Pacific Northwest
NATIONAL LABORATORY

PNNL-32391

Universal Utility Data Exchange (UUDEX) – Protocol Design – Rev 1

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

December 2021

SA Neumann
JL Lochner
S Sridhar
SR Mix
OA Kuchar
SV Singh
MJ Rice
CA Schmidt

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-OE0000190

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes **any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights**. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062;
ph: (865) 576-8401
fax: (865) 576-5728
email: reports@adonis.osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
email: orders@ntis.gov <https://www.ntis.gov/about>
Online ordering: http://www.ntis.gov

Universal Utility Data Exchange (UUDEX) – Protocol Design – Rev 1

Cybersecurity of Energy Delivery Systems (CEDS) Research and Development

December 2021

SA Neumann
JL Lochner
S Sridhar
SR Mix
OA Kuchar
SV Singh
MJ Rice
CA Schmidt

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Revision History

Revision	Date	Deliverable (Reason for Change)	Release #
0	8/31/2019	Initial Release	PNNL-29053
1	12/2021	Updates based on Implementation	PNNL-32391

Summary

This design document describes protocol related aspects of Universal Utility Data Exchange (UUDEX). The focus of the design is to describe the interactions between UUDEX Clients and UUDEX Servers in the UUDEX Infrastructure. This design is purposely transport and programming language agnostic.

Terms, Acronyms and Abbreviations

The following terms and acronyms are relevant to this specification:

ACL	Access Control List
AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange, as defined by ISO/IEC 646
CIM	Common Information Model, as defined by EPRI, the UCA Users Group and the IEC
CME	common message envelope
CSV	Comma Separated Values, as defined by IETF RFC 4180
DDoS	Distributed Denial of Service
DNP	Distributed Network Protocol
DOE	U. S. Department of Energy
DoS	Denial of Service
EMS	Energy Management System
EPRI	Electric Power Research institute
HTML	Hyper Text Markup Language
HMAC	hash-based message authentication code
ICCP:	Inter-control Center Communications Protocol, also known as TASE.2
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ID	Identifier
IP	Internet Protocol
ISO	International Standardization Organization
JSON	JavaScript Object Notation, as defined by IETF RFC 7159
MQTT	Message Queuing Telemetry Transport
MW	megawatt
OE-417	DOE Electric Emergency Incident and Disturbance Report
PDF	Portable Document Format, as specified in ISO 32000
QoS	quality of service
RDF	Resource Description Framework
REST	REpresentational State Transfer
RFC	Request for Comment
SBU	Sensitive but Unclassified
SCADA	Supervisory Control and Data Acquisition
TASE	Telecontrol Application Service Element

TASE.2	Synonym for ICCP
TCP	Transmission Control Protocol
Time series	A sequence of data values captured at different points in time that are telemetered, measured, or calculated that represent some aspect of the state of an object
TLS	transport layer security
UCA	Utility Communications Architecture
UDP	User Datagram Protocol
UML	Unified Modeling Language
URL	Universal Resource Locator
UUDEX	Universal Utility Data Exchange
UUID	Universally Unique Identifier, as defined by IETF RFC 4122
XHTML	eXtensible Hyper Text Markup Language
XML	eXtensible Markup Language

Additional terms pertaining to UUDEX roles and definitions are defined in Section 4 of the *UUDEX Functional Requirements* document.

Conventions

This section describes the conventions that are used within this document.

Nouns and Verbs used in messages are case insensitive.

JSON and XML examples use the following font style:

```
"metadata": {
    "messageID": "d5d1c892-974a-11e9-b198-b0c090a8aac0",
    "noun": "DataSet",
    "origin": "ACME",
    "source": "ScottNe-M.acme.net",
    "timestamp": "2019-06-25 13:12:08.218024",
    "verb": "created"
},
```

Variables are enclosed by “< >”, e.g., `<participantID>`

Data that is specific to an example, will be ***colored red with bold italics.***

Contents

Figures

Tables

1.0 Introduction

The purpose of this document is to describe the Universal Utility Data Exchange (UUDEX) protocol. The UUDEX protocol is used to convey information through the UUDEX Infrastructure (U-Infrastructure), which is a secure, highly available messaging infrastructure that provides UUDEX services. The UUDEX protocol is specified in terms of:

- Messaging patterns between UUDEX Endpoints (U-Endpoints)
- Message structures
- Application programming interfaces (API) that are used by U-Endpoints
- Common data structures, that are used to convey information or manage the U-Infrastructure

The diagram shown in Figure 1-1 provides an overview of the relationships between the U-Infrastructure, protocols, APIs, and U-Endpoints. A U-Endpoint could be a standalone software product that uses the API, or it could be a point of integration with backend applications that serve a UUDEX Participant (U-Participant).

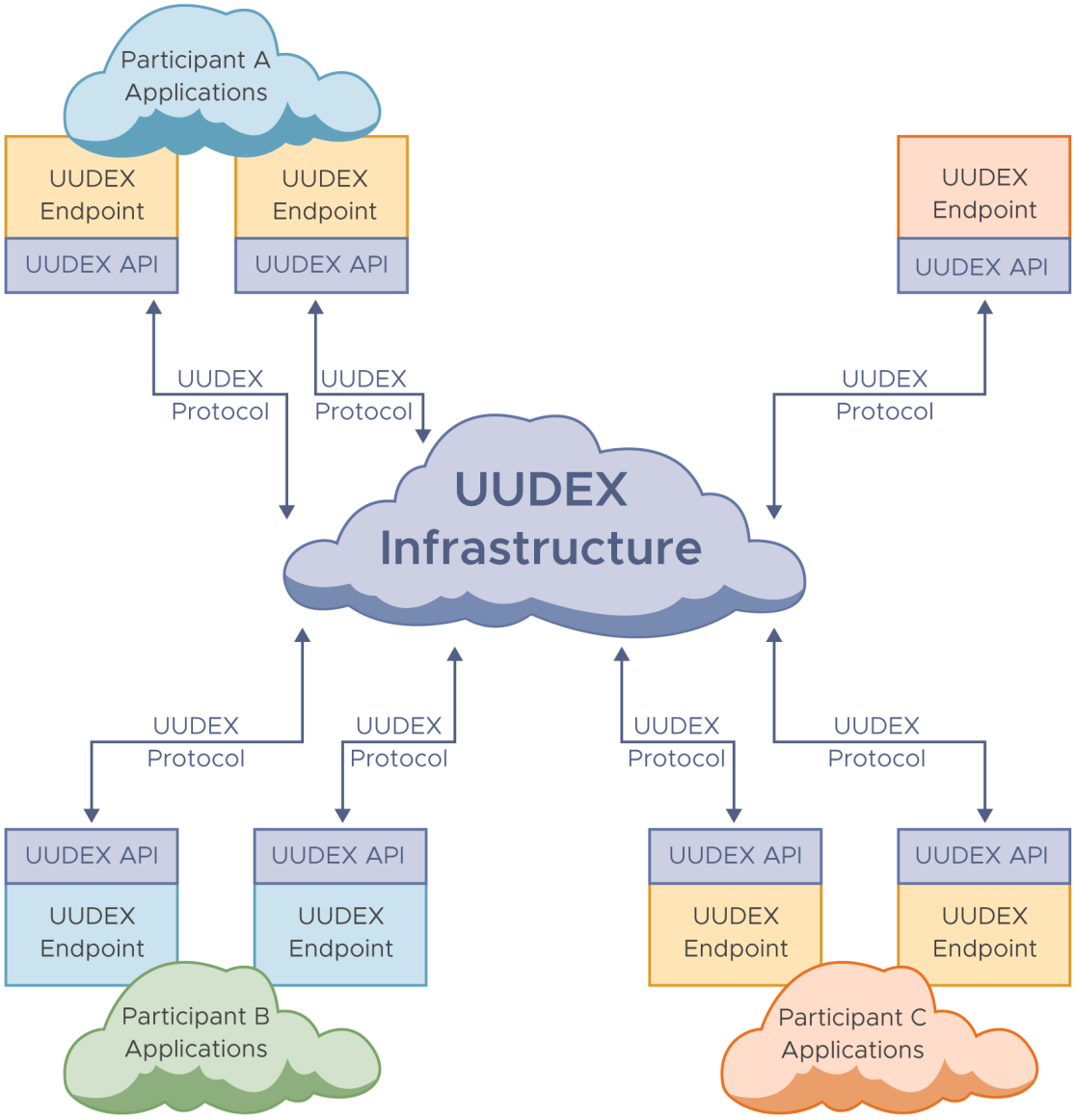

Figure 1-1: UUDEX Overview

It is the intent of UUDEX to provide functional and technical improvements over Telecontrol Application Service Element 2 (TASE.2), also known as Inter-control Center Communications Protocol (ICCP). The interface architecture of UUDEX is contrasted with TASE.2 in some key ways:

- TASE.2 only defines a point-to-point, binary protocol.
- Data items to be exchanged between parties using TASE.2 is explicitly defined, as opposed to using a discovery mechanism.
- TASE.2 is essentially limited to conveyance of measurements and controls, where in practice controls are avoided for many uses.
- UUDEX defines a set of common application programming interfaces (APIs).

- UUDEX is built upon a transport layer that provides for publish and subscribe messaging.
- UUDEX is transport neutral; it should be possible to implement UUDEX on a variety of messaging products including some that do not yet exist. Note that a strategy of “bridging" would be used to allow for interoperability between different transports.
- UUDEX allows for discovery, where publishing parties can impose restrictions on access.

Products that implement TASE.2 are delivered in three forms

- Development toolkits that can be used by applications that exchange information using the TASE.2 protocol
- Applications (e.g., Supervisory Control and Data Acquisition [SCADA], Energy management System [EMS], etc.) that internally leverage TASE.2 toolkits
- Integration hubs that allow for information exchanges using TASE.2 and other protocols (e.g., Distributed Network Protocol [DNP], Modbus, REpresentational State Transfer [REST], etc.)

The UUDEX API must support multiple programming languages. For this reason, the interfaces are described at a message-level. This simplifies the process of mapping UUDEX from a given programming language to a specific message transport.

UUDEX is designed to be largely agnostic with respect to network transport technologies. However, there are some key constraints:

- The transport must support publish and subscribe messaging.
- The transport must provide a mechanism for providing guaranteed delivery.
- UUDEX Subjects (U-Subjects) (or an equivalent) must be supported for UUDEX Subscriptions (U-Subscriptions).
- There must be flexibility with respect to message structures and data payloads.
- Restrictions to U-Subscriptions based on access control lists (ACL) must be allowed. This can be implemented directly by the transport or as a layer over the transport.
- Encryption of messages must be supported.

If the above conditions can be met, the underlying transport may be brokered or brokerless. It would also allow for choices between levels of decentralization or federation of the U-Infrastructure.

2.0 Principles

The UUDEX protocol is designed around the following key principles:

- It leverages publish and subscribe messaging technologies.
- It is agnostic with respect to messaging transports (i.e., avoid lock-in to specific products).
- It is flexible with respect to message payloads, where JavaScript Object Notation (JSON) is the primary method used for formatting of messages.
- It provides an API that can be used by clients to interact with the U-Infrastructure, where the API can be realized using commonly used programming languages.

3.0 Scope

The focus of this design is to describe the interactions between UUDEX Clients (U-Clients) and the U-Infrastructure. The following aspects are within the scope of this design:

- Definition of message structures. In a protocol design, the definition of message structures is central to information exchanges as needed for the interaction of components.
- Definition of key message payload structures for measurements, calculations, models, and structured documents. Each message will convey UUDEX Data Elements (U-Data Elements), often referred to as a “payload”. Within UUDEX, a core set of U-Data Elements are defined. The design of the message structures allows for new U-Data Elements to be defined over time.
- Describe how the Common Information Model (CIM) can be leveraged. The CIM is used in the electric utility industry worldwide, providing logical data models that are used by application software. The CIM will be leveraged to define some of the U-Data Elements that will be conveyed using UUDEX.
- Definition of core APIs. For information that is exchanged by UUDEX to be leveraged by applications, there needs to be an interface that permits the integration of U-Endpoints with local applications.

The following are out of scope:

- Physical design of the U-Infrastructure
- Mapping of the protocol to a specific messaging product
- Definition of API mappings for a specific programming language
- Providing definitions of a comprehensive set of message payload definitions
- Data streaming

4.0 Application Data Models

UUDEX supports a defined set of information exchanges in which a U-Data Element may be published by one participant for consumption by potentially many other participants.

The way these information exchanges are defined allows for augmentation over time as needs evolve. The goal of this section is to define an initial set of “standard” U-Data Elements.

4.1 Logical

The following are key classes of information exchanges that must be functionally supported by UUDEX:

- Measurement or Calculated value definitions
- Measurement or Calculated time series[1] values
- Object identifier mappings
- Models
- Structured documents
- Unstructured documents
- Binary data objects

An upper or top-level ontology, is a high-level, domain-independent ontology, providing a framework from which more domain-specific ontologies may be derived. Figure 4-1 shows the upper UUDEX ontology, a logical data model that classifies the U-Data Elements that might be exchanged using UUDEX. The upper levels of this model are defined for the purposes of UUDEX and are not domain specific (i.e., they can convey information not necessarily related to the management of electrical networks), while the bottom level classes of this model may be derived from domain-specific models such as the CIM.

[1] A time series is sequence of data values captured at different points in time that are telemetered, measured, or calculated that represent some aspect of the state of an object.

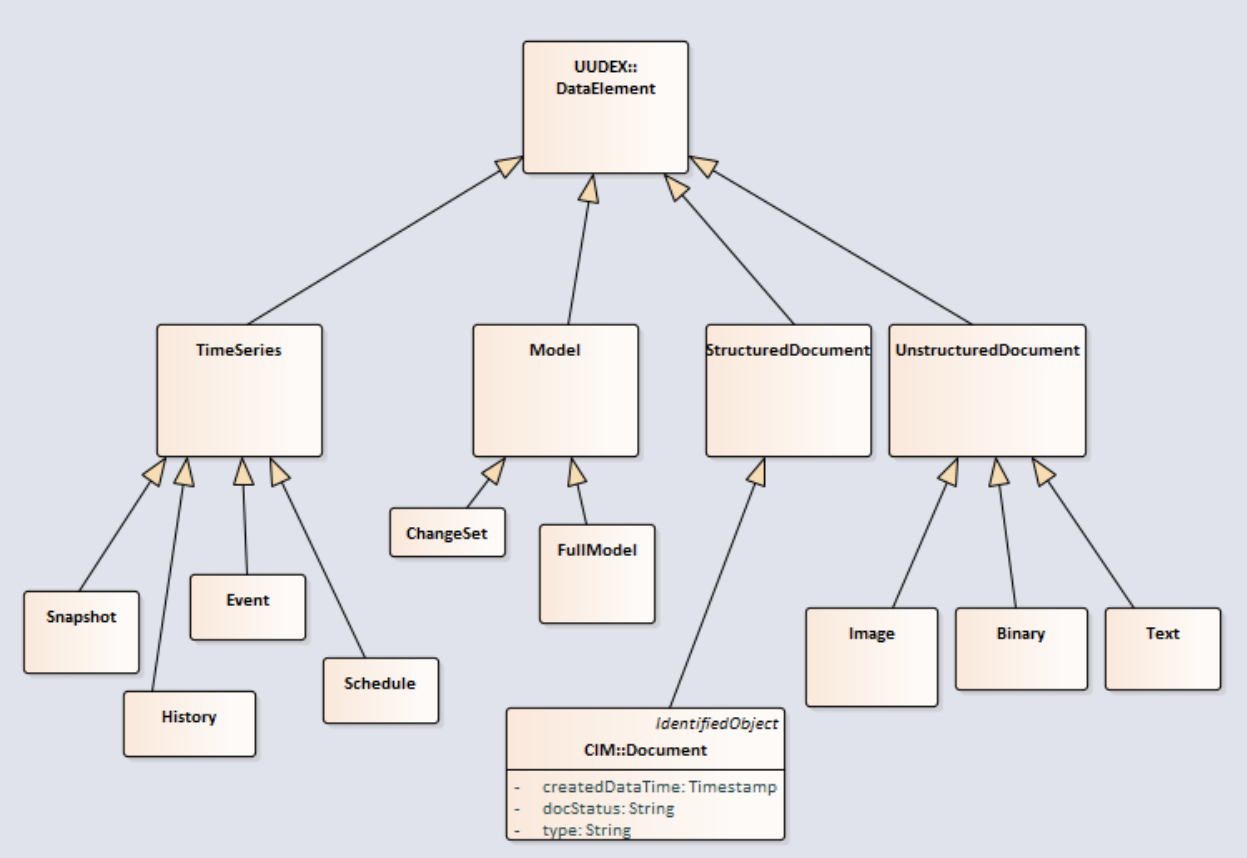

Figure 4-1: UUDEX Upper Ontology

There are many standards, as well as product-specific and custom integrations, that take subsets of the CIM to define “profiles.” Profiles define objects that have a specific subset of the classes, attributes, and relationships that are defined by the CIM logical model. Many International Electrotechnical Commission (IEC) standards currently realize these profiles as eXtensible Markup Language (XML) schemas. Efforts are underway to define and standardize direct mappings from XML to JSON. There are currently many tools and API libraries available that also provide this capability.

4.1.1 CIM Upper Ontology

These classes may be defined using domain-specific data models such as those defined by the Common Information Model (CIM). The CIM is a very detailed, complex model used for many standards. However, the CIM actually has a reasonably simple “top-level” ontology as shown in Figure 4-2. The top-level ontology of the CIM is realized by the following set of classes, relationships, and attributes. It is also important to note that the CIM is single inheritance.

From the perspective of the UUDEX upper ontology, the CIM can be used to define any of the following:

- Models
- Time series
- Structured documents

- Unstructured documents

The diagram shown in Figure 4-2 provides an upper-level view of the CIM ontology, recognizing that there are over a thousand classes currently defined in the full CIM ontology. Lower level classes would typically inherit from one of the classes shown below.

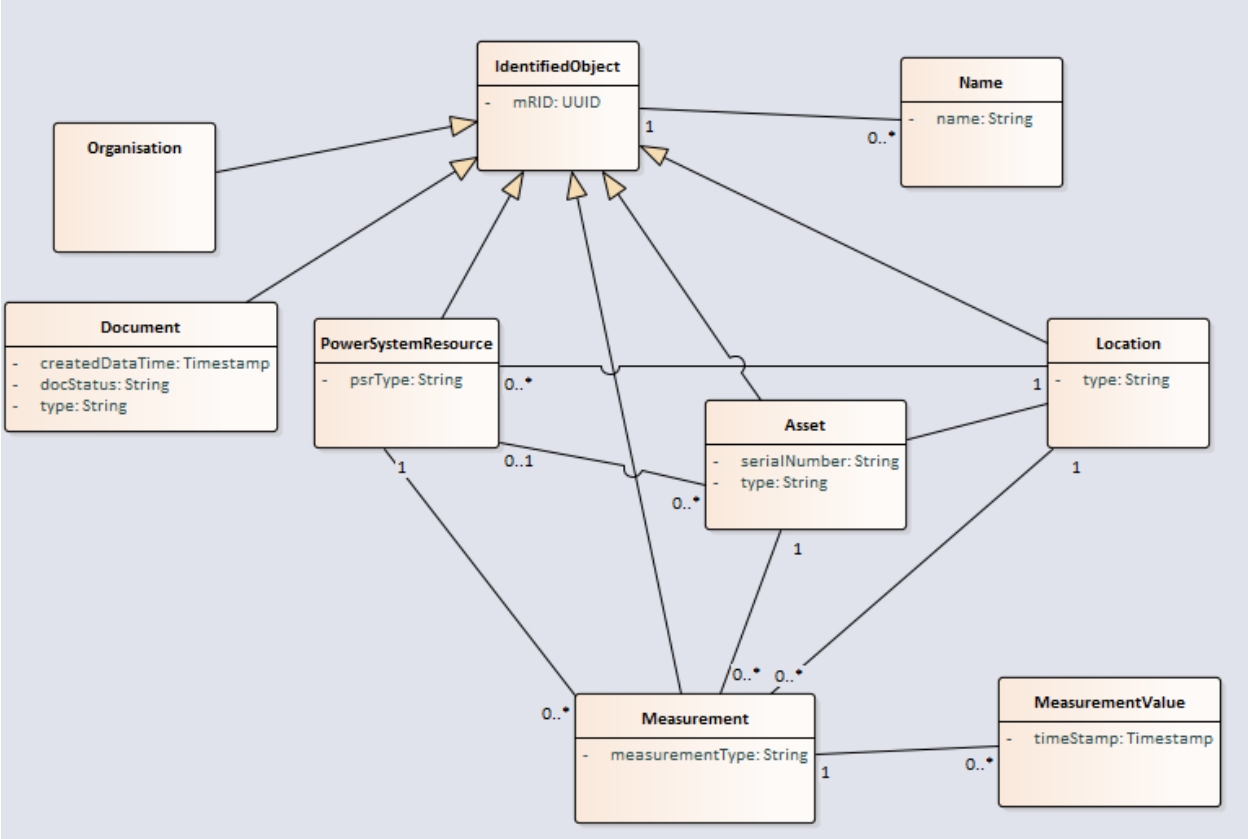

Figure 4-2: CIM Upper Ontology

The CIM upper ontology is summarized as follows:

- Virtually everything inherits from IdentifiedObject, where each object has a unique identifier known as an mRID (which is typically a Universally Unique Identifier [UUID]).
- IdentifiedObjects may have many Names, which are key to the mapping of legible, well-known, legacy, and foreign identifiers for a given object instance.
- Most IdentifiedObjects (certainly the objects of key interest for UUDEX) are classified through the CIM inheritance hierarchy as one of:
  - Asset
  - PowerSystemResource
  - Location
  - Organization
  - Document
  - Measurement.

- PowerSystemResources, Assets, Locations and Measurements are typically inter-related (e.g., at a given Location, a PowerSystemResource can be comprised of multiple Assets, and there can be many related Measurements (each with a defined measurementType) where MeasurementValues can be obtained at different points in time.
- PowerSystemResource, Asset, Location, Document and Measurement all have their types declared as an attribute (or relationship), where the type is often not an explicitly defined class, allow for more data-driven uses of the model as opposed to being tightly bound to explicit class definitions

At lower levels of the ontology that are not shown in Figure 4-2, the full CIM model also describes:

- Assets, Locations and Organizations can be defined hierarchically
- Specific Document types can have a variety of relationships (i.e., references) to PowerSystemResources, Assets, Locations and Organizations
- PowerSystemResources may be related through connectivity or containment.

4.1.2 CIM Naming

Within CIM, virtually all objects inherit from IdentifiedObject. One of the benefits of using CIM is that objects then inherit a common naming scheme. In this naming scheme:

- Each object has a unique identifier (ID), known as an mRID, which is typically implemented as a UUID.
- Each object may have many different names through use of the Name class, where each name may be associated to a specific NameType.

This is illustrated by the diagram shown in Figure 4-3.

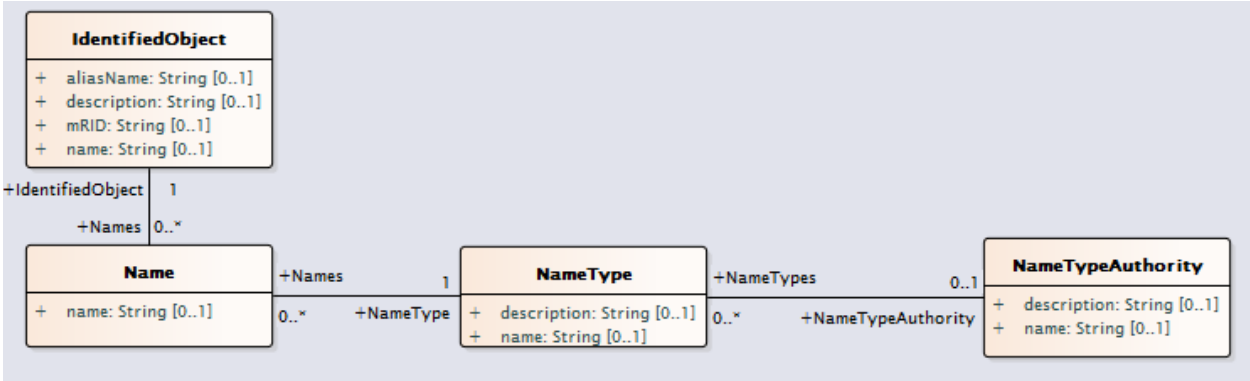

Figure 4-3: CIM Names

4.1.3 CIM Relationships

The diagram shown in Figure 4-4 provides some examples of relationships between objects in the CIM. Relationships can take the following forms:

- Containment: Where a set of objects may be contained by a container object (e.g., a region, a substation, etc.)

- Composition: Where an object may be comprised of a set of objects, where a common example is for a PowerSystemResource to be composed of a set of physical Assets, and Assets may be comprised of other Assets
- Connectivity: Where an object may be physically connected to other objects
- State: Where an object such as a measurement may identify some aspect of the state of an object
- Responsibility: Where an organization or person may have some aspect of responsibility, such as the creation, ownership, or maintenance of an object.

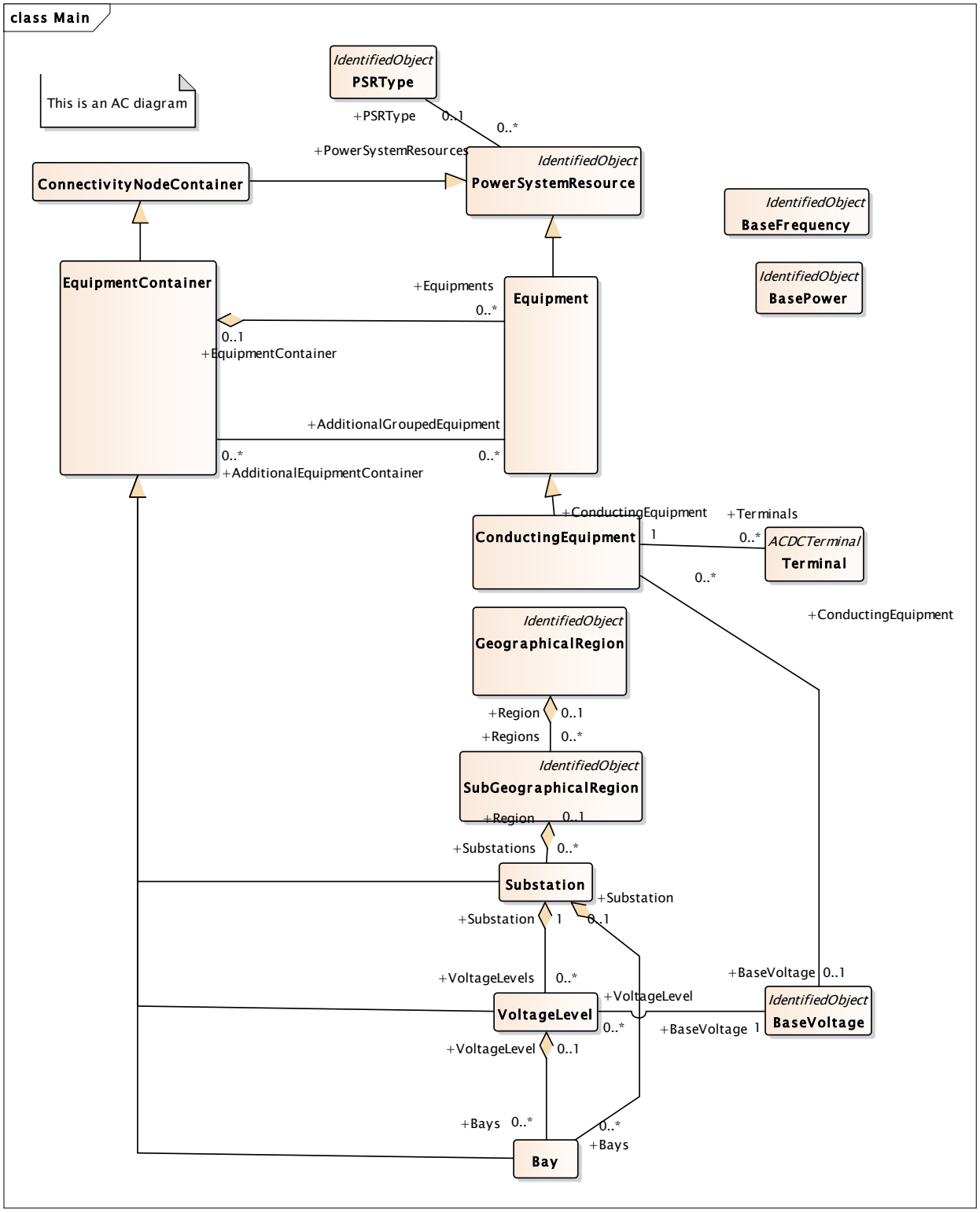

Figure 4-4: CIM Relationships

4.1.4 Time Series Data

One of the key uses of UUDEX is to convey time series data. A time series is simply a sequence of data point values that are ordered in time. Some examples of time series data include the following:

- A sample of a voltage at a substation bus bar at a given point in time
- The total amount of real energy that was measured flowing from a generator over a given time interval, as reported at a given point in time
- A calculated current flow over a transmission line at a given point in time
- The status of a circuit breaker at a given point in time
- The operation of a circuit breaker at a specific point in time
- The ambient temperature at a defined geographic location at a given point in time
- The temperature of oil within a transformer at a given point in time.

It is important to note that while many values may be obtained and reported individually, others may be obtained or need to be reported in terms of a snapshot of many values for a set of time series, where (for example) the values represent the state of some portion of the electricity grid at a given point in time. Relative phase angles and calculations from simulation applications are examples of this.

Using the CIM, a high-level view of time series data can be seen in Figure 4-5.

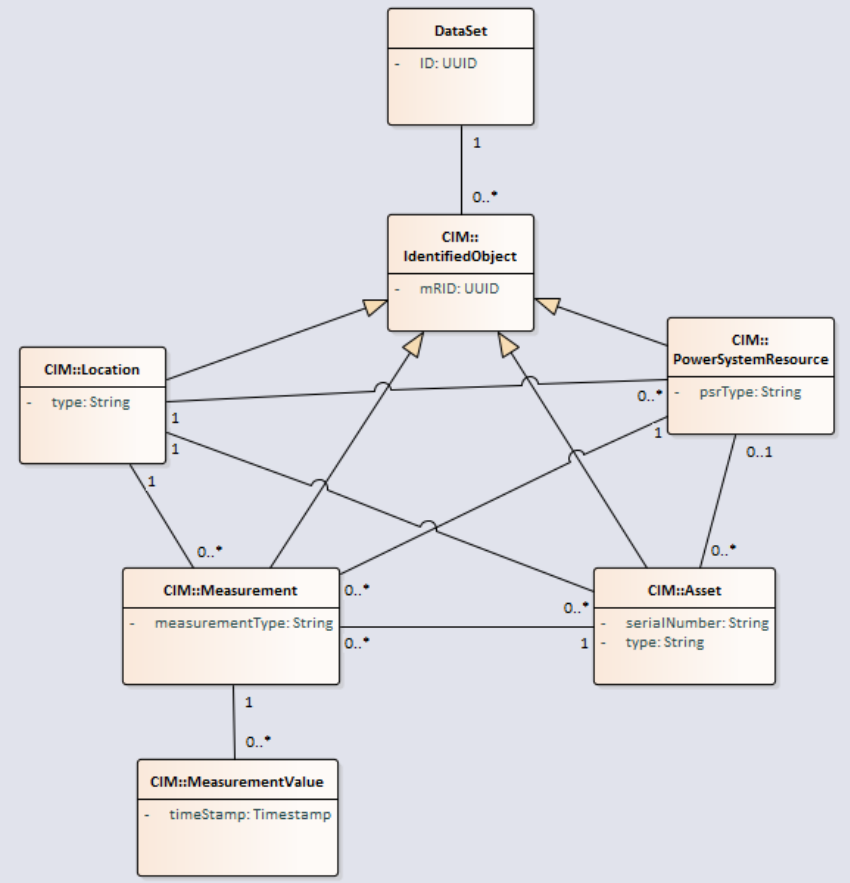

Figure 4-5: Time Series Data Model

This model shows:

- Measurements can be obtained from a variety of different classes of objects.
- Measurements are defined as objects themselves and have a defined type.
- For any defined measurement, there will be different values at different points in time.

The exchange of time series data access can take a variety forms, with the following as some examples:

- An individual value, with:
  - A point identifier
  - A timestamp (formatted using ISO 8601)
  - A value
  - An (optional) quality code
- A set of values for a set of points collected at common point in time (as is currently done with TASE.2)
- A set of values related to some common object captured at a common point in time (i.e., a snapshot), for each value:
  - A measurement type

  - A value
  - An (optional) quality code
- A set of values for a set of objects collected at common point in time
- A series of values of one or more measurement types for an object over a range of time (i.e., a history).
- A series of scheduled, planned, projected, or forecasted values of a given set of types for an object over a future time interval.

4.1.5 Models

The term “model” is heavily overloaded. For the purposes of UUDEX, a model refers to a set of object (e.g., transformer, breaker, generator, transmission line, etc.) definitions that are representative of a physical infrastructure that are needed by an application that monitors, simulates, analyzes, or optimizes the behavior of for that physical infrastructure. Sometimes this is referred to as an “instance” model, as the descriptions of specific objects are conveyed. A wide variety of applications involve the exchange of models or changes to models. An instance model will define a set of objects, relationships, and properties. Models may be exchanged that represent:

- some contiguous portion of the electricity grid for a given point in time
- changes (e.g., add, delete, or update) of objects within a model.

The term model also should be distinguished from metadata that is used to describe the content of an “instance” model. For example, the CIM is often the source of metadata used to define the content of an instance model. In this context, the term model directly refers to a power system model.

4.1.6 Structured Documents

Structured documents are information objects that convey information in a manner that is reflective of (but not necessarily constrained by) a data schema, where there are defined data fields that have a defined meaning. These are typically intended for use of application software. The meaning may be conveyed through the use of a self-describing format (e.g., JSON, XML, or comma separated values [CSV]), or through a specification that can be applied to the document for creation or interpretation. Figure 4-6 shows a high-level view of a document as defined by the CIM.

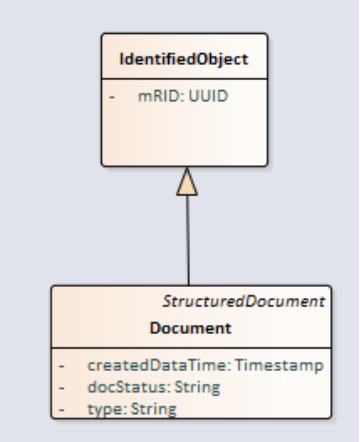

Figure 4-6: CIM Documents

Structured documents are often defined using formats such as JSON, XML, or CSV, where every U-Data Element has some tag giving the U-Data Element meaning that is useful for some application purpose. In the cases of XML and JSON, the tags are adjacent to or surrounding the data item. In the case of CSV, column headers identify the meaning of values in a given column. In all cases, the information can be readily consumed by application software.

The CIM can be a source for the definition of many types of structured documents that would convey information useful to users of UUDEX.

4.1.7 Unstructured Documents

In contrast to structured documents, unstructured documents contain data in which there is not the notion of a schema. These are typically intended for viewing by end users.

A simple example of an unstructured document is a text document that could have contents, such as a chapter from a novel.

Another example can be Hyper Text Markup Language (HTML), eXtensible Hyper Text Markup Language (XHTML) or Portable Document Format (PDF) documents. Although there are tags to describe data within a specified field (e.g., header, paragraph markers, etc.), there is no semantic meaning that can be applied to the data other than formatting the data for end user presentation. Thus, UUDEX treats these as unstructured.

4.1.8 Application Type Definitions

Application type definitions are code that can be used to provide a more detailed semantic meaning to objects, relationships, and time series data values. Within the CIM, the following can be seen:

- PSRType class
- Asset.type
- Location.type
- Measurement.measurementType
- Document.type.

The proper definition and use of type codes can provide the means to keep the underlying UUDEX protocol data driven with respect to application data. These types of definitions should be defined in a common directory, using industry standards where possible, and extending as necessary. The definition of types includes:

- category (e.g., measurement, asset, PSR, location, etc.)
- code (a short string used in messages)
- description (documentation for the type, with references to standards as appropriate)

4.1.9 Quality Codes

Quality codes provide insight as to the usefulness of a specific time series value. TASE.2 used a simple scheme for quality code that leveraged bit fields. UUDEX will use a more self-describing scheme.

Specific quality codes are discussed in Section 4.2.8.

4.2 Physical

The purpose of this section is to describe the format of U-Data Elements that are conveyed using UUDEX.

4.2.1 UUDEX Data Element Formats

UUDEX must be able to handle U-Data Elements that may originate from a wide variety of data sources, where each source may use a variety of data formats. Source data formats may fall into one of two main classifications:

- Legible text formats
- Binary (non-legible) data formats

Legible text formats include the use of standards such as JSON, CSV and a variety of XML dialects, in which data are conveyed in a self-describing manner. Advantages of a self-describing format are readability, flexibility, and extensibility, while disadvantages include verbosity and sometimes unnecessary complexity. One example of verbosity is seen with XML, depending upon the dialect used (e.g., XML schema-based vs. Resource Description Framework [RDF]). To overcome the verbosity, data are often compressed and encoded prior to transmission. See Section 4.2.5 for additional information.

The physical data models used for U-Data Elements can be realized using one of the following forms, which provide options for flexible as well as compact data formats:

- JSON objects
- Text strings that can the consequence of:
  - Simple text strings
  - Compressed and encoded XML documents
  - Compressed and encoded binary formats

Metadata in the UUDEX Message Envelope (U-Message Envelope) will indicate the nature of the U-Data Element provided within the payload element.

4.2.2 JSON Data Elements

JSON is the default format to be used for message structures and U-Data Elements. It is important to note that while throughout this document JSON examples are shown using multiple lines with aligned tabs for readability, an actual message should not use newline or tab characters in order to minimize the message size and not introduce spurious characters in text strings. Even more important, in order to have repeatable hash values for JSON data objects, a canonical JSON format is needed.

There may also be cases where additional efficiencies can be achieved through the use of "short aliases" for element tag names when realizing JSON elements. This is especially useful for repetitious data structures such as the conveyance of measurements and readings. The following are some examples:

- measurementType -> mt
- quality -> q
- timestamp -> ts
- value -> v

In cases like these where there are definite benefits, mappings back to standards such as CIM can be readily managed.

4.2.3 Non-JSON Data Objects

While the UUDEX protocol is based upon JSON, the U-Data Elements that are conveyed are not restricted to the use of JSON. This will allow objects formatted using, for example, XML, PDF, CSV, and binary images to be conveyed. If the object consists of "printable" characters (i.e., no extended or control characters) such as an XML document, it can simply be encapsulated within the JSON structures as a string (with escaped character sequences for certain reserved characters). If the object consists of non-printable characters such as a PDF file, it can be encoded into a neutral format as described in Section 4.2.5.

In the example of an OE-417 completed document shown in Section 4.2.5, the native format is PDF. This PDF file would be compressed and encoded for efficient transfer. It is important to note that XML documents can be derived from PDF forms, but it is outside the scope of this specification to provide the details of that process or normative XML specification.

4.2.4 XML to JSON Mapping

The purpose of this section is to describe how XML documents "could" be mapped to JSON in the event that there is a specific desire to use a JSON format as opposed to an XML format for a specific U-Data Element Type where a legacy XML format might exist. However, unless there has been a specific decision to convert an XML format to JSON for the purposes of UUDEX, XML documents should continue to be conveyed as XML using the formatting procedure defined in section 4.2.3.

Where the CIM defines a logical information model, these models are often realized using XML within related industry standards. The conversion from an XML document to a JSON object is trivial.

The Unified Modeling Language (UML) diagram shown in Figure 4-7 describes a CIM-derived profile for meter readings. This provides a “logical” view.

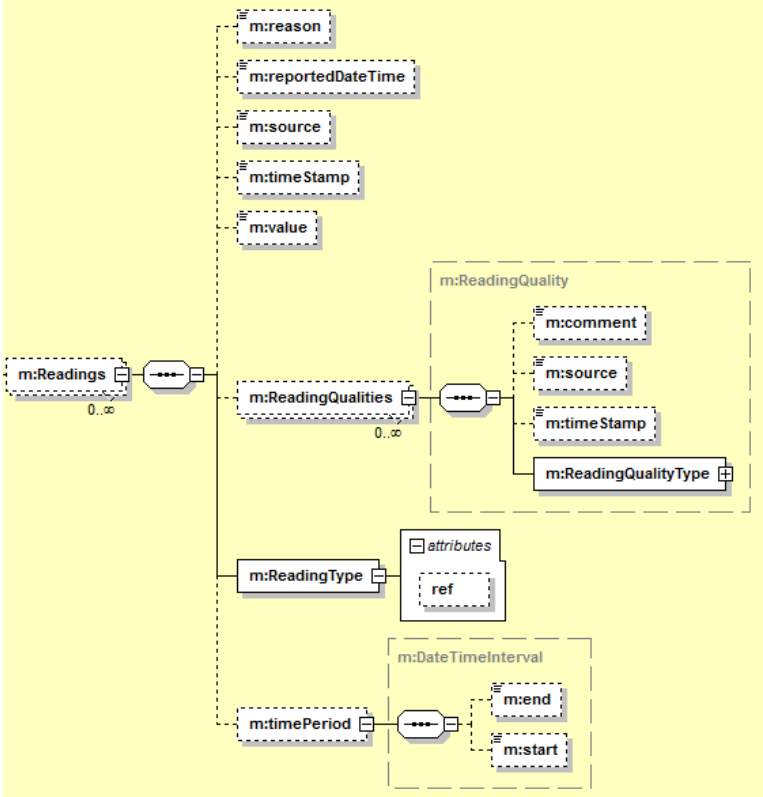

Figure 4-7: CIM Meter Readings Structure

From this view, an XML schema was defined that results in the following instance data example:

```
<ns1:MeterReadings xmlns:ns1="http://iec.ch/TC57/2011/MeterReadings#">
  <ns1:MeterReading>
    <ns1:Meter>
      <ns1:mRID>3dc53ee5-777e-50b4-8699-a1c224f45f3d</ns1:mRID>
      <ns1:Names><ns1:name>Meter34365</ns1:name></ns1:Names>
    </ns1:Meter>
    <ns1:Readings>
      <ns1:timeStamp>2011-10-24T20:06:26.016-04:00</ns1:timeStamp>
      <ns1:value>34644</ns1:value>
      <ns1:ReadingType ref="0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0"/>
    </ns1:Readings>
    <ns1:Readings>
      <ns1:timeStamp>2011-10-24T20:06:26.016-04:00</ns1:timeStamp>
      <ns1:value>1</ns1:value>
      <ns1:ReadingType ref="0.0.0.0.0.1.11.0.0.0.0.0.0.0.0.109.0"/>
    </ns1:Readings>
  </ns1:MeterReading>
</ns1:MeterReadings>
```

A transformation from XML to JSON would result in the following:

```
{
  "MeterReadings": {
    "MeterReading": {
      "Meter": {
        "mRID": "3dc53ee5-777e-50b4-8699-a1c224f45f3d",
        "Names": {
          "name": "Meter34365"
        }
      },
      "Readings": [
        {
          "timeStamp": "2011-10-24T20:06:26.016-04:00",
          "value": "34644",
          "ReadingType": "0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0"
        },
        {
          "timeStamp": "2011-10-24T20:06:26.016-04:00",
          "value": "1",
          "ReadingType": "0.0.0.0.0.1.11.0.0.0.0.0.0.0.0.109.0"
        }
      ]
    }
  }
}
```


Note the key differences listed below:

- Namespaces are removed.
- XML attributes and realized the same as other elements.
- All element values are realized as strings.
- Where the XML conforms to an XML schema, there is no JSON schema.

It is important to note that a document conforming to an XML schema can be readily and generically converted to JSON. However, a generic conversion in the reverse direction (from JSON to XML) would not necessarily result in an XML document that conformed to a specific XML schema. For example, without specific translations being defined, it is not be possible for a generic JSON to XML conversion to decide whether a JSON element should be realized as an element or attribute in the resulting XML. If XML is needed by both publisher and subscriber, the XML document should be conveyed using the approach described in section 4.2.5.

4.2.5 Payload Compression and Encoding

There are situations where very large data payloads, as in the case of network models, need to be exchanged. The IEC 61970 series of standards often uses RDF (an XML dialect) for formatting models and model changes. To reduce the size of the payload, compression (e.g., using gzip) can be used to reduce the size of the transmitted payload, with data reduction ratios in the ranges of 14:1 to 25:1 commonly seen. For some types of objects that are not frequently conveyed, payload size may not be an issue; however, for model or real-time measurements, payload size can be a significant issue.

The basic requirement for the conveyance of a U-Data Element is to be encoded using the UNICODE UTF-8 (was ISO/IEC 8859-1) character set. Often the compression process produces binary (i.e., non-printing) characters that may be inadvertently interpreted during telecommunications processing, so the binary data is converted to a sequence of printable 7-bit ASCII (ISO/IEC 646) characters for transmission. The process of generating a conformant string from an arbitrary binary data object is called encoding. The encoding process results in characters from the ISO/IEC 646 character set (which is a formal subset of UTF-8). The

encoding process increases the size of the data by 35% (when using the BASE64 encoding algorithm), but it is still smaller than the original uncompressed file. Once received, the printable ISO/IEC 646 characters are re-converted to the original binary form by a process known as decoding.

Following is a fragment of an example of a string that can be conveyed as a message U-Data Element, which is the result of the compression and encoding of an OE-417 PDF form:


```
"JVBERi0xLjYNJQ0KMTQgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMzI4NDkvTyAxNi9FIDI3Nzgx
L04gMS9UIDMyNTQ2L0ggWyA0NTUgMTUzXT4+DWVuZG9iag0gICAgICAgICAgICAgICAgICANCjIxI
DAgb2JqDTw8L0RlY29kZVBhcm1zPDwvQ29sdW1ucyA0L1ByZWRpY3RvciAxMj4+L0ZpbHRlci9GbG
F0ZURlY29kZS9JRFs8QkJBODdEMEI0MDY0MzE0QjkxOUI5OUY3MjJCQ0EzMjY+PEIzNEE2NEEzMUJ
FMkNDNDA5QjhGNDFFREREMjJEOUNBPl0vSW5kZXhbMTQgMTFdL0luZm8gMTMgMCBSL0xlbmd0aCA1
NC9QcmV2IDMyNTQ3L1Jvb3QgMTUgMCBSL1NpemUgMjUvVHlwZS9YUmVmL1dbMSAyIDFdPj5zdHJlY
W0NCmhiYmQQYGBiYAoAEgxTBC4QcRBEAlggMVVhHkgJAyN/AQIMAAcNCmVuZHN0cmVhbQ1lbmRvYm
oNc3RhcnR4cmVmDQowDQolJUVPRg0KICAgICAgDQoyNCAwIG9iag08PC9DIDczL0ZpbHRlci9GbGF
0ZURlY29kZS9JIDk1L0xlbmd0aCA2OS9TIDM4Pj5zdHJlYW0NCmhiYGBgZWBgZgAMEVkYOBoEWGEX
Az9hMhBwKmlVcA0sDAx1bFBRWQADADwWBQ0KZW5kc3RyZWFtDWVuZG9iag0xNSAwIG9iag08PC9MY
W5nKABFAE4ALQBVAFMpL01hcmtJbmZvPDwvTWFya2VkIHRydWU+Pi9NZXRhZGF0YSAyIDAgUi9QYW
dlTGF5b3V0L09uZUNvbHVtbi9QYWdlcyAxMiAwIFIvU3RydWN0VHJlZVJvb3QgNiAwIFIvVHlwZS9
DYXRhbG9nPj4NZW5kb2JqDTE2IDAgb2JqDTw8L0NvbnRlbnRzIDE3IDAgUi9Dcm9wQm94WzAuMCAw
LjAgNjEyLjAgNzkyLjBdL01lZGlhQm94WzAuMCAwLjAgNjEyLjAgNzkyLjBdL1BhcmVudCAxMiAwI
FIvUmVzb3VyY2VzPDwvRm9udDw8L1RUMCAyMyA
.
.
.
+PEIzNEE2NEEzMUJFMkNDNDA5QjhGNDFFREREMjJEOUNBPl0vSW5mbyAxMyAwIFIvTGVuZ3RoIDQ4
L1Jvb3QgMTUgMCBSL1NpemUgMTQvVHlwZS9YUmVmL1dbMSAyIDFdPj5zdHJlYW0NCmhiYgACJlYGJ
gYqICFQCWIIWUQ9IBEHG0gwMgAEGAAFIw0KZW5kc3RyZWFtDWVuZG9iag1zdGFydHhyZWYNCjExNg
0KJSVFT0YNCg=="
```


In this way, virtually anything can be conveyed as a U-Data Element in a UUDEX message.

4.2.6 Models

The exchange of U-Data Elements that convey Models will typically have the following characteristics:

- The structure of the data will conform to some specification that is only useful to some specific set of applications (e.g., IEC 61970-452).
- The model will involve descriptions of many objects and as a consequence related exchanges will be very large and should be compressed as described in Section 4.2.5.
- The model format specification may have provisions for transferring “change sets” as opposed to full models.

4.2.7 Application Data Types

UUDEX uses a self-describing method of defining data elements and data types. These are often specific to the message context and type of data being described. Specific descriptions of the application data types are described in the *UUDEX Information Exchange Structures* document.

4.2.8 Quality Codes

Quality code values will be conveyed using the “quality” element in messages. These can be derived from IEC 60870-6-802 standard.

The TASE.2 standard defines the following quality attributes:

1. *Validity:* This attribute may have one of the following values:
   - VALID
   - HELD (e.g., out of service)
   - SUSPECT (e.g., old value)
   - NOT VALID (i.e., bad)
2. *Current Source*: This attribute may have one of the following values, which describes the source of the value:
   - TELEMETERED
   - ENTERED
   - CALCULATED
   - ESTIMATED
3. *Normal Value*: This attribute may have one of the following values, which indicate if the value is within a permissible or expected normal range
   - NORMAL
   - ABNORMAL

These are conveyed using the Data_Flags structure defined by TASE.2 in IEC 60870-6-802:

```
Data_Flags bit-string:
  {
    unused[0],
    unused[1],
    Validity_hi[2],
    Validity_lo[3],
    CurrentSource_hi[4],
    CurrentSource_lo[5],
    NormalValue[6],
    TimeStampQuality[7]
  }
```

In the above definition, the most significant bit is identified as bit 0, which is the reverse of a power of two based ordering. The quality code is conveyed as an eight-bit integer value, which is derived using a bitmask of the above quality attributes. The specific values associated with each of the attributes is described below:

- Validity (bits 4-5):
  - Valid = 0x00 (=0)
  - Held = 0x10 (=16)

  - Suspect = 0x20 (=32)
  - Bad or Invalid = 0x30 (=48)
- Source (bits 2-3):
  - Telemetered = 0x00
  - Calculated = 0x04 (=4)
  - Entered = 0x08 (=8)
  - Estimated = 0x0C (=12)
- State (bit 1):
  - Normal = 0x00 (=0)
  - Abnormal = 0x02 (=2)

The calculated quality code is the sum of the above options for validity, source, and state. Examples are provided in Table 4-1:

Table 4-1: Example Quality Code Calculations

Description	Status	Source	State	Sum	TASE.2 code
Good telemetered value in normal range	Good	Telemetered	Normal	0+0+0	0
Good telemetered value in abnormal range	Good	Telemetered	Abnormal	0+0+2	2
Bad telemetered value	Bad	Telemetered	Normal	48+0+0	48
Good calculated normal value in normal range	Good	Calculated	Normal	0+4+0	4
Suspect telemetered value in abnormal range	Suspect	Telemetered	Abnormal	32+0+2	34
Suspect entered value in abnormal range	Suspect	Entered	Abnormal	32+8+2	42

Additional quality indications will be defined.

UUDEX uses text descriptors rather than bit masks to convey quality codes allowing an extensible representation of quality. For example, the TASE.2 code of 0x00, representing a status of “good”, a source of “telemetered” and a status of “normal” would be represented in UUDEX as the following structure:

```
“quality”:{
  “currentSource”:”TELEMETERED”,
  “normalValue”:”NORMAL”,
  “validity”:[
    “VALID”
  ],
}
```

Additional attributes such as `normalSource`, `selection`, and additional `validity` attributes can also be included in the `quality` block to convey additional quality parameters. Abbreviations for the names and values are also defined, allowing the size of the transferred structure to be smaller than shown.

5.0 Infrastructure Data Models

Infrastructure data models describe the information needed to authorize and enable the exchange of information through the U-Infrastructure.

5.1 Logical

The high-level diagram shown in Figure 5-1describes the logical information model that is used to authorize and manage UUDEX information exchanges.

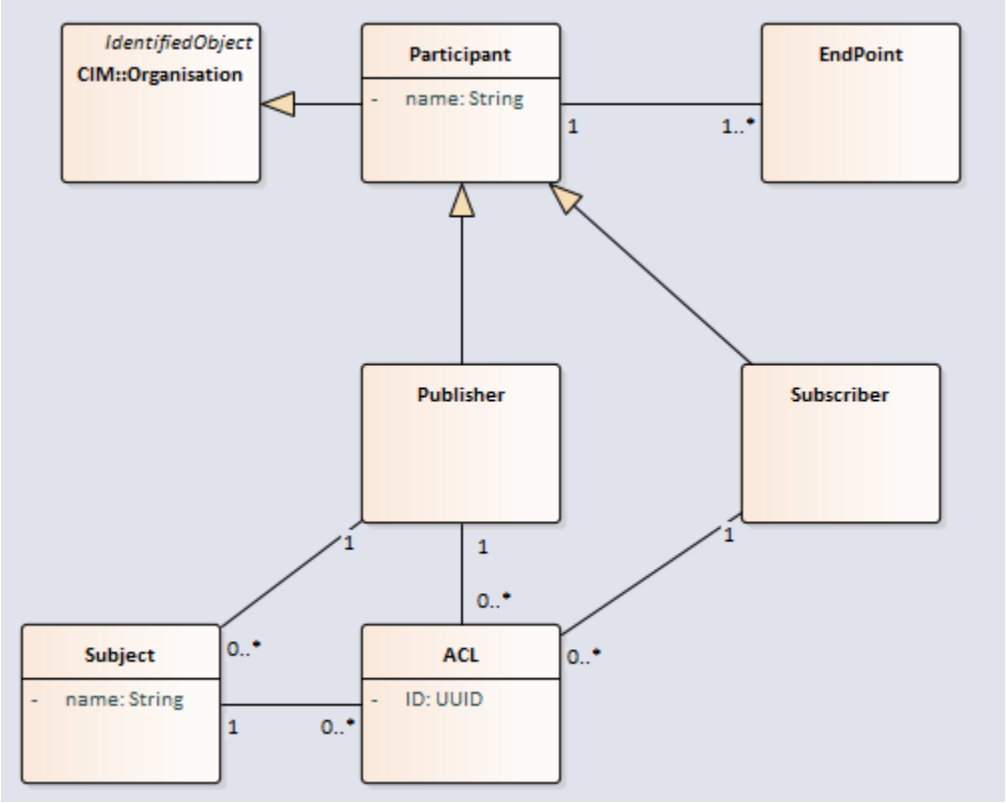

Figure 5-1: UUDEX Participants, UUDEX Endpoints, UUDEX Subjects, UUDEX Publishers, and UUDEX Subscribers

5.1.1 UUDEX Participants

A U-Participant is an organization that is authorized to connect to the U-Infrastructure via one or more U-Endpoints, where U-Endpoints connect using a U-Client interface.

A U-Participant has the following properties, where the details of each are described in subsequent sections:

- An ID (i.e., a name, will be used as “origin” in messages)
- Description
- List of authorized U-Endpoints within the organization
- List of authorized subjects (i.e., specific subjects of interest owned by other participants)

- List of subjects that are offering for publication to other participants
- List of authorized subscribers for each subject
- List of contacts (i.e., persons with contact information)

The details of U-Endpoints and UUDEX Subjects (U-Subjects) are defined in subsequent sections. U-Participant registrations, U-Endpoints, U-Subjects and ACLs will be managed by a U-Infrastructure.

5.1.2 UUDEX Endpoints

A U-Endpoint uses a U-Client interface to interact with the U-Infrastructure, where the U-Endpoint permissions are a subset of those defined for the parent U-Participant. The properties of a U-Endpoint include the following:

- U-Participant ID
- U-Endpoint ID (i.e., a stable name)
- Description
- Internet protocol (IP) address
- Permissions

The intent is for a participant to deploy one or more U-Endpoints, as might be needed to:

- Allow for redundancy
- Allow for decentralization

5.1.3 UUDEX Data Element Types

UUDEX is designed to allow conveyance of U-Data Elements between participants. Each U-Data Element is associated to a defined type. UUDEX Data Element Types (U-Data Element Types) are defined and maintained within the U-Infrastructure. The properties of a U-Data Element Type include the following:

- ID (i.e., a name)
- Description
- Data format (e.g., JSON, text, XML, etc.)
- Version list:
  - Version number
  - Description
  - Schema (optional)
  - Specification reference (optional)

An important note is the UUDEX is mostly agnostic to U-Data Element Types. However, a set of standard U-Data Element Types will be defined by UUDEX. The set of U-Data Element Types may be extended as needed. The U-Infrastructure will identify the set of defined U-Data Element Types, where enough information is provided to permit the publisher and consumer to

agree on structure, format, and content. This is where a schema may be explicitly stored within the U-Infrastructure. The specification reference would be the name of an accepted industry standard (e.g., CIM) or specification, or a Universal Resource Locator (URL) reference.

5.1.4 UUDEX Subject

A U-Subject defines the lowest level of granularity for the publication and subscription of information. A subject has the following properties:

- Name (see below)
- Source (i.e., the ID of the participant that is offering the subject)
- U-Data Element Type ID
- Group key (default = 0; for example, a code used to group objects)
- Persistence flag, which indicates that none, all, or the most recent U-Data Elements published on a U-Subject should be retained.
- An ACL which restricts or allows various access, including publish, subscribe, or discovery
- Priority (optional, 1 is the highest priority)
- Update frequency, which indicates the timeframes during which data are published. Options include:
  - On event (Common to status changes and notifications)
  - Number of seconds (common to analog values and other time series data)

The relationships between subjects and other UUDEX classes is described by the diagram shown in Figure 5-2:

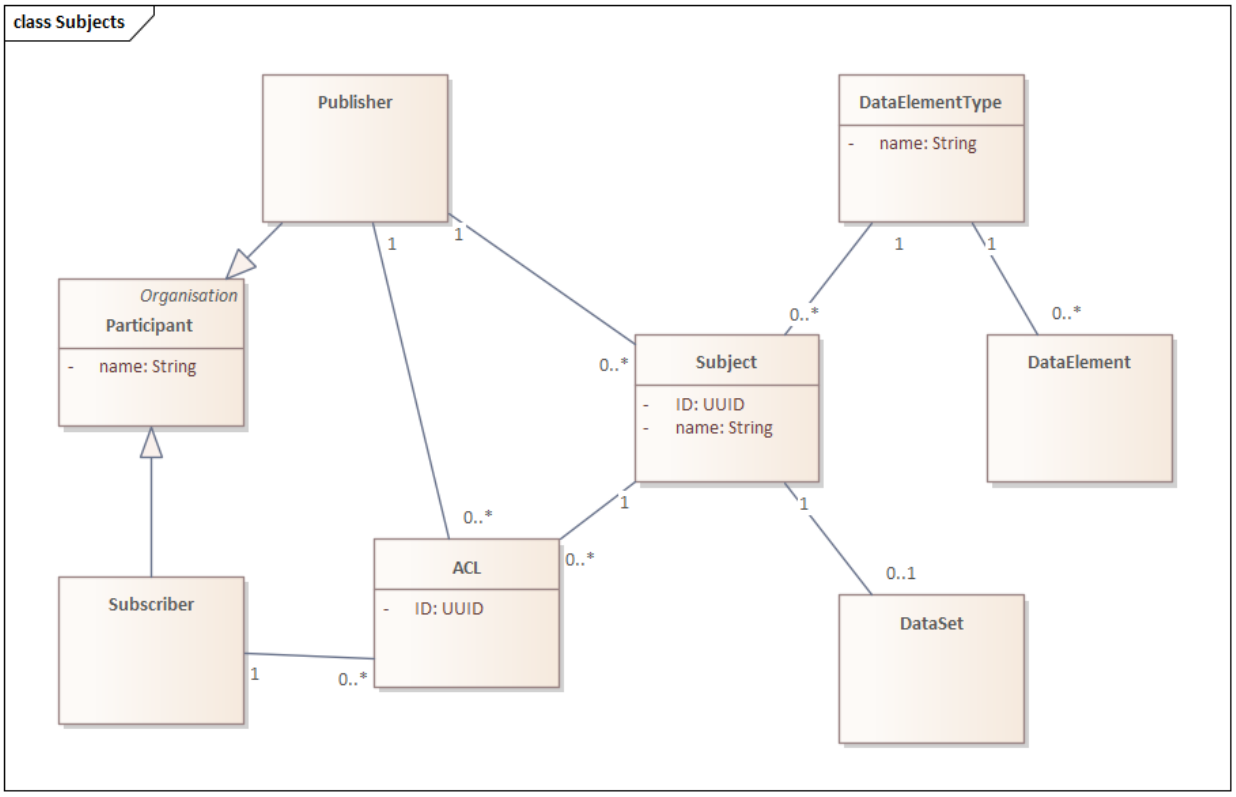

Figure 5-2: UUDEX Subjects and Key Relationships

Following are examples of subjects that might potentially be defined, and related information published, by a given participant:

- Transmission breaker status changes
- Periodic snapshot of transmission breakers
- Periodic snapshot of generator megawatts
- Periodic snapshot of interchange megawatts and frequency
- Transmission model changes
- Transmission outages
- Periodic snapshot of Current Operating Plans

Once a participant defines one or more subjects, the following actions could occur:

- U-Subjects are made visible to other participants by the creating or owning participant through definition of ACLs.
- Based on the ACLs:
  - Other participants can see the subjects exist.
  - Other participants can subscribe to specific subjects of interest.
- U-Data Elements are published to the subjects and consumed by subscribers.

5.1.5 Access Control Lists

The U-Infrastructure manages a set of ACLs. The ACLs are key to the U-Infrastructure, as the ACL definitions are needed to enforce the following basic rules:

- Only valid U-Endpoints that belong to valid U-Participants can connect to the U-Infrastructure. The specific characteristics of U-Endpoints and U-Participants that make them "valid" will be defined in the *UUDEX Workflow Design* document. Each U-Endpoint used within a UUDEX Instance (U-Instance) will be individually registered as a means to control access to that U-Instance.
- A U-Endpoint may be restricted to allow or restrict use of the following client interfaces. This restriction would be an operational limit imposed by a U-Participant's organizational restrictions and not something enforced by ACLs. However, no action will be permitted that violates a U-Subject's ACL regardless of the U-Endpoint's assigned interfaces.
  - Producer (publication of U-Data Elements, and deletion of those published U-Data Elements, to a given U-Subject)
  - Consumer (consumption of U-Data Elements from a given U-Subject)
  - Discovery (view lists of U-Subjects to which the U-Participant may publish or subscribe)
  - Administrator (create or revoke U-Subjects and their permitted subscribers).
- A U-Endpoint may only publish or subscribe to those U-Subjects that are authorized for the U-Endpoint's U-Participant. In most cases, authorization would be ascribed to a U-Participant and all the U-Participant's U-Endpoints would share that access.

For the above reasons, the ACLs will define the following for each subject:

- U-Subject name
- ID of owning U-Participant (also provided within the U-Subject name)
- List of allowed subscribers (U-Participant IDs)

When a U-Participant is registered, an initial set of ACLs should be defined by a UUDEX Administrator (U-Administrator). The default security should be to deny access.

5.1.6 UUDEX Infrastructure

The U-Infrastructure maintains the information needed for a U-Endpoint to publish or subscribe for information. It is intended to allow for discovery of information that could be potentially published by participants. The key logical elements of the U-Infrastructure are described at a high-level in Figure 5-3:

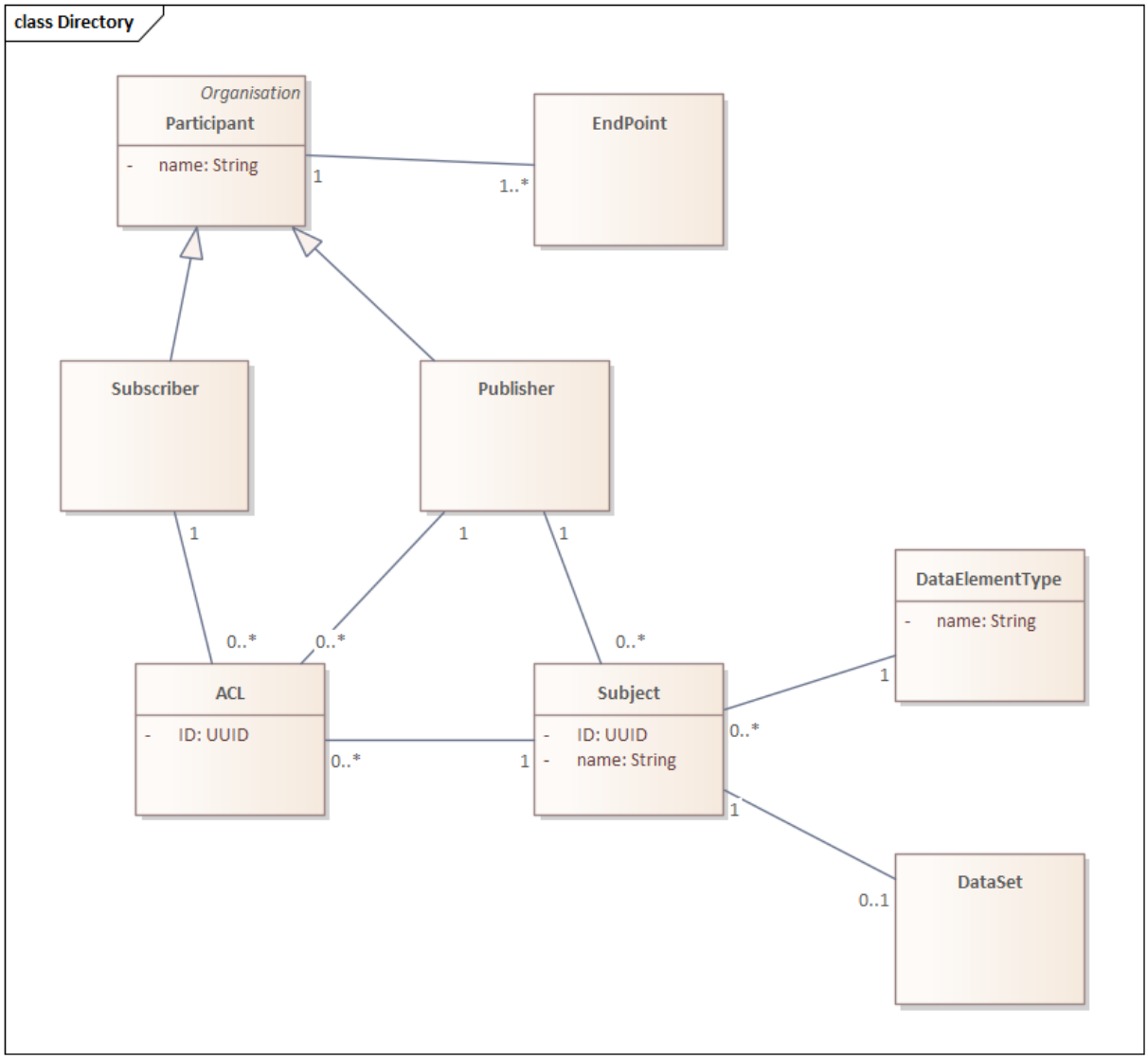

Figure 5-3: UUDEX Infrastructure

A key aspect is the visibility of subjects as per defined ACLs, where not all subjects are visible to all participants. From the perspective of a U-Client, they are able to see the following:

- The set of U-Participants that are visible, as per ACLs
- The set of defined U-Data Element Types
- The set of subjects to which they may subscribe as defined by ACLs
- The set of subjects on which each participant may publish specific U-Data Element Types as per defined subjects and ACLs
- For each visible subject, the details of that subject

It is important to note that the U-Infrastructure could be implemented in a number of ways:

- A logically centralized service with all subjects and associated ACLs, with replicas for availability

- Federated, where each participant maintains their own subject list and ACLs

The U-Infrastructure would also benefit from the inclusion of common “reference” data. When possible, this reference data should be based upon industry standards where those standards and portions of interest to UUDEX are clearly identified. Examples of common reference data include:

- Schema definitions for a given U-Data Element Type
- Reading or Measurement types that provide descriptions of time series data values

5.1.7 UUDEX Data Sets

A UUDEX Data Set (U-Data Set) is a collection of U-Data Elements that are logically grouped together. Unlike TASE.2, U-Data Sets are not pre-defined or ordered, allowing greater flexibility in composing individual UUDEX Messages (U-Messages) that are published to U-Subjects. A U-Data Set is the most common payload sent in U-Messages.

U-Data Elements in a U-Data Set are individual objects, where each object may have a set of values that change over time. These values are commonly known as “time series.” These values reflect the state of an object at a given point in time and may be obtained using telemetry, measurements, calculations, or estimation techniques.

Figure 5-3 describes the relationships among U-Data Sets, IdentifiedObjects, and CIM objects.

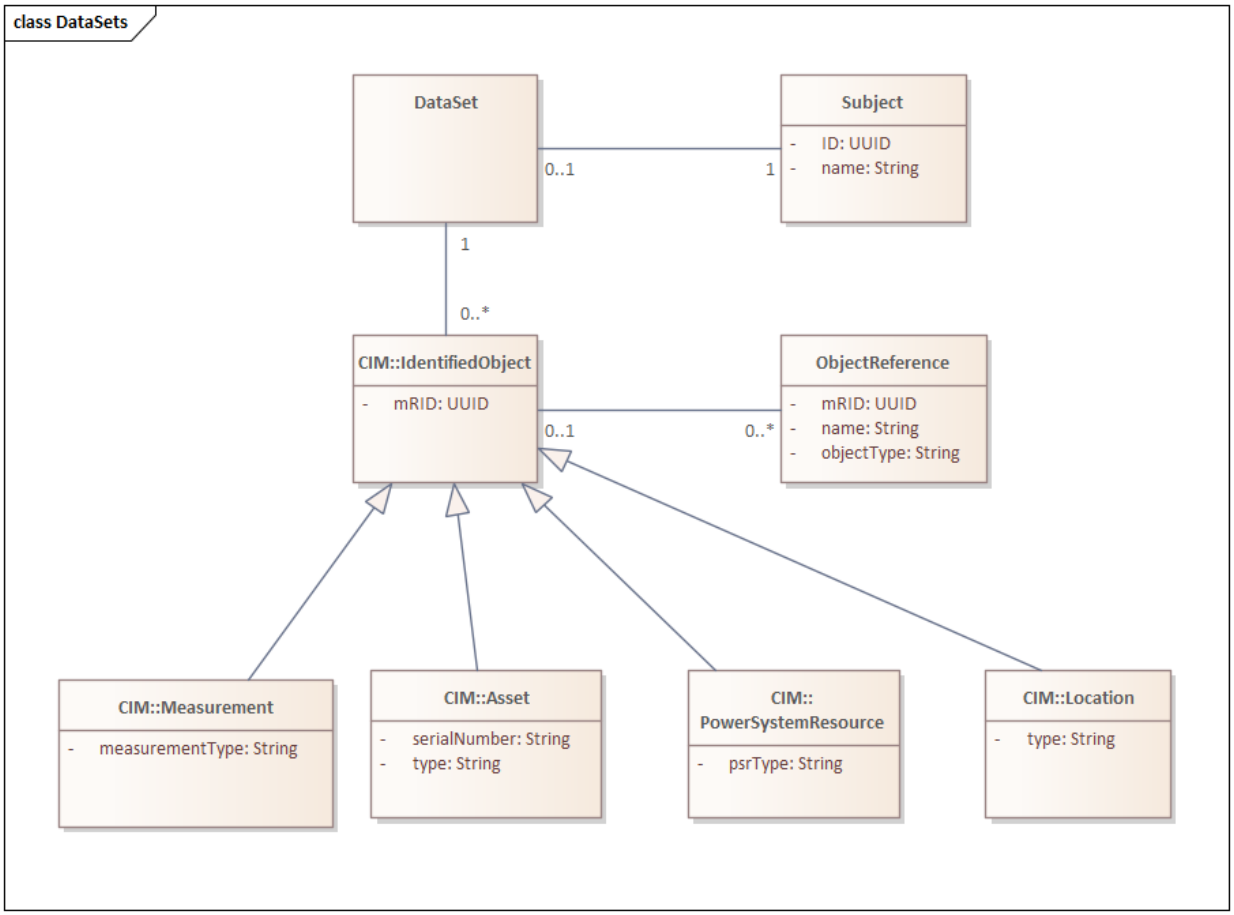

Figure 5-4: DataSet Objects

Figure 5-3 illustrates that:

- An IdentifiedObject (as defined by CIM) can be an object that is defined by the CIM, such as a PowerSystemResource, an Asset, or a Location.
- An IdentifiedObject can be defined with ObjectReferences (i.e., a UUDEX extension that genericizes the management of inter-object relationships), which provide references to potentially many other related objects.
- All objects can have a specified type, which may be more granular or diverse than the name of a CIM class.

Figure 5-5 shows a logical model of the data structures used to convey time series values from a CIM perspective.

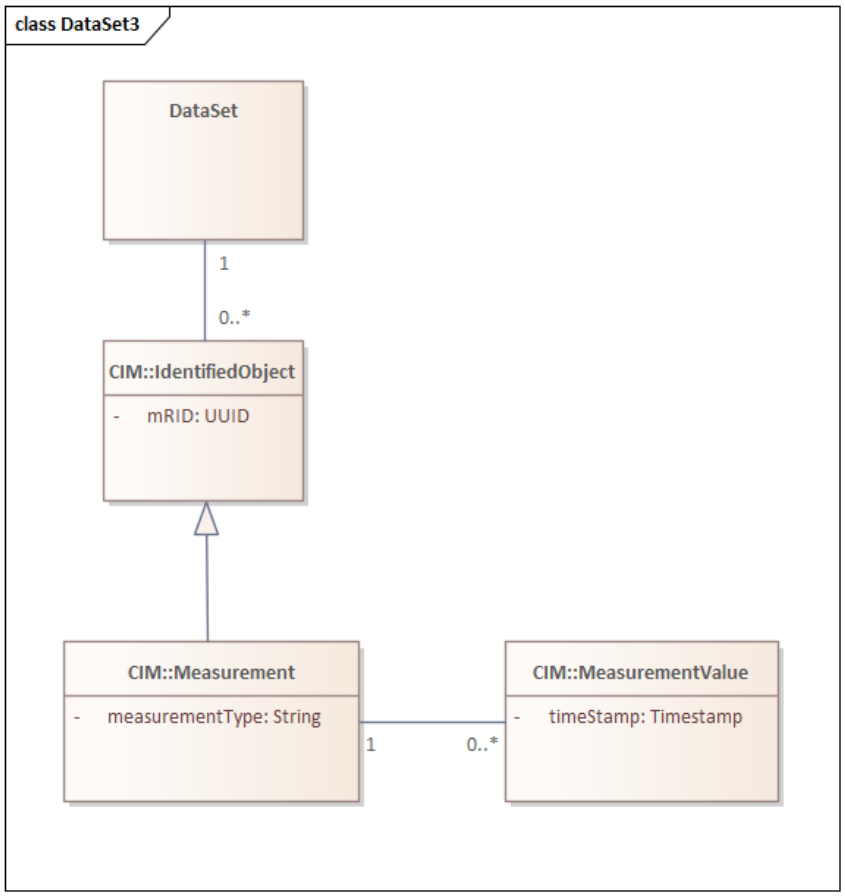

Figure 5-5: DataSets, Measurements and Values

As shown in Figure 5-5, a U-Data Set is used to define a container to convey a set of measurement values, where the container will be known as “MeasurementValues.” An important point is that the set of objects conveyed in a MeasurementValues container need only be a subset of the full set of objects defined in the U-Data Set. There are two important uses for this:

- It allows values to be reported on an exception basis, or with differing periodicities.
- It accommodates very large data sets that cannot be conveyed in a single message as might be due to limitations in message size imposed by the transport layer.

From the perspective of U-Data Elements, information may be conveyed as individual objects or within U-Data Sets. U-Data Sets allow for a collection of objects whose values or states are conveyed as a group. The purposes of this are to:

- Provide for efficiency in data transfer, avoiding replication of message overhead
- Provide a grouping mechanism that simplifies the definition of ACLs.

Following is a basic semantic description of U-Data Sets:

- U-Data Sets are used to collect and group U-Data Elements
- Specific U-Data Elements can be collected by a U-Data Set
- U-Data Set contents are defined by participants

- U-Data Sets are associated with a Subject
- There may be container types defined to convey the dynamic values that belong to a U-Data Set (e.g., MeasurementValues)

5.2 Physical

The purpose of this section is to introduce the physical models that support the infrastructure. This section focuses on those models that will be implemented by the UUDEX protocol.

5.2.1 UUDEX Subjects

UUDEX uses the term U-Subject as the basic message definition. This term represents the more generic term “topic” uses in many publish or subscribe implementations.

The name of a subject is defined using the source, U-Data Element Type, and group key. An example of this is the following:

```
<participant ID>/<UUDEX Data Element Type ID>/<group key>
```

It is envisaged that this subject name would be used to construct U-Subject names for use by specific message transports, noting that UUDEX is intended to be transport agnostic.

An important restriction for subject subscription is that subjects are not hierarchical in nature, where it is not permitted to subscribe to them using wildcards, as is often permitted with topics by many publish and subscribe messaging systems. U-Subscriptions to U-Subjects must be explicit.

U-Subject names will be used by:

- Interfaces for management of U-Subscriptions
- ACLs

It is important to note that U-Subjects will be used for more than the exchange of U-Data Elements. To support request and reply message patterns, U-Subjects will be defined by the U-Infrastructure for the following purposes:

- U-Administration requests
- UUDEX query requests
- Responses, where each U-Endpoint will create a U-Subjects that can be used by the U-Infrastructure to provide by synchronous and asynchronous response messages

5.2.2 UUDEX Data Sets

The following examples show how various information could be transmitted using UUDEX. The examples show U-Data Sets comprised of arrays of one or more U-Data Elements containing various formats of exchanged data. While mRIDs would typically be expressed as UUID, in these examples, common names are used for clarity.

The actual JSON field names are used in these examples but are not defined here. The complete explanation is contained in the *UUDEX Information Structures* document.

While the full names of JSON elements are used here, the *UUDEX Information Structures* document provides standardized abbreviations that can be used to make the individual messages shorter. That document also discusses which fields are required, which are optional, and the default values (if they are defined for a particular field) that are used if the fields are not specified.

These examples are expressed in human-readable format with indenting and whitespace. Actual transmissions would eliminate all extra space characters (unless they are contained within quoted strings) to further reduce the message length using a format known as “compact”.

The purpose of this section is to provide several examples of how U-Data Elements can be combined to create a U-Data Set. Note that the hash values are provided for illustration purposes only.

The following is an example of a U-Data Set that provides the real power (MW) measurements for a set of generators:


```
{
  "header":{
    "messageID":"d5d1c892-974a-11e9-b198-b0c090a8aac0",
    "noun":"powerSystemResource",
    "origin":"ACME",
    "source":"ems.acme.net",
    "subject":"ACME/Measurements/NorthGen",
    "timestamp":"2019-06-25-13:12:08.218024",
    "hashType":"SHA-256",
    "hash": "2db5d1f950aa93eec540b617986139eaf9337f54",
    "verb":"created"
  },
  "dataSet":{
    "dataElements":[
      {
        "powerSystemResource":{
          "powerSystemResourceMRID":"SysGen1",
          "measurements":[
            {
              "measurementMRID":"SysGen1MW",
              "measurementType":"VolAmp",
              "unitSymbol":"MW",
              "values":[
                {
                  "value":163.64455056,
                  "quality":{
                    "currentSource":"TELEMETERED",
                    "validity":[
                      "VALID"
                    ]
                  },
                  "timeStamp":{
                    "quality":"VALID",
                    "value":"2021-03-24 20:02:29"
                  }
                }
              ]
            }
```

```
      ]
    }
  },
  {
    "powerSystemResource":{
      "powerSystemResourceMRID":"SysGen2",
      "measurements":[
        {
          "measurementMRID":"SysGen2MW",
          "measurementType":"VolAmp",
          "unitSymbol":"MW",
          "values":[
            {
              "value":85.64455056,
              "quality":{
                "currentSource":"TELEMETERED",
                "validity":[
                  "VALID"
                ]
              },
              "timeStamp":{
                "quality":"VALID",
                "value":"2021-03-24 20:02:29"
              }
            }
          ]
        }
      ]
    }
  },
  {
    "powerSystemResource":{
      "powerSystemResourceMRID":"SysGen3",
      "measurements":[
        {
          "measurementMRID":"SysGen3MW",
          "measurementType":"VolAmp",
          "unitSymbol":"MW",
          "values":[
            {
              "value":72.64455056,
              "quality":{
                "currentSource":"TELEMETERED",
                "validity":[
                  "VALID"
                ]
              },
              "timeStamp":{
                "quality":"VALID",
                "value":"2021-03-24 20:02:29"
              }
            }
          ]
        }
      ]
    }
  }
```

```
      ]
    }
  }
```

The following is an example of a U-Data Set that provides the real power (MW) and reactive power (MVAR) measurements for a set of generators:


```
{
  "header":{
    "messageID":"d5d4d5d2-974a-11e9-a89e-b0c090a8aac0",
    "noun":"powerSystemResource",
    "origin":"ACME",
    "source":"ems.acme.net",
    "subject":"ACME/Measurements/NorthGen",
    "timestamp":"2019-06-25-13:12:08.238024",
    "hashType":"SHA-256",
    "hash":"21defd5889e178a5f188065990220aab174c2551",
    "verb":"created"
  },
  "dataSet":{
    "dataElements":[
      {
        "powerSystemResource":{
          "powerSystemResourceMRID":"SysGen1",
          "measurements":[
            {
              "measurementMRID":"SysGen1MW",
              "measurementType":"VolAmp",
              "unitSymbol":"MW",
              "values":[
                {
                  "value":163.64455056,
                  "quality":{
                    "currentSource":"TELEMETERED",
                    "validity":[
                      "VALID"
                    ]
                  },
                  "timeStamp":{
                    "quality":"VALID",
                    "value":"2021-03-24 20:02:29"
                  }
                }
              ]
            }, {
              "measurementMRID":"SysGen1MVAR",
              "measurementType":"VolAmpr",
              "unitSymbol":"MVAR",
              "values":[
                {
                  "value":7.455056,
                  "quality":{
                    "currentSource":"TELEMETERED",
                    "validity":[
                      "VALID"
                    ]
```

```
            },
            "timeStamp":{
              "quality":"VALID",
              "value":"2021-03-24 20:02:29"
            }
          }
        ]
      }
    ]
  }
},
{
  "powerSystemResource":{
    "powerSystemResourceMRID":"SysGen2",
    "measurements":[
      {
        "measurementMRID":"SysGen2MW",
        "measurementType":"VolAmp",
        "unitSymbol":"MW",
        "values":[
          {
            "value":85.64455056,
            "quality":{
              "currentSource":"TELEMETERED",
              "validity":[
                "VALID"
              ]
            },
            "timeStamp":{
              "quality":"VALID",
              "value":"2021-03-24 20:02:29"
            }
          }
        ]
      }, {
        "measurementMRID":"SysGen2MVAR",
        "measurementType":"VolAmpr",
        "unitSymbol":"MVAR",
        "values":[
          {
            "value":-11.5056,
            "quality":{
              "currentSource":"TELEMETERED",
              "validity":[
                "VALID"
              ]
            },
            "timeStamp":{
              "quality":"VALID",
              "value":"2021-03-24 20:02:29"
            }
          }
        ]
      }
    ]
  }
},
```

```
        {
          "powerSystemResource":{
            "powerSystemResourceMRID":"SysGen3",
            "measurements":[
              {
                "measurementMRID":"SysGen3MW",
                "measurementType":"VolAmp",
                "unitSymbol":"MW",
                "values":[
                  {
                    "value":72.64455056,
                    "quality":{
                      "currentSource":"TELEMETERED",
                      "validity":[
                        "VALID"
                      ]
                    },
                    "timeStamp":{
                      "quality":"VALID",
                      "value":"2021-03-24 20:02:29"
                    }
                  }
                ]
              }, {
                "measurementMRID":"SysGen3VAR",
                "measurementType":"VolAmpr",
                "unitSymbol":"MVAR",
                "values":[
                  {
                    "value":27.455056,
                    "quality":{
                      "currentSource":"TELEMETERED",
                      "validity":[
                        "VALID"
                      ]
                    },
                    "timeStamp":{
                      "quality":"VALID",
                      "value":"2021-03-24 20:02:29"
                    }
                  }
                ]
              }
            ]
          }
        }
      ]
    }
  }
```


The following is an example of a U-Data Set that provides the telemetered and estimated real power (MW) measurements for a set of generators:


```
{
  "header":{
    "messageID":"26d2cad4-982a-11e9-9ab5-b0c090a8aac0",
```

```
    "noun":"powerSystemResource",
    "origin":"ACME",
    "source":"ems.acme.net",
    "subject":"ACME/Measurements/NorthGen",
    "timestamp":"2019-06-26-15:50:41.890786",
    "hashType":"SHA-256",
    "hash":"45aa615644a6cc1e2cbafb5c9a6564b69f87a71d",
    "verb":"created"
  },
  "dataSet":{
    "dataElements":[
      {
        "powerSystemResource":{
          "powerSystemResourceMRID":"SysGen1",
          "measurements":[
            {
              "measurementMRID":"SysGen1MW",
              "measurementType":"VolAmp",
              "unitSymbol":"MW",
              "values":[
                {
                  "value":163.64455056,
                  "quality":{
                    "currentSource":"TELEMETERED",
                    "validity":[
                      "VALID"
                    ]
                  },
                  "timeStamp":{
                    "quality":"VALID",
                    "value":"2021-03-24 20:02:29"
                  }
                }, {
                  "value":163.775201,
                  "quality":{
                    "currentSource":"ESTIMATED",
                    "validity":[
                      "VALID"
                    ]
                  },
                  "timeStamp":{
                    "quality":"VALID",
                    "value":"2021-03-24 20:02:29"
                  }
                }
              ]
            }
          ]
        }
      },
      {
        "powerSystemResource":{
          "powerSystemResourceMRID":"SysGen2",
          "measurements":[
            {
              "measurementMRID":"SysGen2MW",
              "measurementType":"VolAmp",
```

```
          "unitSymbol":"MW",
          "values":[
            {
              "value":85.64455056,
              "quality":{
                "currentSource":"TELEMETERED",
                "validity":[
                  "VALID"
                ]
              },
              "timeStamp":{
                "quality":"VALID",
                "value":"2021-03-24 20:02:29"
              }
            },{
              "value":86.01128,
              "quality":{
                "currentSource":"ESTIMATED",
                "validity":[
                  "VALID"
                ]
              },
              "timeStamp":{
                "quality":"VALID",
                "value":"2021-03-24 20:02:29"
              }
            }
          ]
        }
      ]
    }
  },
  {
    "powerSystemResource":{
      "powerSystemResourceMRID":"SysGen3",
      "measurements":[
        {
          "measurementMRID":"SysGen3MW",
          "measurementType":"VolAmp",
          "unitSymbol":"MW",
          "values":[
            {
              "value":72.64455056,
              "quality":{
                "currentSource":"TELEMETERED",
                "validity":[
                  "VALID"
                ]
              },
              "timeStamp":{
                "quality":"VALID",
                "value":"2021-03-24 20:02:29"
              }
            }, {
              "value":73.66587,
              "quality":{
                "currentSource":"ESTIMATED",
```

```
                    "validity":[
                      "VALID"
                    ]
                  },
                  "timeStamp":{
                    "quality":"VALID",
                    "value":"2021-03-24 20:02:29"
                  }
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```


The following is an example of a U-Data Set that provides three time-series telemetered real power (MW) measurements for a single generator:


```
{
  "header":{
    "messageID":"26d2cad4-982a-11e9-9ab5-b0c090a8aac0",
    "noun":"powerSystemResource",
    "origin":"ACME",
    "source":"ems.acme.net",
    "subject":"ACME/Measurements/NorthGen",
    "timestamp":"2019-06-26-15:50:41.890786",
    "hashType":"SHA-256",
    "hash":"45aa615644a6cc1e2cbafb5c9a6564b69f87a71d",
    "verb":"created"
  },
  "dataSet": {
    "dataElements": [
      {
        "powerSystemResource": {
          "powerSystemResourceMRID": "SysGen1",
          "measurements": [
            {
              "measurementMRID": "SysGen1MW",
              "measurementType": "VolAmp",
              "unitSymbol": "MW",
              "values": [
                {
                  "value": 163.64455056,
                  "quality": {
                    "currentSource": "TELEMETERED",
                    "validity": [
                      "VALID"
                    ]
                  },
                  "timeStamp": {
                    "quality": "VALID",
                    "value": "2021-03-24 20:02:29"
                  }
```

```
                    },
                    {
                      "value": 163.73496723,
                      "quality": {
                        "currentSource": "TELEMETERED",
                        "validity": [
                          "VALID"
                        ]
                      },
                      "timeStamp": {
                        "quality": "VALID",
                        "value": "2021-03-24 20:02:31"
                      }
                    },
                    {
                      "value": 163.75234897,
                      "quality": {
                        "currentSource": "TELEMETERED",
                        "validity": [
                          "VALID"
                        ]
                      },
                      "timeStamp": {
                        "quality": "VALID",
                        "value": "2021-03-24 20:02:33"
                      }
                    }
                  ]
                }
              ]
            }
          }
        ]
      }
    }
```


A very important consideration is that current TASE.2 implementations do not require the above level of detail for the descriptions of points within a U-Data Set, rather require external configurations to describe the measurement values that are exchanged. The self-describing nature of the U-Message allows the measurement values to be exchanged in any order and only exchange those values which need to be exchanged.

Additionally, metadata that can describe different aspects of the same measurement (e.g., source of the measurement such as telemetered, estimated, etc. and selection of the measurement such as primary, backup, etc.) allow related values to be exchanged within the same U-Data Set.

Beyond that, metadata in the header can describe different intended uses of the entire U-Data Set (i.e., the context of the U-Message), such as U-Data Sets intended for production, test, or exercise to use the same U-Subjects and U-Subscriptions if desired.

6.0 Messaging Patterns

The purpose of this section is to describe the messaging patterns that are used by UUDEX for information exchanges.

6.1 Initialization

Initialization covers the process by which a new U-Participant joins a U-Instance. This will occur whenever an organization is authorized and configured to access the U-Instance.

The first step in initialization will involve the validation of the U-Participant's right to participate in the U-Instance and the creation of a corresponding UUDEX Identity Object (U-Identity Object) (i.e., a digital certificate issued by an appropriate certificate authority) that the U-Participant can use to authenticate its identity to other U-Participants within that U-Instance. This process is described in more detail in Section 9.1.1 and in the *UUDEX Security and Administration* document. The protocol by which a new U-Participant presents evidence of who they are and their right to participate will likely vary between U-Instances based on the policies of the administrator for that U-Instance. Communication of this evidence would likely not use UUDEX communications structures (since the new U-Participant would not yet have access to use that U-Instance). As such, the steps by which a UUDEX Identity Authority (U-Identity Authority) makes the determination to grant a U-Identity Object to a new U-Participant are beyond the scope of the UUDEX specification.

Once the U-Participant and its U-Endpoints have U-Identity Objects, those objects will be used, indirectly, every time a network session is established to or from one of the participant's registered devices. For all outbound sessions, the device will provide evidence based on secrets provided by the U-Identity Authority when the U-Participant's U-Identity Objects were created. The recipient of this connection request will use the corresponding U-Identity Object to validate that the evidence could only have come from the asserted device, thus authenticating the identity of the U-Participant. Similarly, in all connection requests to one of the U-Participant's U-Endpoints, similar evidence will be returned to the requester so the identity of the U-Participant can be validated.

6.2 UUDEX Subscription Enrollment

U-Subscription enrollment is the process by which a U-Consumer client expresses interest in receiving new information from a particular U-Subject in a given U-Instance. To establish a U-Subscription, the U-Consumer makes a request with the full name of the U-Subject and may include additional information used in processing the U-Subscription. The U-Subscription request would also indicate whether a notification message should be sent to the U-Consumer when new data is available (i.e., when a U-Message is published to a U-Subject by a U-Producer), or whether the data should be pushed to the U-Consumer as soon as they are available. The U-Server authenticates the identity of the requesting U-Consumer and validates that they have access rights to consume content from the named U-Subject. It can also evaluate other aspects of the request to ensure it is willing to support other parameters of the request. If the U-Server is willing to honor the requested U-Subscription, it responds with a message indicating the request was successful. Otherwise, it responds with a message indicating the U-Subscription request has been rejected.

Successfully created U-Subscriptions are assigned a unique identifier, which is returned to the U-Consumer in the response message. This identifier is used by the U-Consumer to manage the U-Subscription, allowing them to pause and resume delivery, or delete the U-Subscription.

U-Subscriptions persist until such time as they are explicitly terminated. U-Subscription termination can occur because the U-Consumer sends a request to delete the specific U-Subscription or because the identity associated with the U-Subscription is revoked. In the former case, the U-Subscription holder responds to the request with a message indicating that their request to delete the U-Subscription was successful. In the latter case, no such message is possible since the revocation of a participant's identity prevents U-Messages from being sent to them.

A U-Subscription may be deleted if the associated U-Subject's ACL changes in such a way as to preclude delivery of any data to a U-Consumer. However, even if the U-Subscription is not deleted, no data would be leaked to the U-Consumer since fulfillment of a U-Subscription is required to apply ACLs before U-Data Elements are queued or delivered. Thus, a U-Subject would not allow data to be sent to a U-Consumer without the access rights to consume data from a given U-Subject.

A U-Subscription may also be terminated if the U-Server is no longer willing to fulfill the U-Subscription. (For example, the U-Server is no longer willing to support the U-Subject to which the U-Consumer has subscribed.) In this case, the U-Server must explicitly inform the U-Consumer that the U-Subscription has been terminated.

U-Consumers may request that a given U-Subscription be paused or resumed. These requests use the same request-response process used when establishing or deleting a U-Subscription.

6.3 UUDEX Subscriptions

Once a U-Subscription is established, the U-Server associated with that U-Subscription is responsible for ensuring that it is honored for the lifetime of the U-Subscription. Specifically, whenever a new U-Data Element is added to the subscribed U-Subject, the U-Server must perform any necessary processing and, barring reasons for blocking publication, it must prepare the U-Data Element for delivery to the U-Consumer.

Delivery can take one of two forms: notifying the U-Consumer that data are available or queuing the data for direct delivery to the U-Consumer. Both of these mechanisms involve queuing a message for delivery to the U-Consumer; the notification message being smaller than the data delivery message. As noted in the preceding section, the subscriber can request a particular type of delivery, although the U-Server might choose to only accept U-Subscriptions of a particular delivery type.

For delivery notification, the U-Data Element is stored by the U-Server until such time that the U-Consumer subscriber contacts it for delivery of the queued U-Data Elements. The U-Consumer specifies the unique message identifier of the U-Data Element contained it the notification U-Message in the request.

In response to this request, the U-Server will respond by queueing a message containing the referenced U-Data Element for delivery to the U-Consumer.

(Note – the U-Consumer could also query available U-Data Elements – see Section 6.5.1.)

For direct delivery, as soon as a U-Data Element is determined to apply to a U-Subscription, the U-Server queues a message to the U-Consumer containing the U-Data Elements. Such delivery of U-Data Elements might be desirable in the cases of high-priority U-Data Elements, or in cases where the subscriber invariably wants to receive all U-Data Elements that apply to their U-Subscription.

Determining whether subscription fulfillment results in a notification or U-Data Element delivery can be part of the process of establishing a U-Subscription.

6.4 Publish and Subscribe

Publish and subscribe messaging is the primary means by which information is conveyed within the U-Infrastructure. All publish and subscribe messaging uses a U-Subject as a virtual address. The two primary uses of publish and subscribe messaging are to:

1. Convey time series data that has been defined by a U-Data Set.
2. Convey events as related to the creation, update, or deletion of any other type of U-Data Element.

6.4.1 Time Series

Figure 6-1 describes the sequence of message exchanges related to time series data. A U-Data Set is defined that describes a set of MeasurementValues that may be periodically published. In this example, transmissions of MeasurementValues are performed on a periodic basis. However, the transmission can also be defined to be on a report-by-exception basis, as is common for device status changes.

This sequence shows a UUDEX producer creating an ACL and a U-Subject to transfer measured values to subscribers. The ACL allows subscription by both UUDEX Endpoint A and UUDEX Endpoint B, and both UUDEX Endpoint A and UUDEX Endpoint B know the name of the U-Subject they wish to access.

Both UUDEX Endpoint A and UUDEX Endpoint B establish U-Subscriptions to the U-Subject. As part of the U-Subscription creation, the ACLs are validated to ensure that the subscriber has access to the U-Subject. When a U-Data Element is published to the U-Subject, the U-Data Element is separately queued to both UUDEX Endpoint A and UUDEX Endpoint B.

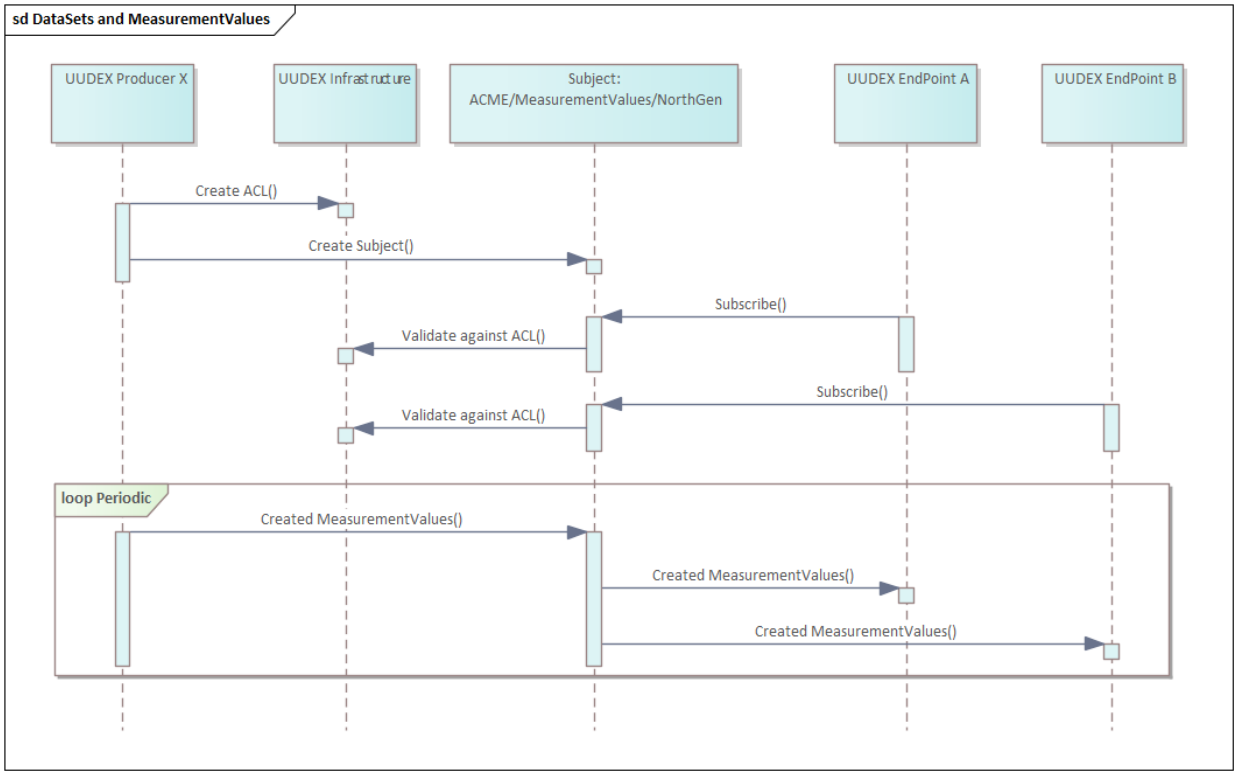

Figure 6-1: UUDEX Data Sets and MeasurementValues

6.4.2 Events

Figure 6-2 describes the sequence of messages for the reporting of events. An example of an event could be the creation of some type of structured or unstructured document, where the document is some defined type of U-Data Element. The posting of a Model or updates would also be an event.

Other verbs that might be used for event messages include: Changed, Deleted, and Closed.

In this sequence the Publisher creates the U-Subject and establishes its ACL. Subscriber A and Subscriber B query the available U-Subjects, and, assuming the U-Subject ACL allows discovery access, returns the U-Subject as available for subscription. Both Subscriber A and Subscriber B establish U-Subscriptions to the U-Subject. When a U-Data Element is published by the Publisher to the U-Subject, a copy of the U-Data Element is queued for delivery to both Subscriber A and Subscriber B.

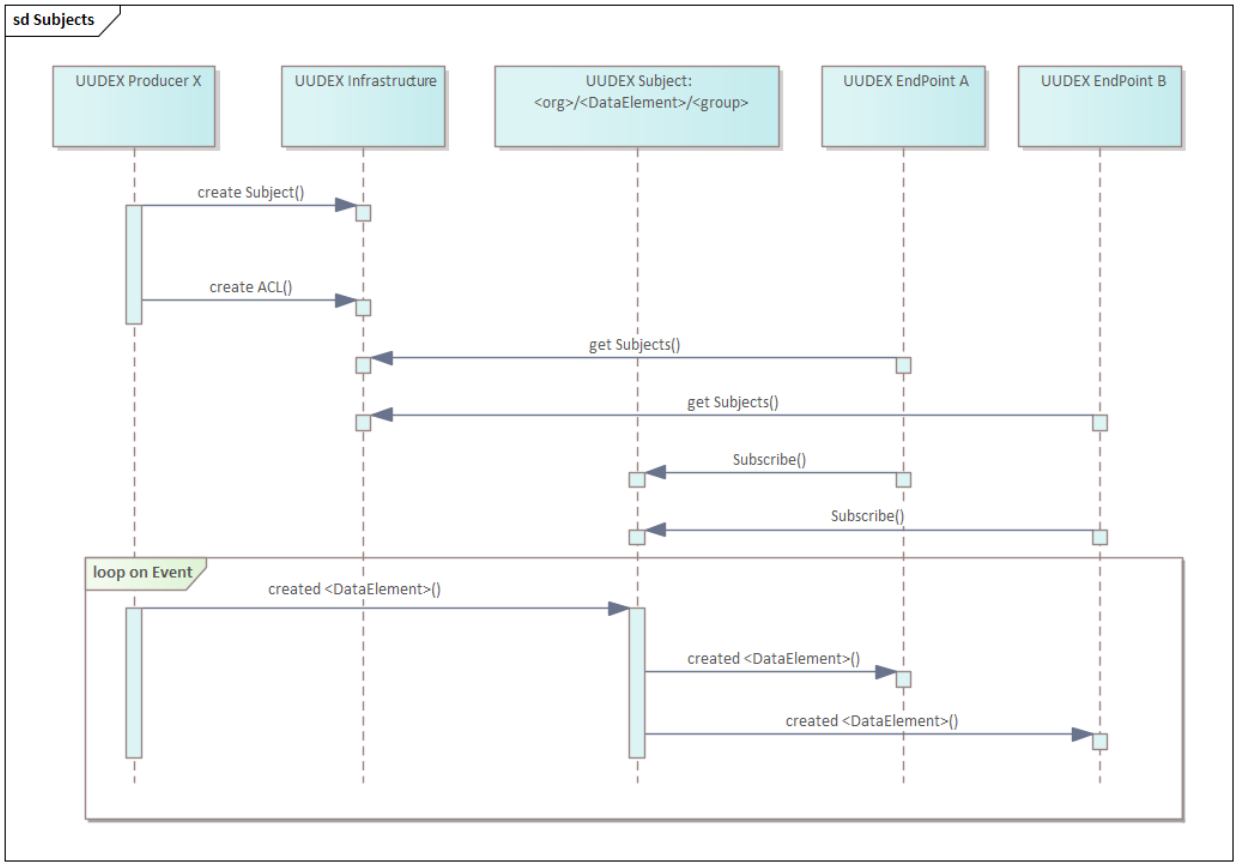

Figure 6-2: Publishing Events

6.5 Request and Reply

The implementation of request and reply messaging can be done in different ways, largely depending upon the design of the U-Infrastructure (e.g., centralized or federated).

6.5.1 Query

There are two types of query (get) requests:

- Queries for information from the U-Infrastructure
- Queries for historical data that was conveyed through UUDEX

The message sequence diagram shown in Figure 6-3 shows a distributed request and reply using publish and subscribe messaging. This pattern allows for a federated implementation of UUDEX Directories.

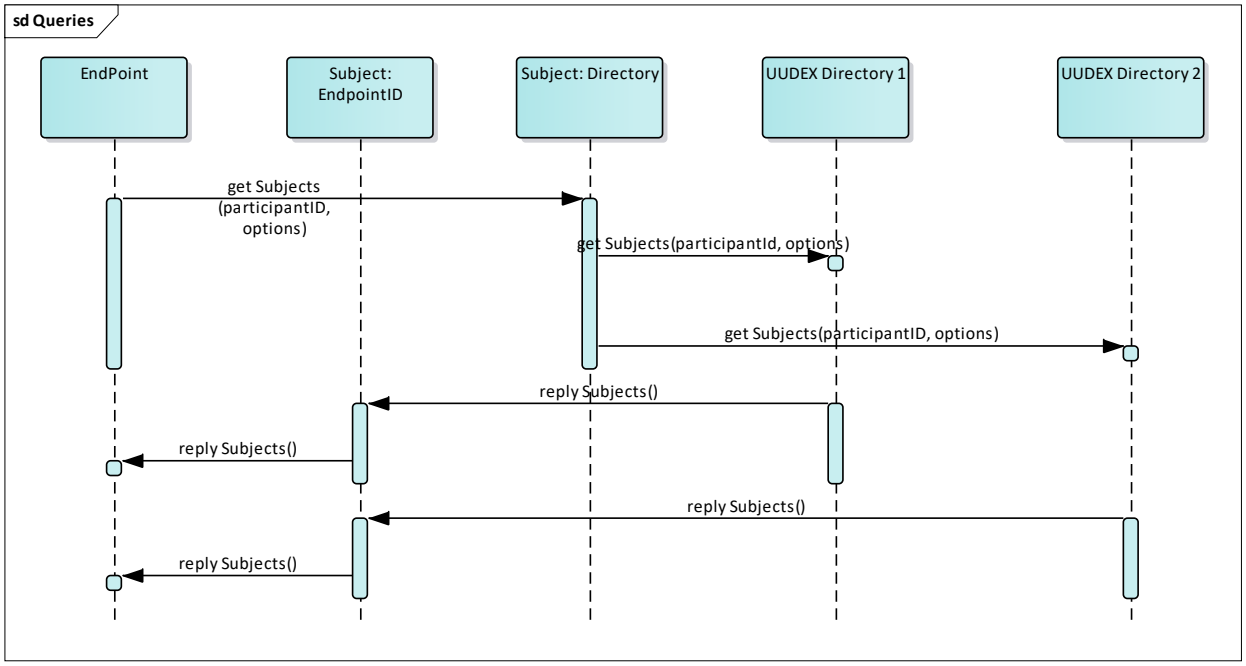

Figure 6-3: Distributed Request and Reply

6.5.2 Transactions

Transactions are used to update information in the U-Infrastructure.

Note – There is a dependency upon the U-Infrastructure design (e.g., federation, replication).

6.6 Persistence

Persistence is used to allow for management of the U-Infrastructure, and to allow for future access of information that was conveyed through UUDEX. In all cases, the definition of a U-Subject by the owning U-Participant defines whether or not the information should be retained by the U-Infrastructure, and for what length of time it should be retained.

In all cases, U-Messages published to U-Subjects are retained until all active U-Subscriptions have been fulfilled (i.e., the U-Messages have been transmitted to all subscriber clients). Since there are no specific performance requirements on how often a U-Subscriber client requests data, U-Messages may be retained in the U-Subscription queue for an indefinite period of time.

In some cases, as specified when a U-Subject is created, U-Messages may be retained by the U-Infrastructure after all U-Subscriptions have been fulfilled. These U-Messages are available for retrieval following a query against keywords contained in the U-Message header returns the U-Message identifier.

6.6.1 Time series

Time series data items are defined by U-Data Sets, where values are conveyed as MeasurementValues on a periodic or exception basis.

6.6.2 Snapshots

Snapshots are a special case of time series data, where a set of data items were collected at a common point in time.

6.6.3 History

History involves the storage of U-Data Elements that were conveyed through UUDEX.

6.7 Message Acknowledgement

UUDEX supports the ability for a U-Publisher to be notified when a U-Message has been consumed and retrieved by U-Subscriber. This is accomplished by the U-Publisher specifying the `ackRequired` parameter in the U-Message header to true and specifying a U-Subject to receive the responses in the `replyAddress` parameter.

When the U-Subscriber retrieves the U-Message, it checks to see if the `ackRequired` parameter is set to true, and if so, it shall generate a U-Message with the verb "`ACKNOWLEDGED`", the `correlationID` parameter of the U-Message set to the `messageID` of the retrieved message and sends the U-Message to the U-Subject specified in the `replyAddress` parameter. This processing may be contained in the consume client code on the U-Subscriber to ensure consistent implementation of the responses. The initial U-Publisher is responsible for monitoring the U-Subject specified in the `replyAddress` parameter for responses and should expect an individual U-Message from each U-Subscriber that consumes the original U-Message. This processing is expected to be used when submitting occasional reports, such as incident reports, where the U-Publisher has a need to confirm that the report has been retrieved, and which participants have retrieved the report. It is not expected that high-volume, periodic telemetry data will make use of this feature.

6.8 Message Tamper Detection Processing

UUDEX supports the ability to generate a secure hash of the payload component of the message to ensure that it has not been tampered while in transit from a U-Publisher to a U-Subscriber. Although U-Messages are protected by TLS encryption during transit from the U-Publisher client to the U-Server, and from the U-Server to the U-Subscriber client, they are potentially vulnerable to tampering while resident in the U-Server. Messages in persistent storage at the U-Server should be stored in encrypted and protected storage, but vulnerabilities in that processing, or malicious or corrupt users with access to the encrypted storage at the U-Server could potentially modify the messages while stored.

UUDEX allows for an optional hash of the message to be generated by the U-Publisher and communicated along with the message, allowing the U-Subscriber to re-compute the hash of the received message and compare it with the hash located in the U-Message Header. The particular hash algorithm is specified in the `hashType` parameter in the U-Message Header, and the hash value itself is stored in the `hash` parameter. The hash is computed using the contents of the `"dataSet":{}` structure (the entire structure between the braces, not including them), when expressed in "compact" form, i.e., a continuous stream of characters without line breaks or other white space for readability, except for strings that are enclosed in quotes. The hash is calculated using the method or algorithm specified in the `hashType` parameter (e.g., SHA-256).

The U-Publisher must generate the hash value and supply it in the U-Message header prior to publishing the message.

If the transmitted and generated hash values at the U-Subscriber do not match, a local alarm should be generated to indicate to the U-Participant Administrator that an error has occurred, and the message should not be processed. If message acknowledgement (Section 6.7) is specified, a "NACKED" message should be generated and returned to the U-Publisher.

If message tamper detection is not desired, the hashType parameter should be blank or not specified.

6.9 Message or dataElement Encryption Processing

UUDEX supports the capability to encrypt entire U-Messages or individual dataElements within a U-Message to ensure that the contents are not tampered while in transit, nor can its contents be read while it is resident in the persistent storage at the U-Server.

This is indicated by specifying an encryption method in the encryption parameter in either the U-Message Hader (for whole-message encryption), or for an individual dataElement (for dataElement encryption). Specific encryption methods are not specified by the UUDEX specification and must be mutually agreed to by the U-Publisher and all U-Subscribers that will consume the U-Message. If additional parameters are required to support the encryption processing, they may be specified using the *properties* parameter to specify one or more additional properties. Encryption key management is not specified by the UUDEX specification but must also be mutually agreed to by the U-Publisher and all U-Subscribers that will consume the U-Message. The U-Identity Authority may be able to offer this as a non-standard service.

The U-Publisher must encrypt the message or dataElement prior to publishing the message.

If the U-Message or dataElement of the U-Message cannot be successfully decrypted, a local alarm should be generated to indicate to the U-Participant Administrator that an error has occurred, and the message should not be processed. If message acknowledgement (Section 6.7) is specified, a "NACKED" message should be generated and returned to the U-Publisher.

If message or dataElement encryption is not desired, the encryption parameter should be blank, set to "NONE", or not specified.

7.0 Message Structures

All UUDEX interfaces are based upon message sequences. The message sequences implement a specific information exchange pattern. Each message uses a common structure. This message structure is called a “common message envelope” or CME.

UUDEX borrows from the IEC 61968-100 standard, which was defined to standardize CIM-based information exchanges using a variety of transport technologies. Where IEC 61968-100 focused on the use of XML for the formatting of a message envelope, UUDEX instead realizes the IEC 61968-100 constructs using JSON (IETF RFC 4627).

7.1 Logical

IEC 61968-100 defines a message structure that primarily has a header and a payload. The header contains information (metadata) that describes the nature and general content of the message. The payload contains the U-Data Elements being conveyed by the message, as well as optional metadata that described the nature of the U-Data Element. This is shown in Figure 7-1.

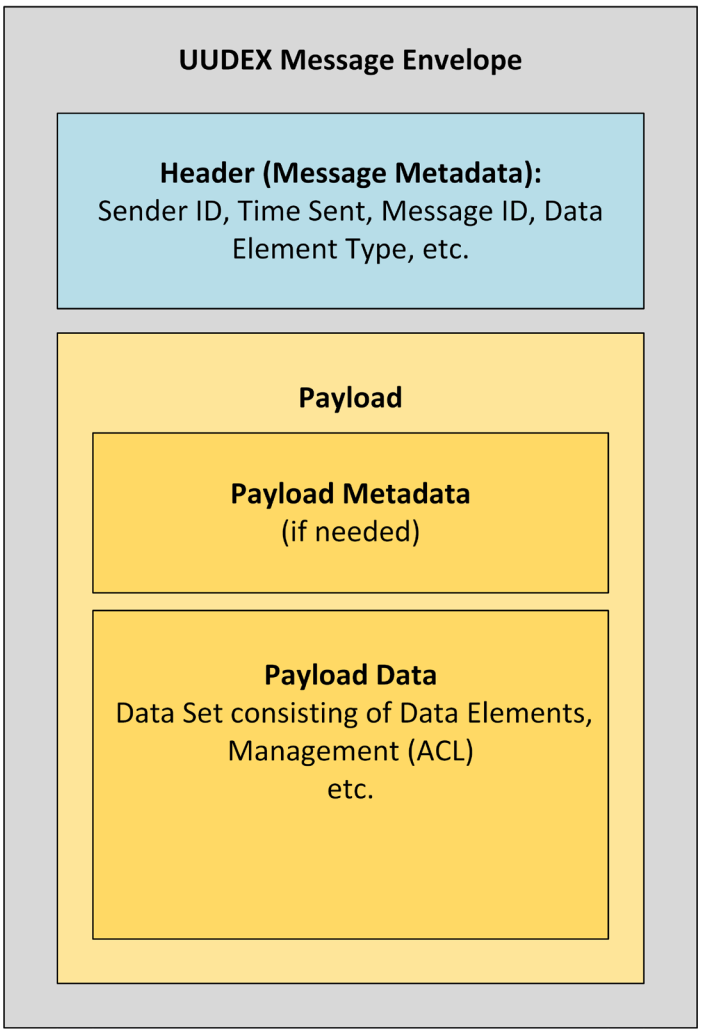

Figure 7-1: Message Structure Overview

This structure is described by the UML diagram shown in Figure 7-2, noting that some of the classes are defined in subsequent sections of this document.

The header (i.e., the message metadata) minimally contains the following elements:

- *MessageID*: The MessageID is a unique UUID used to identify the message.
- *Noun*: The noun defines the type of the U-Data Element. Nouns are extensible, with a base set being defined by this specification. Nouns are case insensitive. Nouns that are unknown by the consumer can be ignored or logged as exceptions, as the set of nouns can be expected to grow over time.
- *Verb*: The verb refines the type of action as related to the information exchange pattern. Verbs are case insensitive.
  - The verbs for events include:
    - “created”,
    - “changed”,
    - “deleted”,

    - “cancelled”,
    - “closed”.
  - The verb used for queries is “get”.
  - The verbs used to initiate transactional requests are
    - “create”,
    - “change”,
    - “delete”,
    - “cancel”, and
    - “close”.
  - The verb used for response messages that are the result of transactions or queries is “reply”.
- *Subject*: The U-Subject to which the message is being published.
- *Source*: The source is the U-Participant that published the message.
- *Timestamp*: The timestamp is an ISO-8601 time string that describes when U-Endpoint published the message.

The header may also include optional values that include:

- *CorrelationID*: The CorrelationID is used on response messages, using the MessageID from the initiating request (query or transaction) message.
- *Origin*: The Origin is the UUDEX defined identifier for the U-Participant that is sending a message.
- *Context*: Context is used to logically segregate messages that might be used for production, testing, or other purposes.
- *User*: This is the user that is responsible for the initiation of the information exchange.
- *Comment*: A comment is text entered for documentation or diagnostic purposes.
- *Hash*: The cryptographic hash of the message payload.
- *Properties*: Any JSON object.

Additional fields and further descriptions are provided in the *UUDEX Information Structures* document.

The payload will contain:

- *Metadata*: An optional set of information that describes the dataElement.
- *dataSet containing dataElements*: The data object being conveyed as per payload formatting rules. It could be null. U-Data Elements are defined by U-Data Element Types which specify metadata, required and optional fields, and how the contents of the U-Data Element are to be expressed in the message. All U-Data Elements are natively expressed using JSON formatting rules, although the contents may be specified using alternate expressions as specified by the formatting specified in the U-Data Element Type specification.

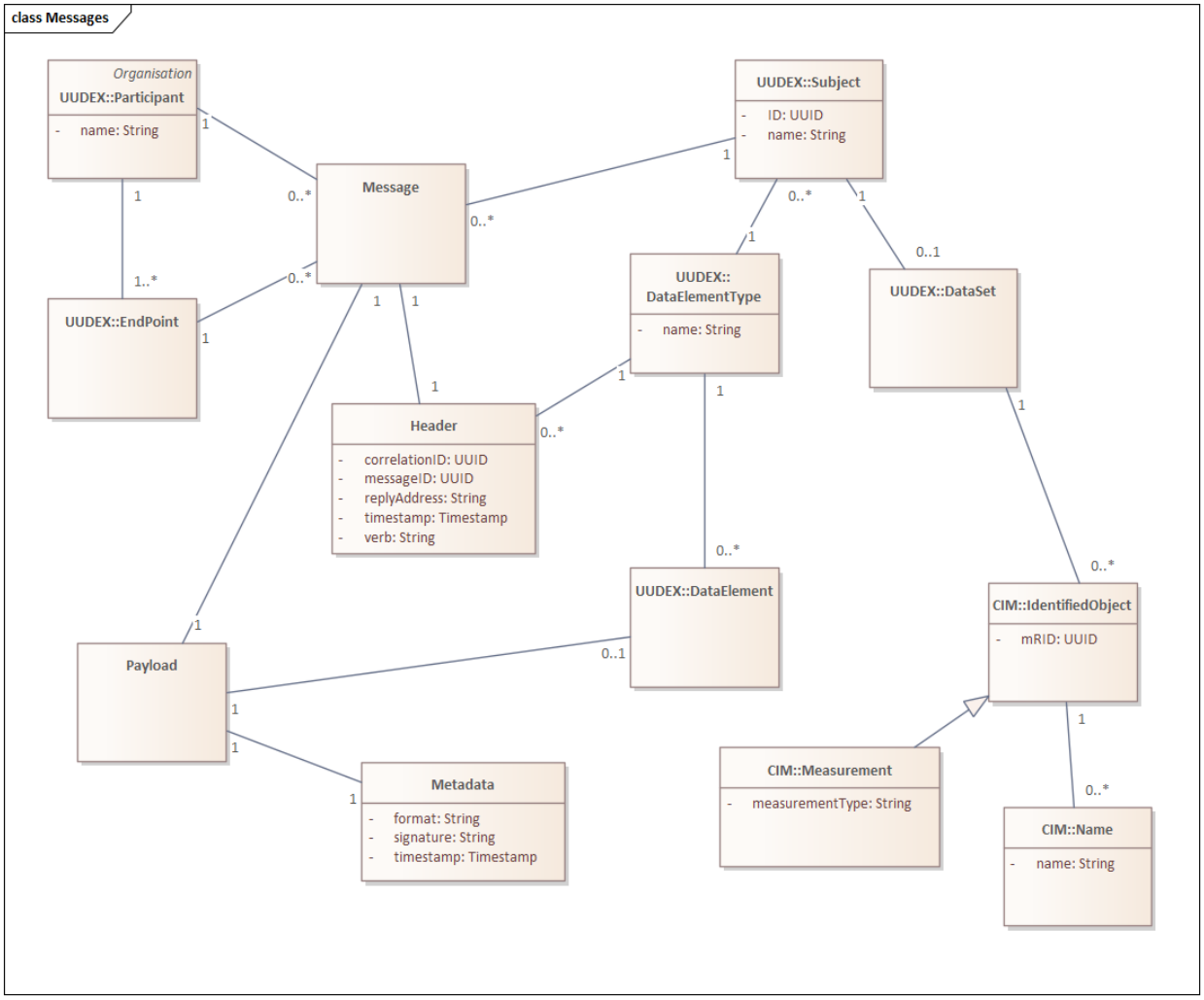

Figure 7-2: Message Structure Logical View

7.2 Physical

The purpose of this section is to describe the message structures and their usage. Following is an annotated example of the message structure, where optional elements are identified. Note that elements appear in alphabetical order within any portion of the message structure.

```
{
  "header": {
    "comment": "opt:<comment>",
    "context": "opt:<context>, examples: PROD, TEST",
    "correlationID": "opt:<correlation ID, a UUID>",
    "hash": "<hash of the dataElement>",
    "messageID": "<UUID>",
    "noun": "<noun>",
    "origin": "<participant ID>",
    "properties": "opt:<JSON properties>",
    "tags": "opt:<query keywords>",
    "source": "<source>",
    "timestamp": "<ISO 8601 time string>",
```

```
    "user": "opt:<user>",
    "verb": "<verb>",
  },
  "dataSet": {
    "dataElement": "opt:<UUDEX Data Element>",
    }
  }
}
```

A key point is to differentiate the message header from the U-Data Element, as the message conveys a U-Data Element. For example, metadata may be included in both the message header and U-Data Elements. The message header metadata has a timestamp that identifies when the message was generated, while the U-Data Element metadata may have a timestamp that identifies when the U-Data Element was created.

The header and payload metadata sections both have an optional “properties” element. This can be used to “tag” the message or payload with additional information if not already defined.

Additional elements that can be provided in the header metadata include the following:

- replyAddress
- asyncReplyFlag
- ackRequired
- expiration

In addition to a payload, a message can also include a response object that is used for response (i.e., verb=“reply”) messages that are the consequence of a request message.

7.2.1 Basic Event Message

The following example shows a basic event message, where the U-Data Element is a simple text string, which is largely for purposes to illustrate that the payload can convey many types of information, so long as the information can be encoded in a string.

```
{
  "header": {
    "hash": "1e2bfb102d67e40af54551014dc9c3b69ef7fe18",
    "messageID": "041c5d40-8780-11e9-b7cd-b0c090a8aac0",
    "noun": "someTypeOfObject",
    "origin": "ACME",
    "source": "ems.acme.net",
    "timestamp": "2019-06-05 10:52:30.604217",
    "verb": "created"
  },
  "dataSet": {
    "dataElement":
      "someTypeOfObject" {
        "something": "something, anything",
      }
    }
  }
}
```

```
}
```

The following example shows a more typical message payload, where the U-Data Element is a JSON object.

```
{
  "header": {
    "context": "PROD",
    "hash": "46e56145e7e920d7bed852f91b9d5c5abf48a12c",
    "messageID": "041de3e2-8780-11e9-b06c-b0c090a8aac0",
    "noun": "anotherObjectType",
    "origin": "ACME",
    "source": "ems.acme.net",
    "timestamp": "2019-06-05 10:52:30.614217",
    "verb": "created"
  },
  "dataSet": {
    "dataElement": {
      "anotherObjectType": {
        "name": "name",
        "value": "value"
      }
    }
  }
}
```

The following example shows a fragment of a message payload, where the U-Data Element is a compressed and encoded object, which could be the result of an object such as an XML object, PDF file or binary object. In this case an OE-417 PDF[1] file is used:

```
{
  "header": {
    "context": "PROD",
    "hash": "46e56145e7e920d7bed852f91b9d5c5abf48a12c",
    "messageID": "041de3e2-8780-11e9-b06c-b0c090a8aac0",
    "noun": "OE-417",
    "origin": "ACME",
    "source": "aserver.acme.net",
    "timestamp": "2019-06-05 10:52:30.614217",
    "verb": "created"
  },
  "dataSet": {
    "dataElements": (
      "OE-417":{
        "format":"PDF",
        "name":"OE-417.pdf",
        "encoding":"BASE64",
        "compression":"GZIP",
        "contents":
"JVBERi0xLjYNJQ0KMTQgMCBvYmoNPDwvTGluZWFyaXplZCAxL0wgMzI4NDkvT
yAxNi9FIDI3NzgxL04gMS9UIDMyNTQ2L0ggWyA0NTUgMTUzXT4+DWVuZG9iag0
gICAgICAgICAgICAgICAgICANCjIxIDAgb2JqDTw8L0Rl Y29kZVBhcm1zPDwvQ
29sdW1ucyA0L1ByZWRpY3RvciAxMj4+L0ZpbHRlci9GbGF0ZURlY29kZS9JRFs
```

[1] Note – the entire file is not shown in this example

```
8QkJBODdEMEI0MDY0MzE0QjkxOUI5OUY3MjJCQ0EzMjY+PEIzNEE2NEEzMUJFM
kNDNDA5QjhGNDFFREREMjJEOUNBPl0vSW5kZXhbMTQgMTFdL0luZm8gMTMgMCB
SL0xlbmd0aCA1NC9QcmV2IDMyNTQ3L1Jvb3QgMTUgMCBSL1NpemUgMjUvVHlwZ
S9YUmVmL1dbMSAyIDFdPj5zdHJlYW0NCmhiYmQQYGBiYAoAEgxTBC4QcRBEAlg
gMVVhHkgJAyN/AQIMAAcNCmVuZHN0cmVhbQ1lbmRvYmoNc3RhcnR4cmVmDQowD
QolJUVPRg0KICAgICAgDQoyNCAwIG9iag08PC9DIDczL0ZpbHRlci9GbGF0ZUR
lY29kZS9JIDk1L0xlbmd0aCA2OS9TIDM4Pj5zdHJlYW0NCmhiYGBgZWBgZgAME
VkYOBoEWGEXAz9hMhBwKmlVcA0sDAx1bFBRWQADADwWBQ0KZW5kc3RyZWFtDWV
uZG9iag0xNSAwIG9iag08PC9MYW5nKABFAE4ALQBVAFMpL01hcmtJbmZvPDwvT
WFya2VkIHRydWU+Pi9NZXRhZGF0YSAyIDAgUi9QYWdlTGF5b3V0L09uZUNvbHV
tbi9QYWdlcyAxMiAwIFIvU3RydWN0VHJlZVJvb3QgNiAwIFIvVHlwZS9DYXRhb
G9nPj4NZW5kb2JqDTE2IDAgb2JqDTw8L0NvbnRlbnRzIDE3IDAgUi9Dcm9wQm9
4WzAuMCAwLjAgNjEyLjAgNzkyLjBdL011ZGlhQm94WzAuMCAwLjAgNjEyLjAgN
zkyLjBdL1BhcmVudCAxMiAwIFIvUmVzb3VyY2VzPDwvRm9udDw8L1RUMCAyMyA
.
.
.
+PEIzNEE2NEEzMUJFMkNDNDA5QjhGNDFFREREMjJEOUNBPl0vSW5mbyAxMyAwI
FIvTGVuZ3RoIDQ4L1Jvb3QgMTUgMCBSL1NpemUgMTQvVHlwZS9YUmVmL1dbMSA
yIDFdPj5zdHJlYW0NCmhiYgACJlYGJgYqICFQCWIIWUQ9IBEHG0gwMgAEGAAFI
w0KZW5kc3RyZWFtDWVuZG9iag1zdGFydHhyZWYNCjExNg0KJSVFT0YNCg==",
```


```
  }
}
```

7.2.2 Transactional Request and Reply

The following is an example request message, where the U-Data Element (an object of type "someObjectClass") is a JSON object that is to be created as a consequence of the transaction. Note that the U-Data Element can be any valid JSON object. A common use of this pattern would be for administrative interfaces, where the actual U-Data Elements could include ACLs, U-Data Sets, U-Subjects and U-Endpoints.

```
{
  "header": {
    "hash": "badb80461432ad3cfc44f6f1f9531f76b969dca9",
    "messageID": "041f6a80-8780-11e9-827e-b0c090a8aac0",
    "noun": "someObjectClass",
    "origin": "ACME",
    "source": "ems.acme.net",
    "timestamp": "2019-06-05 10:52:30.624217",
    "verb": "create"
  },
  "dataSet": {
    "dataElement": {
      "someObjectClass": {
        "list": [
          1,
          2,
          3,
          4,
          5
        ],
        "name": "name",
        "value": "value"
      }
    }
```

```
    }
}
```

The following is an example of the response message that could be returned. Note that the `correlationID` provided is the `messageID` from the initiating request message.

```
{
  "header": {
    "correlationID": "041f6a80-8780-11e9-827e-b0c090a8aac0",
    "messageID": "0420f122-8780-11e9-a365-b0c090a8aac0",
    "noun": "objectType",
    "origin": "ACME",
    "source": "other.acme.net",
    "timestamp": "2019-06-05 10:52:30.634217",
    "verb": "reply"
  },
  "reply": {
    "objectType": {
      "response": "OK"
    }
  }
}
```

7.2.3 Query Request and Reply

The following is an example of a request and reply pattern for a data query. A common use of this pattern would be for discovery. Notice the use of tags to provide qualifiers for the query, and that no payload is provided.

```
{
  "header": {
    "messageID": "6a71a6c0-8782-11e9-8583-b0c090a8aac0",
    "noun": "objectType",
    "origin": "ACME",
    "tags": {
      "assetID": 53737,
      "endHour": 24,
      "startHour": 1
    },
    "source": "ems.acme.net",
    "timestamp": "2019-06-05 11:09:41.284217",
    "verb": "get"
  }
}
```

The response message would be in the following form. Note the use of the correlation ID, and the verb of “reply.” There is also a response object that could be used to convey errors.

```
{
  "header": {
    "correlationID": "6a71a6c0 8782 11e9 8583 b0c090a8aac0",
    "hash": "badb80461432ad3cfc44f6f1f9531f76b969dca9",
    "messageID": "6a732d62-8782-11e9-ab84-b0c090a8aac0",
    "noun": "objectType",
    "origin": "ACME",
```

```
    "source": "other.acme.net",
    "timestamp": "2019-06-05 11:09:41.294217",
    "verb": "reply"
  },
  "dataSet": {
    "dataElement": {
      "objectType": {
        "list": [ 1,
          2,
          3,
          4,
          5 ],
        "name": "name",
        "value": "value"
      }
    }
  },
  "reply": {
    "response": "OK"
  }
}
```


Following is a message used to convey measurement values that are defined for a given data set.


```
{
  "header": {
    "hash": "cbc968077d500dfea328c68bcdb90795e8e3f70e",
    "messageID": "376038f0-8953-11e9-93aa-b0c090a8aac0",
    "noun": "MeasurementValues",
    "origin": "ACME",
    "source": "ems.acme.net",
    "subject": "ACME/Measurements/NorthGen",
    "timestamp": "2019-06-07 18:36:51.585457",
    "verb": "created"
  },
  "dataSet": {
    "dataElement": {
      "MeasurementValues": {
        "GenMW1": {
          "Q": 0,
          "TS": 1559932611.5854576,
          "V": 54.357
        },
        "GenMW2": {
          "Q": 0,
          "TS": 1559932611.5854576,
          "V": 0.742
        },
        "GenMW3": {
          "Q": 0,
          "TS": 1559932611.5854576,
          "V": 1.634
        }
      }
    }
  }
```

```
        }
    }
```

7.2.4 Message Interoperability

The JSON message specification forms the basis of interoperability and reference for the definition of client APIs. However, a given transport may provide structures that provide for greater efficiency over the JSON message structures. In those cases, it is permissible for the implementation to deviate from the JSON message specification and leverage a native message specification, as long as the following rules are strictly observed:

- There is a transport-specific mapping layer that losslessly maps from or to the JSON message specification to or from the native message specification.
- These details are hidden from the client API.

8.0 Application Programming Interfaces (APIs)

Application Programming Interfaces (APIs) are used by U-Clients to interact with the U-Infrastructure. Figure 8-1 shows a view of the relationships between a UUDEX client and transport-specific APIs.

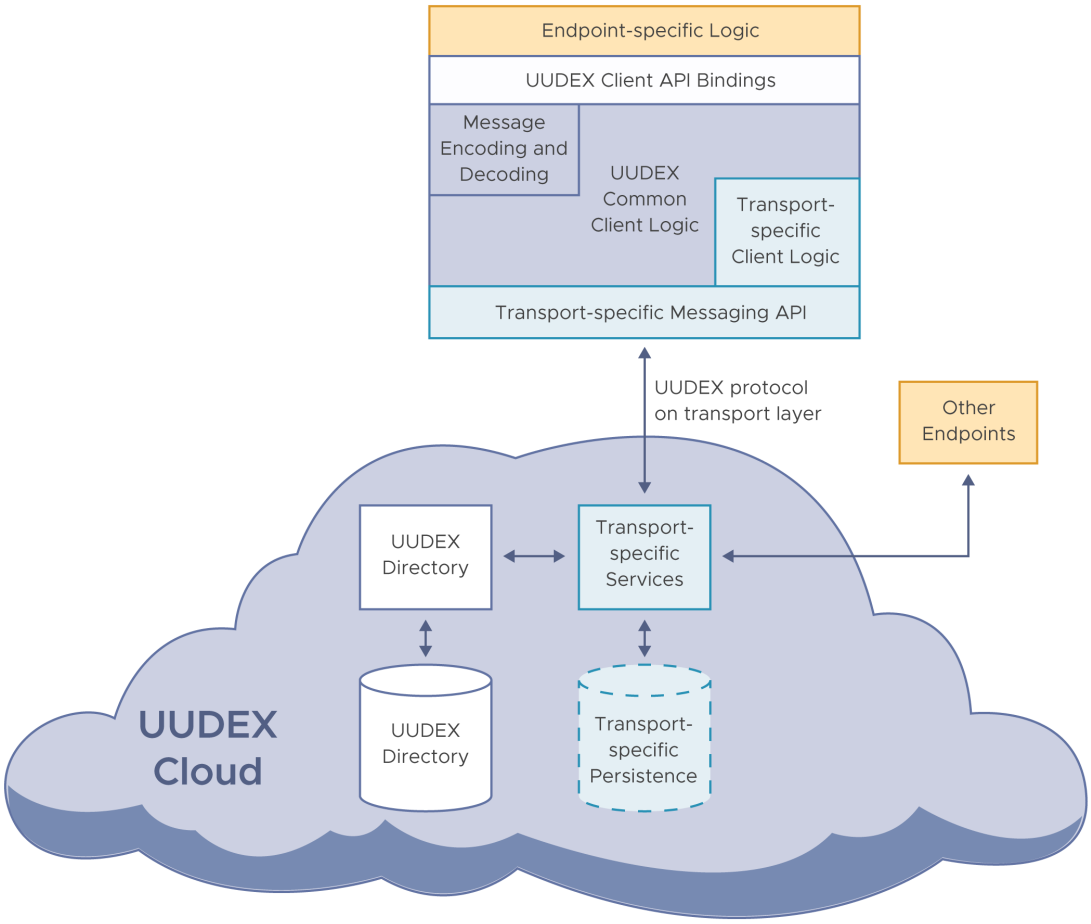

Figure 8-1: UUDEX API Overview

The UUDEX API must support multiple programming languages. For this reason, the interfaces are described at a message-level. This simplifies the process of mapping UUDEX from a given programming language to a specific message transport.

The UUDEX API provides several categories of basic interfaces:

- Utility classes for encoding and parsing UUDEX data structures, where UUDEX data structures are defined as JSON objects.
- Messaging primitives that are used to send and receive messages constructed from UUDEX data structures.
- Administrative interfaces.
- Discovery interfaces.

From these basic interfaces, it is also possible to define higher-level interfaces that could serve to minimize application code, but this is beyond the scope of this specification. It is expected that these API definitions would be refined and extended during the initial implementation using a selected transport.

8.1 Utility Classes

These are simply classes that allow an application programmer to create and parse JSON data structures used within messages. The intent of these classes is to minimize the number of lines of application code and avoid creation of data structures that are incorrect.

It is possible to interface to UUDEX without these classes by directly generating or parsing the JSON message structures.

The utility classes support creation, updating, and interrogation of the following JSON objects:

- Message, including:
  - metadata
  - payload U-Data Element
  - payload metadata (if needed)
- Response (which is a combination of a Message and a Result object)
- Result (which indicates success or failure and specific information that describes an error or exception condition)
- ACL
- Connection (which identifies information returned as a result of a connection)
- Credentials (which includes information needed for authentication).

Additional utilities include:

- Generation of a U-Data Element hash for use in payload metadata, where appropriate canonical rules are applied
- Verification of a U-Data Element with corresponding hash, where appropriate canonical rules are applied
- Formatting a U-Data Element using canonical rules
- Generation of a compressed and encoded U-Data Element from a file or memory object
- Generation of a file or memory object from a compressed and encoded U-Data Element
- Generation of a current timestamp.

8.2 Basic Messaging Interfaces

A set of primitives that allow for the conveyance of U-Data Elements using the UUDEX Framework (U-Framework) are listed below:

- Connect to U-Infrastructure, with authentication
- Subscribe to U-subject

- Consume next message
- Publish message to U-subject
- Issue a synchronous request
- Issue an asynchronous request

It is important to note that these interfaces are used to implement “high-level” interfaces, as might be used for administration and discovery purposes.

8.2.1 Connect

A connection request to UUDEX will have the following parameters:

- Connection string, which is a list of U-server host addresses
- U-Participant ID
- U-Endpoint ID
- Credentials (i.e., a digital certificate)

This returns a JSON Connection object.

It is important to note that while a connection is made to one U-Server, U-Server instances run in an active-active mode where reconnection to the next available U-Instance is automated.

8.2.2 Subscribe

A U-Subscription request may be made by a connected U-Client. It will include the following parameters:

- U-Subject name
- Optional: Quality of Service (QoS) (may depend on transport used)
- Optional: handler function (if not specified all messages published to this subject will be consumed using the ConsumeNext interface)
- Optional: filter parameters (used to filter messages consumed based upon element in message metadata, may include time-based criteria)

This returns a JSON Result object. The results should indicate an error if the U-Subscription is not authorized or any other error or exception condition was encountered.

8.2.3 ConsumeNext

This is a request to consume the next message, from all of the U-Subjects to which the U-Client has subscribed, as well as any asynchronous responses that were addressed to the response U-Subjects created by the U-Endpoint. Messages will be consumed by order of receipt and priority. The parameters include:

- Optional: Timeout in milliseconds (call is non-blocking)

This returns a JSON message object. If no message received within timeout period, the message object should be null.

Internally, the underlying software should apply filters that are specified on the U-Subscribe request.

8.2.4 Publish

This interface publishes a message to a subject. The following parameters are provided:

- U-Subject name
- JSON message object
- Optional parameters

This will return a JSON Result object. The Result object will indicate:

- Success or fail indication
- MessageID assigned (i.e., a UUID)

8.2.5 SyncRequest

This interface publishes a request message to the U-Infrastructure, where the response will be returned in a synchronous manner (i.e., the caller will block until a response is received or a timeout occurs). The request will provide the following parameters:

- JSON message object
- Optional: timeout in milliseconds

This will return when a response is received or timeout expires, noting that response messages are addressed by the U-Infrastructure to a specific response U-Subjects defined for each U-Endpoint. This should return a JSON response object that included:

- Message object or null
- Result object, that indicates success or failure

8.2.6 AsynchRequest

This interface publishes a request message to the U-Infrastructure, where the response will be returned asynchronously (i.e., the caller will not block) on a response subject that has been allocated for the U-Endpoint. It will be the responsibility of the application to consumer the response using the ConsumeNext interface and use the message correlationID to correlate the response with the messageID of the request.

The request will provide the following parameters:

- JSON Message object
- Optional: handler function (if not specified, the response messages will be consumer by the ConsumeNext interface)
- Optional: expiration time (time after which any response message should be discarded)

This will return immediately with a Result object that indicates success or failure.

8.2.7 Disconnect

This interface allows a U-Client to disconnect from UUDEX, where the time and nature of the U-Endpoint disconnection can be recorded by the U-Infrastructure. The Disconnect interface will have the following parameters:

- Optional: reason string

8.2.8 Flush

This interface will cause all queued messages for the U-Endpoint to be flushed. There are no parameters.

8.2.9 Unsubscribe

This interface will unsubscribe a U-Endpoint from a specified subject. The parameters are:

- U-Subject name

8.3 Administrative Interfaces

The following are a set of administrative interfaces that should be used by participants:

- Define U-Endpoints and associated permissions
- Define U-Subjects (to which U-Data Elements will be published by U-Participant U-Endpoints)
- Enable or disable U-Participant access to a U-Subject
- Define new U-Data Element Types, or versions thereof

The requests will be defined as “wrappers” over the SyncRequest interface, where requests are addressed to U-Subjects defined for the U-Infrastructure. The message verb will be “CREATE”, “CHANGE” or “DELETE,” and the message noun will be one of U-Endpoints, U-Participants, ACLs, U-Data Element Types, or U-Subjects.

These interfaces require a U-Endpoint to have made a successful connection to the U-Infrastructure. The information updated is limited by the current set of ACLs and rules defined within the U-Infrastructure.

8.4 Discovery Interfaces

Following are a set of discovery interfaces that can be used by U-Endpoints for a given participant. These expose “visible” information to a U-Endpoint, where ACLs allow access to a given U-Participant:

- View defined U-Participants
- View defined U-Endpoints
- View defined U-Data Element Types
- View U-Subjects for a given U-Data Element Type
- View U-Subjects available for subscription from a publishing participant

The above are accessed using the following “GET” requests:

- Get U-Endpoints
- Get U-Participants
- Get U-Data Element Types
- Get U-Subjects

The requests will be implemented as wrappers that use the SyncRequest interface, where messages are addressed to U-Subjects defined for the U-Infrastructure. The message verb will be “GET,” and the message noun will be one of U-Endpoints, U-Participants, U-Data Element Types, or U-Subjects.

These interfaces require a U-Endpoint to have made a successful connection to the U-Infrastructure. The information retrieved is limited by the current set of ACLs and rules defined within the U-Infrastructure.

9.0 Security

The purpose of this section is to describe security from the perspective of the UUDEX information exchanges. It is important to note that once a participant has received information through the U-Infrastructure, they need to abide by use agreements as to how that information is used and disseminated elsewhere. UUDEX is not intended to be used for anonymous, public access of information.

9.1 Authentication

9.1.1 To The UUDEX Instance

UUDEX employs U-Identity Authorities to perform identity proofing and provide authentication and authorization services to a U-Instance. U-Identity Authorities function as certificate authorities for the Private Key Infrastructure (PKI) services used in the U-instance. When a U-Participant is initially introduced to the U-instance, they will provide contextual details, such as organization name and function, physical location, and contact information (defined in Section 5.1.1) to the U-Identity Authority. Once the information has been verified and the U-Participant has been approved to operate on this U-Instance, the U-Identity Authority that the U-Participant contacted will generate a U-Identity Object, a digital certificate that contains the entity information, which is defined in Section 5.1, and is tied to a certificate authority hierarchy that provides verification to other participants later. Once a U-Participant is registered, it will be allowed to register U-Endpoints that will interact with the U-Instance.

The U-Identity Object that is maintained contains all of the information about a U-Participant that is laid out in Section 5.1.1, including the U-Participant ID, list of authorized U-Endpoints, ID of the U-Identity Authority it registered with, and a digital certificate with meta-information. The U-Identity Authority keeps track of whether or not the U-Participant's access has expired or been revoked. The digital certificate that is generated by the U-Identity Authority is signed and given to the U-Participant, which allows other U-Participants they want to communicate with to ascertain their identity. An example of a public key format that could be used is X.509 certificates, which is the standard used by Transport Layer Security (TLS).

By default, all U-Endpoints for every U-Participant have their own unique digital certificate that is used for communication in a U-Instance. In the event that a U-Participant can't manage that number of digital certificates due to the amount or workload, there are other options available for managing the U-Identity Objects, described in the *UUDEX Security and Administration* document.

9.1.2 Between UUDEX Endpoints

Every U-Participant in the U-instance decides who it trusts, and what specific information they trust them with. This trust information is expressed as access permissions either granted or denied in the ACL that is associated with each U-Subject managed by the U-Participant.

All relationships that are started are required to authenticate to the U-Instance using appropriate mechanisms, so each side of the connection can determine if that trust is there, and specifically whether both are members of the same U-Instance. Upon initial communication, each side of the U-Connection presents their digital certificate given to them by the appropriate U-Identity Authority, so that the other party can cross check the information and decide whether a

U-Connection should be established between them. If both sides accept the connection, then an encrypted communications channel will begin; otherwise, the connection is dropped.

All communications within UUDEX happens between U-Endpoints, i.e., between U-Participant Endpoints and U-Infrastructure Endpoints (i.e., U-Servers). U-Infrastructure Endpoints perform all the processing necessary to ensure that U-Participant Endpoints are authorized to either produce or consume U-Data Elements to U-Subjects. This mechanism also hides any communication patterns between U-Participant Endpoints that could be observable if individual U-Participant Endpoints communicated with each other.

Communication between U-Endpoints follows the standard mutual TLS handshake. An initial "Client Hello" message is sent from *Participant A* (i.e., a U-Participant Endpoint) to *Participant B (i.e., a* U-Infrastructure Endpoint). *Participant B* responds with their digital certificate that can be verified by *Participant A*, and a request that *Participant A* sends their digital certificate in response. Both sides of the relationship check with the U-Identity Authorities that the digital certificates provided are valid and are associated with the correct U-instance, and if they are, then they can begin communication.

9.1.3 Revocation and Expiration

U-Identity Objects contain a field marking a U-Endpoint's digital certificate as valid, expired, or revoked. In the event a U-Endpoint is no longer needed or if it is deemed a threat, the associated digital certificate will be marked "revoked" in all U-Identity Authorities, and the U-Endpoint will no longer be able to authenticate with any other U-Endpoint in the U-Instance. Any revocation that happens is explicit, as UUDEX supports real-time, mission-critical applications that must not be erroneously interrupted. If a U-Participant's access to a U-instance is revoked, all U-Endpoints associated with the U-Participant would have their digital certificates revoked.

Each U-Endpoint shall periodically check the revocation status of all U-Identity Objects in use, and if the U-Identity Objects is marked as revoked, the U-Connection shall be terminated.

Note that if the U-Identity Object is marked as expired while a U-Connection is established, consistent with IEC 62351-3:2014 Clause 5.6.4.5, a warning message may be issued, but the communication should continue. However, if the U-Identity Object is marked as expired and U-Connection is not yet established, it should not be established, and an alert raised to allow the U-Administrator or U-Participant Administrator to correct the issue.

9.2 Authorization

9.2.1 Trust Relationships

Trust relationships exist where parties must rely on others to conform to certain behaviors or properly perform certain functions without being able to explicitly enforce those behaviors. The behaviors that are expected of a U-Server conforming to these trust relationships are defined below:

- To enforce security policies on U-Data Elements with regard to requests to post, read, modify, and delete U-Data Elements. This includes being trusted to enforce their own access rights to the U-Data Elements (i.e., the U-Server is trusted not to read U-Data Elements to which it does not have read access).

- To accurately process queries for U-Data Elements. This means they must correctly identify matching U-Data Elements to which a requesting party has access and accurately respond to the requestor based on this information.
- To accurately maintain and serve U-Subscriptions established by U-Clients.
- Not to add, modify, or delete U-Data Elements except at the direct instruction of an authorized U-Client. This is the case even if the U-Server is granted access rights to perform these activities.
- To execute commands from authorized U-Clients (e.g., if a U-Server is instructed to delete a U-Data Element by an authorized U-Client, the server is trusted to perform that action).
- To accurately report its status (e.g., whether its services are currently degraded).
- To conform to behaviors dictated by prioritization policies.

U-Clients also have expected behaviors:

- To adequately protect U-Data Elements they retrieve from U-Servers. In particular, they are trusted not to disclose the U-Data Element (intentionally or unintentionally) to parties that are not authorized to view the U-Data Element.
- Not to send false information in U-Data Elements.
- Not to create undue communications load by sending excessively large amounts of U-Data Elements to U-Servers.
- Not to create undue processing loads on U-Servers by making excessive UUDEX Query or UUDEX Subscribe requests.

9.2.2 Access Control

Each U-Endpoint must be authenticated with the U-Infrastructure. Presenting valid digital certificates verifies the identity of each party, but it is still up to each U-Participant to decide if the opposite party has authorization to view or publish certain types of data. UUDEX supports this in the form of ACLs. An ACL acts as a whitelist, allowing only certain U-Endpoints to perform certain functions (view, publish, subscribe, etc.) on certain data, and denying all other interactions. The structure of an ACL is outlined in 5.1.5 and further discussed in the *UUDEX Security and Administration* document, and follows the basic principles that each U-Participant must be a registered and authenticated part of the U-Instance, and must have explicit access to the U-Subject it is either subscribing to, or publishing information to.

9.3 Confidentiality

9.3.1 Data In Transit

All UUDEX Exchanges (U-Exchanges) are encrypted using algorithms deemed sufficient for protecting Sensitive but Unclassified (SBU) information by using an encrypted tunnel supported by a transport protocol like TLS v1.3. TLS provides encrypted end-to-end communication over networks using approved algorithms, as well as additional support for data integrity through hash-based message authentication code (HMAC). Once both U-Participants have had their identities verified, and the U-Server has deemed the U-Client has authorization to communicate with them, the U-Client creates a session key and encrypts it with the server's certificate and a

supported TLS algorithm. The session key is used to facilitate an encrypted tunnel for the rest of their communication.

9.3.2 Data At Rest

All information in U-Subjects are maintained and protected by U-Servers. One of the trust relationships that the U-Servers perform is that they enforce security policies including how the data is handled while it is at rest. Therefore, U-Servers will encrypt data that are not in use.

U-Servers use approved symmetric key encryptions, such as the 256-bit Advanced Encryption Standard (AES-256), provided by the U-Identity Authority for data that is not currently in use. In most cases, the underlying messaging system, data storage software, or supporting operating system may be able to provide the required data protection.

9.3.3 Digital Certificate and Key Management

All keys and digital certificates are maintained by U-Identity Authorities. Digital certificates are used for communications between U-Participants, whereas keys are used to encrypt and decrypt data housed by U-Servers in U-Subjects. All digital certificates and keys are created on request when needed. Each U-Participant receives a different digital certificate, and every U-Server is encrypted with a different key, but keys can be shared between distributed U-Server instances that are maintaining the same logical U-Server. All digital certificates and keys are maintained and used until the end of a usage period, or a set expiration date, and stored but not used for a certain amount of time afterwards just in case an old U-Server needs to be decrypted then encrypted with the new key. In general, the more sensitive the data and the more data that is processed with a specific digital certificate or key, the more often a key or digital certificate rollover should be performed, and new ones generated.

9.3.4 Resource Considerations

Increasingly, security comes with the cost of increasing the computational resources necessary to communicate on the network. Although not a primary design consideration, certain embedded devices do not have the resources to properly perform their main function and simultaneously securely communicate data to U-Servers. TLS provides support for some features that decrease the processing power needed, like storing sessions to reduce the amount of overhead in establishing relationships, using encryption algorithms that are weaker but easier to implement, or providing authentication-only services that do not encrypt the message.

In the event that TLS encryption cannot be supported at all, clients and servers can use payload encryption at the application level. This will not encrypt the transport-level encapsulation of the message or the meta-data associated with the message, but it will encrypt the application-specific data that is being sent. Key maintenance is required for payload encryption. In an environment where one has a few trusted subscribers, asymmetric encryption can be used but requires public and private keys. The private key is shared between trusted subscribers and U-Servers, and a public key that can be accessed by anyone. For publishers to send information to subscribers or U-Servers, they encrypt their data with the public key, and it can only be decrypted and then stored or read with a private key. This is the ideal setup where there are many sensors that need to send data but do not need to receive any data, but do not have the computational power to run TLS. In an environment where every U-Endpoint is trusted, symmetric encryption which is easier to implement but not as secure could be used. In this case, all messages could be encrypted and decrypted using the same key or password. Using

payload encryption creates a more secure environment when TLS encryption cannot be implemented, but it is not as secure because replay and man-in-the-middle attacks can still be used against the network.

If a sensor or other device cannot perform any amount of encryption due to resource restrictions, the device can send any data that needs to be published to the U-Instance to a broker that can then encrypt the data for further communication. Performing this over an IP-based network is not ideal, as it does not inherently protect that communications link against any form of attack to compromise the data.

9.4 Integrity

Data integrity of U-Exchanges depends on verification of messages while transmitted and the proper storage and synchronization of those messages in the U-Subject. After authentication, nodes will begin communicating using the message patterns defined in Section 6.0, but authentication does not necessarily guarantee the integrity of messages which can still be accidently altered during transmission.

The encryption requirements outlined in Section 9.3 provide protection against undetected data tampering at the transport level when coupled with proper authentication. Adding a hash or HMAC on top of a properly deployed authentication and authorization mechanism does not add much additional security, but it does provide data integrity. TLS implements an HMAC system that ensures both security and integrity of the data sent, as well as who it was sent by, but if a U-Instance cannot support a full TLS installation (due to reasons outlined in Section 9.3.3), UUDEX implements a message-level integrity check in the form of a message hash, which allows any node to verify any message they receive but does not provide the additional security measures supplied by an HMAC. The message hash algorithm can be chosen based on computational or security needs, for example, the MD5 hash is not computationally expensive and can be generated quickly even with large data although it is no longer considered secure, while the SHA-256 hash requires more computational resources but is considered more secure. Once a message is sent and appended with the hash, the receiver can verify that there was no data corruption.

U-Servers also store their data with an accompanying HMAC. Storing messages with an HMAC has the additional benefit of allowing U-Servers to verify the integrity of their stored messages before sending them, which can help in the detection of hardware faults. All data stored at U-Servers also are encrypted while at rest, including metadata and configuration information, such as U-Subscriptions lists, to prevent undetected data tampering at the node and propagation of manipulated data.

9.5 Resilience

9.5.1 Denial of Service and Distributed Denial of Service Protections

Denial of Service (DoS) occurs when the operational capacity of an essential part of a U-Instance is breached and that component is no longer able to perform its function to the point where the U-Instance becomes partially or completely inoperable. A Distributed Denial of Service (DDoS) can happen when multiple network clients (either legitimate U-Clients or malicious network nodes) attempt to request service from a U-Server. A DoS or DDoS can be accidental (e.g., too many updates hitting a U-Server at once, resulting with the U-Server being unable to keep up), or malicious (e.g., an attacker flooding the network with high priority packets

to the point of failure). Both DoS and DDoS can be intentional when initiated by a malicious attacker or could be unintentional when initiated by a configuration error.

The best protection against DoS or DDoS is to understand the capacity of each component of the U-Instance and throttle U-Client connections appropriately. The network bandwidth and the computational power of U-Servers is measurable, therefore picking a rate limit for U-Clients should be possible. Because UUDEX has a QoS priority system, the rate limiting will need to apply to messages of all prioritization levels, otherwise an attacker could just flood the network with high priority messages. Rate limits are enforced at the U-Servers and are based on U-Client IDs; if a U-Server finds that it is receiving messages from a specific U-Client too quickly, it will stop reading from that connection and allow “backpressure mechanisms” built into the communications protocol like the Transmission Control Protocol (TCP) to slow the client’s send speed down. In the event that this rate limiting does not work, a U-Server can begin dropping the packets completely, but this is not optimal in a mission-critical environment.

Another highly effective way to prevent DoS is to use clustering by having multiple U-Server instances in strategic locations on the network handle U-Client traffic. It is also important to properly partition U-Subjects in a U-Instance. In such a distributed U-Server environment, synchronization or replication of U-Subjects between all the U-Server instances is required.

If the network has one global U-Subject that all messages are sent through rather than a stratified approach for specific types of messages, then it is more likely that a U-Server could get overloaded and slow operations. For this reason, among others, consideration of creating U-Subjects that are specific to U-Data Element Types and U-Participants Is suggested.

9.5.2 Redundancy and Backups

U-Instances are not meant to operate without multiple U-Identity Authorities. Because identity verification is an essential piece to the communications on the network, a single U-Identity Authority could be a bottleneck, and if the sole U-Identity Authority was compromised, then an attacker could have full authority to all the certificates on the network and be able to communicate with any U-Participant freely. If the U-Identity Authority was compromised, an entirely new U-Instance would have to be deployed.

In an ideal situation, there would also be redundant instances of distributed U-Servers across multiple locations in the U-Infrastructure that would decrease the time it takes to access information in that U-Server, as well as provide resilience against the U-Server instance containing that information failing or becoming compromised. To maintain the correct information across multiple instances of a U-Server, UUDEX synchronizes or replicates them when new information has been published. Each U-Server keeps track of what U-Subjects other U-Servers are maintaining, and authorize those servers through ACLs to publish to their U-Subjects. When a new message comes from a U-Participant, the U-Server sends that message to the other U-Servers maintaining that U-Subject in order to update them on the latest messages. If a message is received from another U-Server, then the receiver verifies that the message has not already been received by looking at its message ID, and then publishes it to the correct U-Server instance. Depending on the network infrastructure, this feature may need to be restricted to lower the number of messages being sent on the network.

Replication of and resiliency for U-Subjects may also be accomplished using features of the underlying messaging transport software which can minimize the amount of UUDEX-specific code that needs to be implemented in a U-Instance.

UUDEX also supports systematic hot and cold backups, which increase the ability to recover from a disaster. Hot backups in a U-Instance work similarly to synchronizing U-Servers; a hot backup is subscribed to a U-Subject and gets any new messages that come in, either on an interval or upon storing. Cold backups are faster and safer but require the U-Server to be taken off-line during the backup process. Backups of U-Servers are not meant to be stored on the same U-Server instance that houses them to allow recovery from storage hardware failures.

10.0 Bridging

UUDEX may be deployed using multiple logical (or physical) U-Infrastructures. A U-Infrastructure refers to a network infrastructure and connectivity mechanisms that a set of U-Endpoints use to communicate. There are a variety of reasons why a single U-Instance could be deployed using multiple U-Infrastructures:

- A different messaging transport is used in each U-Infrastructure. UUDEX is designed to avoid "lock-in" to a specific messaging product, being largely transport neutral. One reason for this is to help "future proof" UUDEX, allowing future messaging products or protocols to be leveraged. For this reason, there may be implementations of UUDEX that use different messaging technologies.
- A U-Instance might be divided into multiple U-Infrastructures deployed on a regional basis, reflecting different sets of participants (for example by an Independent System Operator [ISO] or Regional Transmission Operator [RTO]). This could be done to support distributed management of the U-Instance or optimize communications within each regional group.
- A U-Instance might be divided into multiple U-Infrastructures on a functional basis, where the primary needs of participants differ (e.g., U-Data Element Types to be exchanged). Again, this could be done to support distributed maintenance. It could also ensure different functional groups had dedicated resources for communication, independent of other uses of that U-Instance.
- A U-Instance might be deployed in a way that major U-Participants manage their own U-Infrastructures. This might be desired in cases where physical control over certain classes of data is seen as important. Communications across all the U-Infrastructures of the U-Instance would remain possible but placing separate U-Infrastructures with each U-Participant would give greater physical control of data at rest.

Figure 10-1 shows two U-Infrastructures within a single U-Instance, where a "bridge" is used for the conveyance of information between each U-Infrastructure.

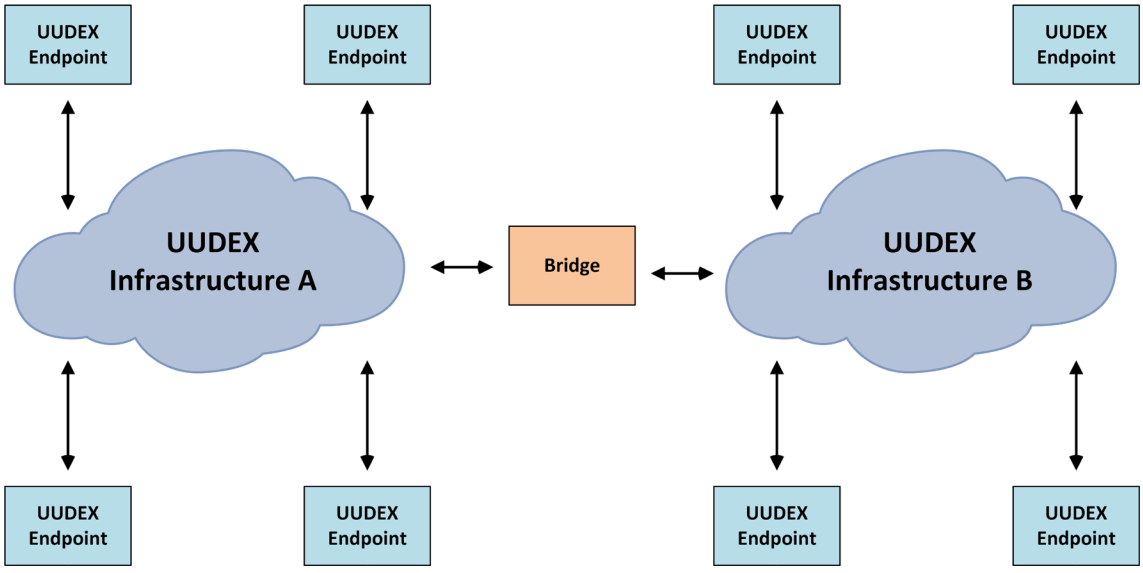

Figure 10-1: UUDEX Bridging

UUDEX will provide for the ability to securely bridge (i.e., exchange U-Data Elements and messages) between different U-Infrastructures within a single U-Instance. Implementation of a bridge will permit messages conveyed between two (or more) U-Infrastructures that use different transport technologies to be exchanged in a manner that is consistent with restrictions imposed by ACLs. Note that both U-Infrastructures would need to belong to the same U-Instance since separate U-Instances form independent trust environments. Credentials from one U-Instance are not valid in a different U-Instance, which would prevent direct communications between the two. As such, in this case a bridge is used in UUDEX to connect separate technology environments, not separate trust environments.

Alternatively, each U-Infrastructure could represent a separate U-Instance, where the bridge function is used to exchange U-Data Elements between the two U-Instances. In this case, the bridge node would be a client member of both U-instances, subscribing to U-Subjects in one U-Instance, publishing to U-Subjects the other, and vice-versa. The bridge node would need to have separate credentials from each U-Instance since credentials cannot be shared between different U-Instances, and U-Participants and U-Administrators from both U-Instances would need to understand the rules used in the bridge node to transfer information between the two U-Instances. Specific permission or roles within each U-Instance may be required to manage the access afforded to the bridge node.

11.0 UUDEX Performance Metrics

A U-Implementation is largely dependent upon the basic capabilities of the messaging transport upon which it is implemented. Central to prioritization of messages is the definition of U-Subjects, which identifies a relative priority. It is important to note that a publisher and a consumer may have different views on the relative importance of the information on a given subject.

A publish and subscribe system can be evaluated on its performance in several aspects: scalability, availability, latency, and QoS, details of these aspects follow.

11.1 Scalability

A scalable service should be able to handle increases in load without noticeable degradation of latency or availability. “Load” can refer to various dimensions of usage. For example:

- Number of U-Subjects
- Number of publishers
- Number of U-Subscriptions
- Number of subscribers
- Number of messages
- Size of messages (minimum size, maximum size, average size)
- Rate of messages (throughput) published or consumed
- Size of backlog on any given U-Subscription
- Prioritization of messages

11.2 Availability

In a distributed system, the types and severity of availability problems can vary greatly. System availability is measured on how well it deals with different types of issues, gracefully failing over in a way that is unnoticeable to end users. Failures can occur in hardware, in software, and from the load encountered. Hardware failures generally result in broken components such hard disks or network adapters, and generally require the failed component be removed from service for repair. Software failures generally are the result of underlying software bugs that are triggered under specific traffic or data patterns, incompatible software upgrades, or configuration errors. Failure due to load could happen when a sudden increase in traffic in the service (or in other software components running on the same hardware or in software dependencies) results in resource scarcity. Availability can also degrade due to human error, where one makes mistakes in building or deploying software or configurations.

UUDEX is a mission-critical system that needs to operate 24/7. To maintain such a system, at least three environments will need to be deployed: development, staging, and production (which may itself have multiple environments for resiliency and recovery). Development and staging do not contain any production data. Development provides an environment in which developers can create new features for the production system. Staging provides an environment where maintainers will execute continuously-running tests and monitoring that help to find any issues

with releases. In some cases, separate environments may be needed to support testing, quality assurance and final staging. Once a release has been staged, tested, and approved for release, the release then is moved to the production environment. This development lifecycle will aid in availability and help to detect, deter, and fix any production issues that may arise.

The procedure for developing, testing, and implementing UUDEX is designed to minimize potential impacts. The procedure includes the following steps:

- Ensure all unit tests and integration tests pass
- Build a new version of all the servers
- Deploy the new servers to the staging environments and integrate small amount of production traffic
- Run the servers on the staging environment for several days
- If no problems are detected in staging, implement the new release to production.

UUDEX is designed to be resilient to failures; thus, rollouts of new UUDEX versions are seamless to end users and should have no impact on performance.

Like any real-time system, the rollout procedure should also include a back-out process in the event that bugs are found in spite of all the testing performed in the development and staging environments.

In particular, the message exchange formats are expected to allow continuous evolution for the message content and format with minimal impact on existing exchanges. See Section 13.0 for additional information.

11.3 Latency and Jitter

Latency is a time-based measure of the performance of a system. A service generally wants to minimize latency where possible. For UUDEX, two important latency metrics are:

- The amount of time it takes to acknowledge a published message.
- The amount of time it takes to deliver a published message to a subscriber.

Jitter is a measure of how regular a particular periodic message is transmitted or received on the network and may be observed when periodic data does not arrive at scheduled intervals. Application programs generally require periodic refresh of data, and therefore jitter should be minimized.

To aid with latency and jitter, the ability to control the flow of data is essential. There may be a trade-off associated with flow control where latency is slightly increased in order to manage jitter. The flow control feature allows administrators to maximize throughput while preventing overload in UUDEX. Flow control is a form of traffic shaping whereby sudden unexpected spikes in load can be smoothed out over time for greater service stability. Flow control operates system-wide or on a per-U-Subject or per- U-Subscriber basis to limit the number of messages or the number of bytes that are transferred or are outstanding.

11.4 Quality of Service (QoS)

UUDEX provides QoS and prioritization for network interactions and request processing, depending on the client's network reliability and needs. Higher prioritization can be used to respond to a client's request to increase the likelihood that the request is processed through the queue first (by default, a request with a high priority skips through the queue until it hits another request with equal or higher priority). Priority and QoS are associated with the U-Data Elements and are enforced by policy set on the U-Server. The QoS service level discussion is based on the Message Queue Telemetry Transport (MQTT) v5.0 standard[1].

The QoS defines the level of delivery assurance that the client needs, either due to network reliability or importance of the information. The QoS concepts described are for application-to-application message transmissions and are above any reliable network reliability mechanisms. This allows messages to be passed reliably between a publisher and a subscriber using otherwise unreliable network services (such as User Datagram Protocol [UDP]).

QoS is defined at three levels: "At most once", "At least once", and "Exactly once."

- "At most once" is the lowest priority level, where a request is processed, and the result is sent out with no guarantee from the server or the client that the message was delivered. This is similar to how UDP sends data without acknowledgment from the client that the data has made it. This is the quickest and most efficient message transmission but has the lowest level of assurance that the message has been received.
- "At least once" tells the sender to repeatedly send the data until it gets an acknowledgment from the recipient that the message was delivered. The message may be repeated (with an indication that it is a duplicate message) if the sender does not receive an acknowledgement of some kind within a reasonable timeframe (where reasonable is dependent on the end-use application). This is similar to how TCP sends data and provides an acknowledgement and retry mechanism in the event that the data is not successfully transmitted. In most cases, this is an efficient method of transmitting messages, requiring a simple acknowledgement message. However, if messages are lost, the retransmit interval and duplication of messages may lead to messages not arriving in a timely manner, and the increased overhead of transmitting and receiving multiple duplicate messages.
- "Exactly once" is the highest level of network QoS where the sender waits until it gets an acknowledgement from the recipient. The sender then sends a second packet to the recipient indicating that it acknowledges the successful transmission and marks the transmission successful. The recipient then responds acknowledging the successful transmission. If any of the messages are not acknowledged, they are re-sent and marked as duplicate messages. This method has the highest QoS, but at the cost of the resources required to transmit (at least) four messages between the sender and recipient for each data packet exchanged, as well as increased processing and storage requirements for managing the additional message exchanges.

U-Servers attempt to avoid as much data loss as possible, even in the event of a U-Server crash or maintenance window. Messages and requests that are received by a U-Server are put into a queue awaiting processing. That queue is periodically snapshotted and saved so that in

[1] See https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

the event of a crash, the U-Server could pick up where it left off. Because the nature of the data being processed is sensitive, the state file and contents of the queue are temporary.

11.5 Monitoring and Metrics

The key to keeping UUDEX running is to automatically detect and mitigate performance and connectivity issues before they become visible to end users. Accomplishing this requires extensive monitoring of the system at both the client and server locations. UUDEX will have a set of well-defined metrics that describe the behavior of the system. The health of a U-Instance will be determined using these metrics, thereby providing supporting information for maintenance activities. Examples of these metrics include the following -

- Number of undelivered messages
- Acknowledge message count
- Number of retransmission requests
- Rate of errors generated by publish requests
- Number of unsubscribed publish requests
- Link throughput
- Link latency
- Link jitter
- Active connection count
- Number of connection drops
- Number of messages delivered not meeting desired QoS
- Rejected connection count.

These metrics can be measured in a variety of ways. Monitoring tests to measure performance would perform specific actions just as an end user would and also would measure how long the operations take. For instance, a test could be performed that creates a new U-Subscription, publishes a message, and measures how long it takes to both create the U-Subscription and receive the message. If the duration of such a test is over an “acceptable” limit, the U-Administrators could be notified.

The U-Framework will provide necessary APIs to measure and report system performance based on these metrics.

12.0 Maintenance, Diagnostics, and Testing

UUDEX will provide an API to allow access to maintenance and diagnostic data that could be made available on dashboard displays or other analytic or display tools to summarize system health. Using a dashboard to display this data will provide quick access to statistics and graphs of the UUDEX service that will enable a user to identify ongoing or emergent problems, anomalies, and general trending to mitigate future problems. The data could also be filtered by keyword or publisher to provide the user with a more focused view of UUDEX health.

In addition to regular monitoring of UUDEX health, there may be certain diagnostic tests that need to be run periodically to ensure proper functioning. These include tests that measure the performance of infrequently used functions such as creating a U-Subscription service for a new client or refreshing the list of subscribed U-Subjects for an existing client. Testing for potential issues with these services must be scheduled periodically to ensure smooth operation at the time of need. The U-Framework will contain tools meant for periodically testing these specific use cases.

Maintenance and tests would also have to be run for cases where an interruption or decline in quality of UUDEX performance has been observed by users. The U-Framework will also come equipped with toolsets required to identify and debug these issues.

UUDEX supports the transmission of test messages to assist with problem diagnosis and performance monitoring. These test messages do not contain any real data but are otherwise treated like messages containing real data. Test messages by default are assigned the lowest priority and QoS for transmission and processing to minimize the impact to real data transmissions but may be assigned higher priority or QoS in order to test prioritization processing or QoS capabilities.

U-instances may also perform automated periodic maintenance, for example, to reorganize disk storage, or to clean up and optimize U-Subscriptions lists. These actions are largely implementation dependent, but their operation may be initiated or monitored using a standard API.

13.0 Extensions

UUDEX is designed to implicitly handle extensions, where:

- Additional information is added to existing U-Data Element Types.
- New U-Data Element Types may be defined as needed.

The U-Infrastructure defines a set of known U-Data Element Types. Some of these may be designated as “standard,” and others can be defined to meet the needs of a given set of U-Participants. The extensions to an existing U-Data Element Type can be handled in one of two ways:

- Defining a new version of the U-Data Element Type, as should be done when the extension is a potentially “breaking” change (for example, when removing information fields or changing the meaning of existing information fields)
- Simply adding information to an existing U-Data Element, typically leveraging JSON, where the extension is otherwise a “non-breaking” change

U-Clients should be implemented in a manner that allows for these types of extensions. When new U-Data Element Types are defined, participants that desire to publish U-Data Elements of that type may need to define new U-Subjects for publication.

14.0 Responsibilities

14.1 "Bus"

Following are the responsibilities of the UUDEX API, protocol, and infrastructure:

- Provide a publish and subscribe messaging transport for the conveyance of messages
- Manage the authorization of connections
- Provide the ability to publish information to defined U-Subjects
- Manage and respect ACLs for U-Subscriptions to U-Subjects
- Support the exchange of a core set of U-Data Element Types
- Support an extended set of U-Data Element Types
- Provide a secure transport layer

14.2 Users of the "Bus"

Following are responsibilities for participants and their associated U-Endpoints:

- Register U-Participants and U-Endpoints
- Deploy U-Endpoints that use the UUDEX API to communicate with the U-Infrastructure using the UUDEX protocol
- Create U-Subjects as needed for publication of information
- Define U-Data Sets for those subjects that involve publication of time series data
- Set ACLs for other participants to view or subscribe to their published subjects
- Perform any needed integration between U-Endpoints and their "back end" applications
- Deploy multiple U-Endpoints as may be needed for redundancy
- Sign or encrypt payload U-Data Elements as needed
- Avoid forwarding information to any other U-Endpoint or for any other use except as explicitly permitted by participant information sharing agreements

15.0 References

The following are key references for this specification:

- UUDEX Functional Specification
- IETF RFC 4122 (UUID) (https://tools.ietf.org/html/rfc4122)
- IETF RFC 4180 (csv) (https://tools.ietf.org/html/rfc4180)
- IETC RFC 5246 (TLS 1.2) (https://tools.ietf.org/html/rfc5246)
- IETF RFC 6713 (gzip) (https://tools.ietf.org/html/rfc6713)
- IETF RFC 7159 (JSON) (https://tools.ietf.org/html/rfc7159)
- IETF RFC 8448 (TLS 1.3) (https://tools.ietf.org/html/rfc8448)
- IEC 60870-6-802 (TASE.2)
- ISO/IEC 646 (7-bit ASCII)
- ISO 8859
- ISO/IEC 10646: The Universal Character Set (UCS)
- UNICODE “The Unicode® Standard Version 13.0 – Core Specification” (UTF-8)
- IEC 61968-100 (Messaging)
- IEC 61970-301 (CIM Base)
- IEC 61970-452 (CIM Model Exchange)
- IEC 61970-501 (CIM RDF Schema)
- IEC 61970-503 (CIM XML Model Exchange Format)
- ISO 8601 (time representation)
- ITU-T X.509 (public key certificates)
- NIST FIPS-197 (AES) (https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf)
- OE-417 (DOE Electric Emergency Incident and Disturbance Report) (https://www.oe.netl.doe.gov/oe417/Form/Home.aspx#)

Appendix A – TASE.2 Mappings

The purpose of this appendix is to describe the high-level mapping of TASE.2 to UUDEX.

TASE.2 Domain – Replaced by an ID for the participant.

Transfer Set – Equivalent to a U-Data Set, defines a set of data points that have dynamic values. UUDEX uses U-Data Sets contained in U-Messages and published to and subscribed from U-Subjects to exchange data.

Quality code – UUDEX takes the quality codes defined by TASE.2 and extends them. UUDEX uses keyword codes rather than bit masks to allow for extensions beyond any specific fixed-length value.

Bilateral Table – UUDEX uses the concept of self-describing data descriptions and universally understood point ID names (e.g., IEC CIM master Resource Identifiers) to control how data is exchanged. Access to data points is controlled by specifying ACLs assigned to individual USubjects for publish and consume functions. It is the responsibility of each UPublishing Client to publish data values to USubjects that contain the correct access controls for USubscribing Clients to access the data.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99352
1-888-375-PNNL (7665)

www.pnnl.gov