL Behavioural Atom data directly to external data storage if such link might directly identify Consumers	*P1*

	Hold any Consumer's Directly Identifying Personal Information (DIPI)	*P1*
	Act as a Service Provider or Operator itself	*P1 & P4*
	Request more than the Segment Data as defined in the COEL Specification (gender, year of birth, time zone & latitude to 0 decimal points) on registration of a Consumer	*P1*
	Knowingly receive DIPI	*P1*
	Levy unreasonably punitive charges for the complete download of stored COEL Behavioural Atoms	*Supports good data rights practice and an open, competitive Ecosystem*
	Utilise IDA unique Pseudonymous Keys outside of the Ecosystem	*P1*
can	*Add non-personal data to the Atom store to deliver enhanced services (e.g. local weather data)*	*P1 While COEL Behavioural Atoms cannot be linked out, additional information can be linked in*
	Use suitable aggregation techniques rendering the data non-personal to provide indirect services to parties other than contracted Service Providers	*P1 & P6*
	Act as a Technical Service Developer to create software infrastructure that is then run by one of their Service Providers	*P5*
	Host multiple Service Providers	

11.3.3 Service Provider

Requirement		***Guiding principles & notes***
will	*Ensure that their Operators have the minimum standard consent or notice with Consumers*	*P3*
	Ensure that their Operators have secured additional consent (or equivalent due process) from Consumers when sending personal information outside the Ecosystem	*P3 & P6*
	When sending COEL Behavioural Atom information outside the Ecosystem, remove the ConsumerID	*P6 This ensures that information that has left the Ecosystem can be clearly identified*
	Ensure that their Operators follow the full COEL Specification	*P6*
	For any one service and at any one time, have only one Data Engine	*Avoids potential data loss for the Consumer and ensures the complete audit map of the Ecosystem*
	On a request from an Operator for a Consumer, supply all Segment Data, COEL Behavioural Atoms and any stored Report Data	*P2 Basic tenet of good data rights practice*
	On a request from an Operator for a Consumer to be forgotten, instruct their Data Engine to remove or render data to be non-personal	*P2 & P3*
	On a request from an Operator, provide the identity of the Data Engine	*P7*

	Notify Operators of any mergers and acquisitions or other changes that would result in a change of control over the Consumers' data	*P7*
	Check the credentials of an Operator every time a request is made to release data for a ConsumerID	*Security*
	Ensure that all Operators within a specific embodiment are working under equivalent terms (e.g. consent, purpose, retention periods etc.)	*P7*
	Use different passwords to interact with different actors in the Ecosystem (within the same Service Embodiment)	*Security*
	Use a different ServiceProviderID for every instance of a Service Embodiment in which they are an actor	*Security*
	Hold ConsumerIDs with the same security level as DIPI	*Security*
	Provide a secure interface to Operators such that communication is done in an appropriate manner with basic authentication as a minimum	*Security*
	Allocate new ConsumerIDs in the event that a clash is encountered when transferring data between Data Engines.	*Data Integrity: Deal with the rare occurrence of a clash in Pseudonymous Keys when merging two stores of Atoms.*
Will not	*Receive COEL Behavioural Atoms or DIPI directly*	*P1*
can	*Transfer its operations between Data Engines*	*Supports open, competitive Ecosystem*
	Host multiple Operators	

An Associated Service Provider is an actor that has been permissioned access to data collected by another Service Provider to provide a service to the existing Operator or Service Provider. An Associated Service Provider has no right to grant a third-party any access to the data held by the original Service Provider. All of the technical requirements on a Service Provider above will apply to an Associated Service Provider except for Consumer requests to access or modify data held by the Data Engine which will be passed to the original Service Provider that collected the data.

11.3.4 Operator

Requirement		***Guiding principles & notes***
will	*Ensure the minimum standard of consent or notice with Consumers when allocating a ConsumerID or DeviceID*	*P3*
	Secure additional consent (or equivalent due process) from Consumers when sending personal information outside the Ecosystem	*P3 & P6*
	When sending COEL Behavioural Atom information outside the Ecosystem, remove the ConsumerID and replace with DIPI	*P6 This ensures that information that has left the Ecosystem can be clearly identified*
	On a request from a Consumer, supply all DIPI, Segment Data, COEL Behavioural Atoms and any stored Report Data	*P2* *Basic tenet of good data rights practice*

	On a request from a Consumer to be forgotten, remove or render DIPI to be non-personal	*Basic tenet of good data rights practice*
	Provide a mechanism for the Consumer to access their ConsumerID	*P7 This allows the Identity Authority to audit the 'forgetting' process*
	For any one purpose and at any one time, have only one Service Provider	*Avoids potential data loss for the Consumer and ensure the complete audit map of the Ecosystem*
	Clearly identify the Service Provider and Data Engine to the Consumer	*P7*
	Notify Consumers of any mergers and acquisitions or other changes that would result in a change of control over the Consumers' data	*P7*
	Hold ConsumerIDs with the same security level as DIPI	*Security*
	Use different passwords to interact with different actors in the Ecosystem (within the same Service Embodiment)	*Security*
	Use a different OperatorID for every instance of a Service Embodiment in which they are an actor	*Security*
will not	*Store COEL Behavioural Atoms other than for the purposes of transmission to the Data Engine*	*P1*
	Send DIPI to a Data Engine	*P1*
	Share DIPI with another Operator or Service Provider without additional consent (or equivalent due process) from the Consumer	*P3*
	Utilise IDA unique Pseudonymous Keys outside of the Ecosystem	*P1*
can	*Host multiple Consumers*	

11.3.5 Consumer

Requirement		***Guiding principles & notes***
can	*Request to be 'forgotten' in the Ecosystem*	*Basic tenet of good data rights practice*
	Request the Identity Authority to audit their status in the Ecosystem	*P5 & P7*
	Request the Operator to supply their DIPI, Segment Data, Behavioural Atoms and Report Data	*Basic tenet of good data rights practice*

12 Identity Management (non-normative)

The COEL Specification provides tools for the collection and processing of the Behavioural Data of individuals and therefore identity management will be essential to any overall system implementation.

The COEL Specification provides a unique Pseudonymous Key that links all data for a specific Consumer. There is no requirement for a Consumer to have only one unique Pseudonymous Key and so, in true identity terms, a unique Pseudonymous Key links data for a profile. The ability of Consumer to maintain multiple profiles is an important method by which they can control their privacy. Outside the scope of the COEL Specification, multiple unique Pseudonymous Keys (profiles) could be maintained formally or informally by a Consumer to control their personal data and visibility of combinations of their data to any Service Provider.

The unique Pseudonymous Key is a private subject key where the Operator has the responsibility to validate, assure and authenticate the identity of the Consumer to a level appropriate to the application. There is no requirement for the Consumer and Operate to initiate their relationship with a strong identity negotiation. The unique Pseudonymous Key provides a reference for the collection of data under an ‘assertion of sameness’ principle that could later lead to a strong identity negotiation.

13 Conformance

13.1 Conformance Targets

Sections 4, 5, 6, 7, 8, 9 and 10 contain the following implementations that are subject to compliance:

COEL Model
COEL Behavioural Atom
COEL Minimal Management Interface (MMI)
COEL Behavioural Atom Protocol Interface (BAP)
COEL Public Query Interface (PQI)
COEL Identity Authority Interface (IDA)

13.2 Conformance Clause 1: COEL Model

A data object conforms to this specification as COEL Model if it satisfies all the statements below:

(a) It is valid according to the structure and rules defined in section 4.2 "COEL Model Specification".

13.3 Conformance Clause 2: COEL Behavioural Atom

A data object conforms to this specification as COEL Behavioural Atom if it satisfies all the statements below:

(a) It is valid according to the structure and rules defined in section 5.2 "COEL Behavioural Atom Specification".

13.4 Conformance Clause 3: COEL Minimal Management Interface

A Data Engine process or program conforms to this specification as COEL Minimal Management Interface if it satisfies all the statements below:

(a) It can parse and process the functions defined in section 7.2 "COEL Minimal Management Interface Specification (MMI)" according to their rules and semantics.
(b) It generates errors as REQUIRED in error cases described in section 7.2.
(c) It complies with the security requirements in section 6.1 "General Technical Principles".

13.5 Conformance Clause 4: COEL Behavioural Atom Protocol Interface

A Service Provider process or program conforms to this specification as COEL Behavioural Atom Protocol Interface if it satisfies all the statements below:

(a) It classifies events with the COEL Model as defined in Clause 1: COEL Model.
(b) It can correctly form COEL Behavioural Atoms as defined in Clause 2: COEL Behavioural Atom.
(c) It can transmit or transfer COEL Behavioural Atoms as defined in section 8.2 "COEL Behavioural Atom Protocol Interface Specification (BAP)".
(d) It complies with the security requirements in section 6.1 "General Technical Principles".

A Data Engine process or program conforms to this specification as COEL Behavioural Atom Protocol Interface if it satisfies all the statements below:

(e) It can parse and recognize the elements of any conforming COEL Behavioural Atom, and generates the specified errors for those data objects that fail to conform as COEL Behavioural Atom.
(f) It can receive COEL Behavioural Atoms as defined in section 8.2 “COEL Behavioural Atom Protocol Interface Specification (BAP)”.
(g) It generates errors as REQUIRED in error cases described in section 8.2.
(h) It correctly implements the Information Request defined in section 7.2.2 “COEL Minimal Management Interface, Information Request”.
(i) It complies with the security requirements in section 6.1 “General Technical Principles”.

13.6 Conformance Clause 5: COEL Public Query Interface

A Data Engine process or program conforms to this specification as COEL Public Query Interface if it satisfies all the statements below:

(a) It can correctly form COEL Behavioural Atoms as defined in Clause 2: COEL Behavioural Atom.
(b) It can parse and process the functions defined in section 9.2 “COEL Public Query Interface Specification (PQI)” according to their rules and semantics.
(c) It generates errors as REQUIRED in error cases described in section 9.2.
(d) It correctly implements the Information Request defined in section 7.2.2 “COEL Minimal Management Interface, Information Request”.
(e) It complies with the security requirements in section 6.1 “General Technical Principles”.

13.7 Conformance Clause 6: COEL Identity Authority Interface

An Identity Authority process or program conforms to this specification as COEL Identity Authority Interface if it satisfies all the statements below:

(a) It can parse and process the functions defined in section 10.2 “COEL Identity Authority Interface Specification (IDA)” according to their rules and semantics.
(b) It generates errors as REQUIRED in error cases described in section 10.2.
(c) It complies with the security requirements in section 6.1 “General Technical Principles”.

Appendix A. Enumerated Fields

The following tables are the normative enumerated fields for the COEL Behavioural Atoms specified in Section 5.

When: Accuracy (Section 5.2.4 "When")

Value	***Meaning***
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)*

How: How (Section 5.2.6 "How")

Value	***Meaning***
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: Exactness (Section 5.2.8 “Where”)

Value	***Meaning***
0	*Unknown*
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 worse than 100m*

Where: Place (Section 5.2.8 “Where”)

Value	***Meaning***
0	*Home*
1	*Work*
2	*School*

Context: Social (Section 5.2.9 “Context”)

Value	***Meaning***
0	*Don’t Know*
1	*Family*
2	*Colleagues*
3	*Guests*
4	*Partner*
5	*Myself*
6	*Friends*

Context: Weather (Section 5.2.9 “Context”)

Value	***Meaning***
200	*thunderstorm with light rain*

201	*thunderstorm with rain*
202	*thunderstorm with heavy rain*
210	*light thunderstorm*
211	*thunderstorm*
212	*heavy thunderstorm*
221	*ragged thunderstorm*
230	*thunderstorm with light drizzle*
231	*thunderstorm with drizzle*
232	*thunderstorm with heavy drizzle*
301	*drizzle*
302	*heavy intensity drizzle*
310	*light intensity drizzle rain*
311	*drizzle rain*
312	*heavy intensity drizzle rain*
313	*shower rain and drizzle*
314	*heavy shower rain and drizzle*
321	*shower drizzle*
500	*light rain*
501	*moderate rain*
502	*heavy intensity rain*
503	*very heavy rain*
504	*extreme rain*
511	*freezing rain*
520	*light intensity shower rain*
521	*shower rain*
522	*heavy intensity shower rain*
531	*ragged shower rain*
600	*light snow*
601	*snow*
602	*heavy snow*
611	*sleet*
612	*shower sleet*
615	*light rain and snow*
616	*rain and snow*
620	*light shower snow*
621	*shower snow*

622	*heavy shower snow*
701	*mist*
711	*smoke*
721	*haze*
731	*sand, dust whirls*
741	*fog*
751	*sand*
761	*dust*
762	*volcanic ash*
771	*squalls*
781	*tornado*
800	*clear sky*
801	*few clouds*
802	*scattered clouds*
803	*broken clouds*
804	*overcast clouds*
900	*tornado*
901	*tropical storm*
902	*hurricane*
903	*cold*
904	*hot*
905	*windy*
906	*hail*
951	*calm*
952	*light breeze*
953	*gentle breeze*
954	*moderate breeze*
955	*fresh breeze*
956	*strong breeze*
957	*high wind, near gale*
958	*gale*
959	*severe gale*
960	*storm*
961	*violent storm*
962	*hurricane*

Consent: Purpose (Section 5.2.10 “Consent and Notice”)

Value	***Meaning***
1	*Core Function*
2	*Contracted Service*
3	*Delivery*
4	*Contact Requested*
5	*Personalized Experience*
6	*Marketing*
7	*Marketing Third Parties*
8	*Use for Delivery*
9	*Disclosure for Marketing*
10	*3rd Party Disclosure for Core Function*
11	{deliberately blank}
12	*Legally Required Data Retention*
13	*Required by Law Enforcement or Government*
14	*Protecting Your Health*
15	*Protecting Our Interests*
16	*Improve Performance*

Extension: ExtIntTag / ExtFltTag (Section 5.2.11 “Extension”)

Value	***Meaning (Format)***
1001	*Resting heart rate (bpm)*
1002	*Average heart rate (bpm)*
1003	*Maximum heart rate (bp)*
1004	*Blood pressure (SSSDDD)*
1005	*Weight (kg)*
1006	*Respiratory rate (bpm)*
1007	*Lung capacity (cl)*
1008	*Temperature (Celsius)*
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)*
1001	*Resting heart rate (bpm)*

Appendix B. Acknowledgments

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

Participants:

Paul Bruton, Individual Member
Joss Langford, Activinsights
Matthew Reed, Coelition
David Snelling, Fujitsu

Appendix C. Revision History

Revision	Date	Editor	Changes Made
0	23/01/2017	David Snelling	Initial document outline
1		Joss Langford	Atom -> Behavioural Atom Added Behavioural Atom section Moved IDA section Standardised section formats & titles Basic conformance text added
2	27/01/2017	Paul Bruton	Added section '10: Data Engine' to separate out the data engine information request from the BAP. For discussion: Further restructuring needed.
3	27/01/2017	Paul Bruton	Reverted previous changes after discussion: Decided to move Information Request from BAP to MMI Section. Re-ordered sections and conformance clauses to put MMI first
4	02/02/2017	David Snelling	Added first two sections of the MMI for discussion.
5	02/09/2017	David Snelling	Updated the definitions section for authorization protocols and two data types.
6	21/02/2017	Paul Bruton	Put in references for TLS, BasicAuth. Some minor corrections and comments.
7	24/02/2017	Joss Langford	Added sections for 'Privacy-by-Design', 'Security' & 'Identity Management'. Brought over all text from RPE.
8	24/02/2017	Matt Reed	Brought in materials on section 4.
9	01/03/2017	Paul Bruton	Accepted previous changes and copied in IDA content to section 10.
10	01/03/2017	Paul Bruton	IDA content altered to fit with new style and approach
11	01/03/2017	Paul Bruton	Review and minor correction/comments, section 1-9
12	06/03/2017	Paul Bruton	Brought in content from BAP.
13	06/03/2017	Paul Bruton	COEL-111: Corrected RFC3986 citation; COEL-86: Corrected BAP examples Some textual changes to BAP content.
14	15/03/2017	Joss Langford	Earlier changes accepted. Removed unrequired sections and updated contents.

			Removed ‘ecosystem’ from sections other than privacy by design & examples. Contained the use of ‘Classification of Everyday Living’ to only the front page. Forced the use of defined terms: COEL Architecture, COEL Model, COEL Specification, OASIS COEL-TC. Re-organised Glossary but no re-write or checking through doc.
15	16/03/2017	Paul Bruton	Section 5.3: Corrected JSON syntax of example atoms.
16	17/03/2017	Paul Bruton	Reviewed and Accepted Joss’s changes from revision 14.
17	17/03/2017	Paul Bruton	Illustration of use of JSON Schema for Behavioural Atom (Section 5.2)
18	17/03/2017	Paul Bruton	Corrected case and spelling of “Cluster, Class, SubClass, Element” controlled terms. Consistent use of either “COEL Behavioural Atom” or “Atom”
19	20/03/2017	Paul Bruton	Refining use of JSON Schema for Behavioural Atom (Section 5.2)
20	24/03/2017	Paul Bruton	Accepted previous change; Minor spelling corrections
21	31/03/2017	Joss Langford	Reviewed some previous notes & accepted changes. Reviewed and edited privacy-by-design section including COEL-87 tick & crosses
22	31/03/2017	Matt Reed	Added material on Abstract, Introduction, Objective, Summary of Concepts & Implementations.
23	31/03/2017	Paul Bruton	Accepted changes, minor spelling corrections, removed old comments.
24	3/04/2017	Matt Reed	Added material to sections 1.3 & 1.4
25	07/04/2017	David Snelling	Incorporated MMI operations.
26	07/04/2017	David Snelling	Incorporated PQI operations.
27	19/04/2017	Paul Bruton	First-pass review of MMI and PQI. Accepted changes and made a few mods and comments. Will review Query/Response structure of PQI in more detail. Reviewed Matt's changes in 1.3 & 1.4 and accepted
28	20/04/2017	Paul Bruton	Additional work on PQI to define input and output objects. This section is work-in-progress and contains a number of examples intended to illustrate the options. Once

			finalized, we will move these examples to the relevant non-normative section.
29,30,31	28/04/17	Joss Langford	Edited abstract & sections 1.1 to 1.4 Messed up and fixed auto format quotes Aligned version numbers
32	08/05/17	Paul Bruton	Accepted 29-31 changes. Clarified use of BasicAuth and NoAuth protocols. Section 6 simplified to only encompass architecture-wide requirements.
33	08/05/17	Paul Bruton	Compliance clauses 3,4,5,6 reference section 6.1 for security
34	11/05/17	Joss Langford	Accepted 32 & 33 changes. Several minor text alterations as corrections or to fix comments. Completed first draft of section 2 – introduction to architecture. Updated the complete Glossary and asserted the controlled terms throughout the document. COEL-141, COEL-123 & COEL-126 applied.
35	16/05/17	David Snelling	COEL-84: Added text for splitting Certainty among consumers associated with an IoT device. COEL-117 & COEL-137: Changed Unassign device to POST from DELETE. COEL-122: Reworked Atom Post return codes. Now one simple scheme. COEL-136 & COEL-134: Changed get operators to from GET to a POST and included suspended state flag.
36	17/05/17	Joss Langford	Updated all informative introduction texts.
37	17/05/17	Joss Langford	Section 4 COEL Model updated.
38	17/05/17	David Snelling	Added operation for getting a list of devices for a service provider. COEL-135.
39	17/05/17	David Snelling	Added text to indicate that Atoms may be returned with missing elements or the most recent or initial version. COEL-133.
40	17/05/17	Joss Langford	Updated Consent section and added potential resolutions for COEL-85, COEL-118 & COEL-120. Updated diagrams.
41	19/05/17	Paul Bruton	Accepted changes from v35,38,39 after review.
42	26/05/17	Paul Bruton	Reviewed Section 4 (COEL Model, v37 change) and accepted with few minor

			alterations; Accepted changes to Consent element. Accepted various non-normative text changes
43	26/05/17	Paul Bruton	Review section 2 and accepted changes, added some inline comments.
44	05/06/2017	David Snelling	Updated against the following issues: COEL-148 (Consent field names), COEL-146 (Length of Model Version array, and COEL-139 (Notice atoms in the model).
45	08/06/2017	Joss Langford	Reviewed comments – deleted redundant ones & replied to some; removed UPK acronym; minor corrections; changed IDA to Identity Authority to help readability in a few places; tightened the use of 'specification' as an uncontrolled term; accepted previous changes (other than PQI).
46	09/06/2017	Matt Reed	Updated Normative references [RFC2616] and [RFC3986] to point to correct citations that also include details of how the citation has been superseded and updated.
47	14/06/2017	Joss Langford	Reviewed application of COEL-95 & COEL-111 (references). Applied COEL-105 to add legend to roles/actors table.
48	16/06/2017	Paul Bruton	COEL-102, COEL-116: Reviewed use of the words 'format' and 'structure' in relation to JSON objects. Removed these words or replaced with 'content', 'form', 'communicate' etc.
49	16/06/2017	Paul Bruton	COEL-97. Removed references to interfaces being extensible. (MMI: 7.1, 7.2.2; PQI: 9.1; IDA: 10.1, 10.2.2)
50	26/06/2017	David Snelling	PQI updates for issues COEL-132, COEL-138, & COEL-140
51	27/06/2017	Paul Bruton	Accepted changes from revision 47 and 50. Made a few minor changes to wording in PQI section and added a few comments for discussion.
52	29/06/2017	Joss Langford	Moved all enumerated fields to Appendix and applied COEL-100 resolution (weather). Accepted agreed changes.
53-55	03/07/2017	Joss Langford	Fixing documents corruptions.
56	13/07/2017	Paul Bruton	Reviewed 52-55 and accepted changes from COEL-100. Inserted <links>. In place of section names/numbers

57	04/08/2017	David Snelling	Accepted edits in PQI; turned comments into issues, and proposed a table style for Query request and response.
58	04/08/2017	David Snelling	Adding another approach to the table proposal. Applied COEL-152. Applied COEL-156.
59	05/08/2017	Joss Langford	Pushed COEL-158 JSON table changes through rest of document. Clarified use of "TimeStamp" and "time stamp" removing use of "timestamp". Applied COEL-150 to clarify normative & non-normative content. All example sections formatted to 'Heading 5' with number removed. COEL-153 no longer relevant.
60	10/08/2017	Paul Bruton	Reviewed and accepted changes from 57-59. Applied COEL-147 (Version numbers in information request) Applied COEL-161 (Schema for Query Result. Also corrected text example of result)
61	13/08/2017	Joss Langford	Applied additional changes for COEL-150. Applied COEL-93 changes, Applied additional changes for COEL-153. Reviewed & accepted changes from 60. Added Matt's review input as text changes, comments or issues.
62	14/08/2017	Paul Bruton	Formatted table in 9.2.2.3. Added 'required' field in section 5.
63	18/08/2017	David Snelling	Tweaked the Query response to match the table description in verifying the schema added to his section (COEL-161). Applied COEL-154, OperatorID in Segment Data request.
64	18/01/2017	David Snelling	Minor edits and removed change tracking in Query section.
65	23/08/2017	David Snelling	Removed the Project element from Query, COEL-160. Tidied errors in Query summary and schema. Added wording for requiring the DE to distinguish non-identical Atoms, COEL-149.
66	24/08/2017	Paul Bruton	Accepted changes to Query section. Changed © to (Celsius) to avoid 'auto-correction'. Accepted text changes introduced from Matt's review (61)

67	08/09/2017	Joss Langford	Accepted previous changes. COEL-87 (1) made all text ragged right. COEL-166 applied (no charging for receiving atoms removed). COEL-162 applied (RFC2119 language).
68	08/09/2017	Paul Bruton	Accepted previous changes. While on a TC call and quorate: Removed comments and created new issues where required. Deleted reference to negative version numbers from 4.2.2. Deleted security diagram from 6.1.3. Corrected list of sections referenced from 13.1.
69	12/09/2017	Paul Bruton	COEL-164: Amended sections 7.2.3, 7.2.8, 7.2.13, 10.1, 11.3.3 to handle the rare case of a duplicate Pseudonymous Key
70	14/09/2017	Joss Langford	Previous changes accepted. COEL-163: Note added to 1.4 to say figures are informative only, figures renumbered / formatted, references and table of figures added. Removed spare carriage returns other than before level 2 titles. COEL-165: text added to 7.1 to give guidance on data protection compliance options.
71	21/09/2017	Paul Bruton	Previous changes accepted (but minor corrections to text in 7.1) 4.2.6: Added a valid subset of the COEL Model showing two example Elements and their SubClass, Class and Cluster. COEL-173
72	21/09/2017	David Snelling	Applied issues COEL-170 and COEL-149
73	21/09/2017	Joss Langford	A few minor typos & capitalisations spotted & corrected. Resolution for COEL-175 proposed in text (section 13.5). Resolution for COEL-172 proposed in text: 'events' left as a general term relating to life & Atoms used when referring to COEL encoded events. 'Device' added to glossary with definition and capitalized throughout where appropriate. 'COEL Behavioural Atom' definition updated in glossary and sections 1.2 & 4.2.2.

74	22/9/2017	Paul Bruton	Accepted previous changes. One additional change for COEL-172 in 5.2.5 and minor correction for COEL-149 in 8.2.2. COEL-177: Clarified that a new PK will be used. Removed 8.2.3 (COEL-170) and clarified behaviour when DID and CID are not registered in 8.2.2.
75	29/9/2017	Joss Langford	Accepted previous changes. COEL-171 applied which included improvements to sections 1.1, 1.2, 1.3 & 4.2.2.
76	03/10/2017	David Snelling	Accepted change from revision 75. Applied COEL-174, creating bullet lists for MMI operations in the introduction.
77	08/102017	Joss Langford	Accepted previous changes. Added COEL-176 changes for changes in Segment Data.
78	13/10/2017	Joss Langford	COEL-157 examples populated. Minor typo & style corrections.
79	20/10/2017	David Snelling	Accepted existing changes. Changed IDA info request to /home COEL-183. Clarified Suspended operators and assured and query processing COEL-182. Added better examples to Atom protocol, COEL-181. Applied COEL-180
80	20/10/2017	Paul Bruton	Accepted changes from revision 79. While quorate, made two minor edits: 10.2.2 'GET /home' and example in 8.2.2 '400 Bad Request.
81	22/10/2017	Joss Langford	Standardised punctuation, formatting and capitalization of key terms throughout document. Updated Dave's email address. Updated Fig 1 to show SP – IDA link. Corrected obvious errors and made minor changes to improve readability. Suggested wording changes in sections 11.3.3 to 11.3.5 to resolve COEL-185.
82	02/11/2017	Paul Bruton	Accepted changes from #81. Cross-references in Section 13 changed to hyperlinks. More cross-references in remainder of document still need to be addressed.

83	02/11/2017	Paul Bruton	Changed remaining cross-references and altered phrase in 5.2.9 – see COEL-186
84	03/11/2017	Matt Reed	Accepted changes made in version #83. Clarified phrase in 5.2.9 describing relation between where the Social and Weather enumeration values came from and what are normative
85	04/11/2017	Paul Bruton	All changes accepted, Submitted for public review (Quorate in meeting of 4th November).