The Energy Mashup Lab

Common Transactive Services 1.0

The Energy Mashup Lab Draft Specification

Draft of 28 October 2020

Editor:

William T. Cox (cto@theenergymashuplab.org wtcox@coxsoftwarearchitects.com)

Toby Considine (director@theenergymashuplab.org toby.considine@gmail.com)

Additional artifacts:

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

- UML models
- JSON schemas
- Simple Binary Encoding binding (FIX)
- XML schemas

Related work:

This specification is related to:

- OASIS Energy Interoperation v1.0 (OASIS Standard)
- OASIS WS-Calendar Platform-Independent Model v1.0
- OASIS WS-Calendar Streams v1.0

Abstract:

TBD

Status:

This document is produced by The Energy Mashup Lab

Notices

Table of Contents

Index to Figures

Index to Tables

1 Introduction

Common Transactive Services (CTS) allows actor interaction with any market. Technically, CTS is a streamlined and simplified profile of the OASIS Energy Interoperation (EI) specification, which describes an information and communication model to coordinate energy between any two parties, such as energy suppliers and customers, markets and service providers.

Transactive Resource Management (TRM) has been used for many non-energy resources, such as water delivery, network bandwidth, and even internet advertising. The initial research in TRM used a market to allocate heat within a commercial building [TRM]. In TRM, a resource is defined as a tradable commodity whose value depends on price, location, and time of delivery [EMIX][1]. TRM balances supply and demand over time using automated voluntary transactions between market participants.

TRM applied to energy is commonly referred to as Transactive Energy (TE). Neither EI nor CTS specifies which technologies participants will use; rather they define a technology-agnostic interface to enable accelerated market development of such technologies.

TRM is a means to allocate transactable energy resources including the delivery of commodities such as electrical energy, electrical power, natural gas, and thermal energy such as steam, hot water, or chilled water. Transactable energy resources also include the capability to deliver resources, such as transmission line capacity and flow-rate capacity.[2]

The Common Transactive Services are a lightweight profile of the OASIS Energy Interoperation specification. All CTS messages are simple, and make no assumptions about the systems behind the messages.

The target actors for CTS include but are not limited to

- Smart Buildings/Homes/Industrial Facility
- Building systems/devices
- Business Enterprises
- Vehicles
- Microgrids
- IoT (Internet of Things) devices

Transactive Energy has the potential to make our electrical system more efficient, by better matching supply and demand in real time. TE enable actors to use energy when it is less expensive and produce energy when it's more valuable, thus reducing reliance on distant suppliers while maximizing use of local power sources. TE relies on markets and consumer choice to make better decisions about power supply and use.

TE demonstrations and deployments to date have been unique systems—each uses its own message model and its own market dynamics. Many early implementations required the use of

---

[1] See http://docs.oasis-open.org/emix/emix/v1.0/cs02/emix-v1.0-cs02.html#_Toc319594576

[2] In North American wholesale electricity markets, transmission rights are bought and sold.

central or cloud-based markets. Central markets discount local decision making while introducing new barriers to resilience. Others rely on a single price-setting supplier. None are interoperable either at the system level or for the actors involved.

Turning back to the more general Transactive Resource Management, nothing in CTS restricts its use to electricity-based markets. Natural gas markets share many characteristics with electricity markets. Local thermal energy distribution systems can balance electricity markets while having their own surpluses and shortages.

Progress toward TE can be accelerated if a common interaction model is used across systems. Looked at from another perspective, a client written for a participant in one such system should be able to interoperate with another TE system. The Common Transactive Services from The Energy Mashup Lab fulfil that promise.

1.1 Generality of the Common Transactive Services

CTS can be used to trade (Tender, Transact on) any [Transactive Resource]. While our focus is generally on electrical energy or power, in the rest of this document we use **[power]** to mean *electrical energy or power or any other Transactive Resource*.

The actual product in EML-CTS (next section) is implicit in the market with which one communicates. This limits complexity of product definition to a useful level, so market and product definition may be considered configuration rather than data.

1.2 Application of the Common Transactive Services

The purpose of this specification is to codify the common interactions and messages required for simple markets, hence for simple transactive energy markets. Any system able to use CTS should be able to interoperate with any CTS-conforming market with minimal or no change.

CTS defines communications between market actors and does not define the market or the device controls. Autonomous market actors must be able to recognize patterns and make choices to best support their own needs. Actors need not share details of their internal operations with others.

CTS is valuable for creating micromarkets to manage power within microgrids. Micromarkets support the capability for dynamic restructuring of grids for fault resilience and efficiency [GridFaultResilience]. Micromarkets contain complexity by abstracting interactions to the few common messages of CTS.

CTS does not presume a market with a single seller (e.g., a utility). CTS recognizes two parties to a transaction, and the role of any party can switch from buyer to seller from one transaction to the next. Each Resource Offer (Tender) has a Buy or Sell side attribute. We assume that when each transaction is committed (once power has been purchased) it is owned by the purchaser, and it can be re-sold as desired or needed.

A CTS-operated micromarket may balance power over time in a traditional distribution system attached to a larger power grid or it may bind to and operate a stand-alone autonomous microgrid [BusinessCase].

1.3 The EML-CTS System

In 2015, the US National Institute for Standards and Technology (NIST) began the Transactive Energy Modeling and Simulation Challenge (TE Challenge). A report delivered to TE Challenge in 2016 [CTS2016] defined a minimal subset of Energy Interoperation, which became known as the Common Transactive Services.

In 2019, The Energy Mashup Lab, working with NIST, began developing an open source software system (Apache 2.0 license) that uses a robust financial or "order book" market for peer-to-peer transactions. The system architecture separates market interactions from the actors buying and selling power. The architecture also permits changing the market engine itself. This system is called EML-CTS and is available today.[3]

TE demonstrations have used different market engines, including double auction markets. EML-CTS was designed to be able to use any (e.g. either, both, or some other market engine) while keeping interactions between systems and the market unchanged.

The EML-CTS 1.0 implementation uses Java class definitions similar to those in the UML in this specification. Messages are sent using REST POST operations, and JSON serialization uses the Java classes.

1.4 Terminology

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

1.5 Normative References

[EMIX] *Energy Market Information Exchange (EMIX) Version 1.0*, January 2012, OASIS Committee Specification 02, Latest version: http://docs.oasis-open.org/emix/emix/v1.0/emix-v1.0.html.

[EnergyInterop] *Energy Interoperation Version 1.0*. Edited by Toby Considine. 11 June 2014. OASIS Standard. http://docs.oasis-open.org/energyinterop/ei/v1.0/energyinterop-v1.0.html.

[JSON] JavaScript Object Notation and JSON Schema. https://cswr.github.io/JsonSchema/

[RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.

[RFC2246] T. Dierks, C. Allen *Transport Layer Security (TLS) Protocol Version 1.0*, http://www.ietf.org/rfc/rfc2246.txt, IETF RFC 2246, January 1999.

[SBE] Simple Binary Encoding Technical Specification 1.0. FIX Trading Community, June 16, 2016. https://www.fixtrading.org/standards/sbe/

[WS-Calendar-PIM] *WS-Calendar Platform Independent Model (PIM)* Version 1.0. Edited by William Cox and Toby Considine. 21 August 2015. OASIS

[3] https://github.com/EnergyMashupLab/eml-cts

Committee Specification. http://docs.oasis-open.org/ws-calendar/ws-calendar-pim/v1.0/ws-calendar-pim-v1.0.html.

[Streams] *Schedule Signals and Streams Version 1.0.* Edited by Toby Considine and William T. Cox. 18 September 2016. OASIS Committee Specification. http://docs.oasis-open.org/ws-calendar/streams/v1.0/streams-v1.0.html.

[XSD] *W3C XML Schema Definition Language (XSD) 1.1*. Part 1: Structures, S Gao, C. M. Sperberg-McQueen, H Thompson, N Mendelsohn, D Beech, M Maloney http://www.w3.org/TR/xmlschema11-1/, April 2012, Part 2: Datatypes, D Peterson, S Gao, A Malhotra, C. M. Sperberg-McQueen, H Thompson, P Biron, http://www.w3.org/TR/xmlschema11-2/ April 2012

1.6 Non-Normative References

[Actors] C. Hewitt, "Actor Model of Computation: Scalable Robust Information Systems," arxiv.org, 2010.

[Framework] National Institute of Standards and Technology, *NIST Framework and Roadmap for Smart Grid Interoperability Standards, Release 1.0*, January 2010, http://nist.gov/public_affairs/releases/upload/smartgrid_interoperability_final.pdf

[CTS2016] Cox, W. T., Cazalet, E., Krstulovic, A., Miller, W., & Wijbrandi, W. *Common Transactive Services*. TESC 2016. Available at http://coxsoftwarearchitects.com/Resources/TransactiveSystemsConf2016/Common%20Transactive%20Services%20Paper%2020160516.pdf

[EML-CTS] Energy Mashup Lab Common Transactive Services (open source software) https://github.com/EnergyMashupLab/eml-cts)

[FSGIM] *Facility smart grid information model*. ISO 17800. https://www.iso.org/standard/71547.html 2017

[iCalendar] *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*, https://tools.ietf.org/html/rfc5545. 2009, B. Desruisseaux, See also *Calendar Availability*, https://tools.ietf.org/html/rfc7953, C. Daboo, M. Douglas. 2016

[SmartGridBusiness] Toby Considine and William Cox, *Smart Loads and Smart Grids—Creating the Smart Grid Business Case*. Grid-Interop 2009. Available at http://coxsoftwarearchitects.com/Resources/Grid-Interop2009/Smart%20Loads%20and%20Smart%20Grids.pdf

[StructuredEnergy] *Structured Energy: Microgrids and Autonomous Transactive Operation*, http://coxsoftwarearchitects.com/Resources/ISGT_2013/ISGT-Cox_StructuredEnergyPaper518.pdf . Innovative Smart Grid Technologies 2013 (IEEE).

[GridFaultResilience] William Cox and Toby Considine, *Grid Fault Recovery and Resilience: Applying Structured Energy and Microgrids*. IEEE Innovative Smart Grid Technologies 2014. Available at http://coxsoftwarearchitects.com/Resources/ISGT_2014/ISGT2014_GridFaultRecoveryResilienceStructuredMicrogrids_Paper.pdf

[TRM] B. Huberman and S. H. Clearwater, *Thermal markets for controlling building environments*, Energy Engineering*,* vol. 91, no. 3, pp. 26- 56, January 1994.

[UML] Object Management Group, *Unified Modeling Language (UML), V2.4.1*, August 2011. http://www.omg.org/spec/UML/2.4.1/

1.7 Naming Conventions

This specification follows some naming conventions for artifacts defined by the specification, as follows:

For the names of elements and the names of attributes within XSD files and UML models, the names follow the lowerCamelCase convention, with all names starting with a lower-case letter. For example,

```
<element name="componentType" type="ei:ComponentType"/>
```

For the names of types within XSD files, the names follow the UpperCamelCase convention with all names starting with a lower-case letter prefixed by "type-". For example,

```
<complexType name="ComponentServiceType">
```

For clarity in UML models the suffix "type" is not always used.

For the names of intents, the names follow the lowerCamelCase convention, with all names starting with a lower-case letter, EXCEPT for cases where the intent represents an established acronym, in which case the entire name is in upper case.

JSON and where possible SBE names follow the same conventions.

1.8 Editing Conventions

For readability, element names in tables appear as separate words. The actual names are lowerCamelCase, as specified above, and as they appear in the UML models, and in the XML and JSON schemas.

All elements in the tables not marked as "optional" are mandatory.

Information in the **Meaning** column of the tables is normative. Information appearing in the **Notes** column is explanatory and non-normative.[4]

Examples and Appendices are non-normative.

1.9 Architecture

Services requests and responses are public actions of each interoperating system. Service actions are independent from private actions behind the interface (i.e., device control actions). A service is used without needing to know all the details of its implementation. Services are generally paid for results, not effort.

[4] In ISO and IEC terminology, portions that are not normative are *informative*. We follow the OASIS approach.

1.9.1 Security Considerations

Loose integration using the SOA style assumes careful definition of security requirements between partners. Size of transactions, costs of failure to perform, confidentiality agreements, information stewardship, and even changing regulatory requirements can require similar transactions be expressed within quite different security contexts. It is a feature of the SOA approach that security is composed in to meet the specific and evolving needs of different markets and transactions. Security implementation is free to evolve over time and to support different needs. The Common Transactive Services allow for this composition, without prescribing any particular security implementation.

1.9.2 CTS Extended Example

As an extended example, using the Common Transactive Services, a microgrid is comprised of a number of interacting nodes (parties). Those parties interact in a micromarket co-extensive in scope with the microgrid. No actor reveals any internal mechanisms, but only its interest in buying and selling power.

CTS can also be used for the fractal integration of microgrids. Any micromarket can be bound to or co-extensive with a node in a larger microgrid. A micromarket participating in this way exposes only its aggregate market position. Any participant in CTS effectively aggregates resource it logically contains.

In a similar way, in considering a topology of microgrids, any participant in the original micromarket MAY itself represent a contained autonomous microgrid or, in fact, any autonomous entity whether or not it is managed in turn by a market. [StructuredEnergy][SmartGridBusiness]

2 Overview of Common Transactive Services

2.1 Scope of Common Transactive Services

CTS engages Transactive Resources, e.g. Distributed Energy Resources (DER) and any provider or consumer of energy, while making no assumptions as to their processes or technology.

This specification supports agreements and transactional obligations, while offering flexibility of implementation to support specific approaches and goals of the various participants.

No particular agreements are endorsed, proposed or required in order to implement this specification. Energy market operations are beyond the scope of this specification although interactions that enable management of the actual delivery and acceptance are within scope but not included in CTS 1.0.[5]

As shown in [CTS2016] the Common Transactive Services with suitable product definitions can be used to communicate with essentially any market.

2.2 Specific scope statements

Interaction patterns and service definitions to support the following are in scope for Common Transactive Services:

- Interaction patterns to support transactive energy.
- Information models for price and product communication.
- Payload definitions for Common Transactive Services

The following are out of scope for Common Transactive Services:

- Requirements specifying the type of agreement, contract, product definition, or tariff used by a particular market.
- Computations or agreements that describe how power is sold into or sold out of a market.

Section 1 describes standard bindings, which may be extended by The Energy Mashup Lab or others in the future.

2.3 Assumptions

2.3.1 Conformance with Energy Interoperation

OASIS Energy Interoperation [Energy Interop] Transactive Services is the basis for CTS, which draws definitions of actors, parties, and transactive interactions from the Energy Interoperation TEMIX profile.

[5] See e.g. Energy Interoperation EiDelivery Service https://docs.oasis-open.org/energyinterop/ei/v1.0/os/energyinterop-v1.0-os.html#_Toc388604056

Energy Interop assumes an Energy Services Interface (ESI) as the external face of the energy-consuming or supplying node. Energy Interop defines an interaction model in which there is no direct interaction across the ESI; this characteristic is shared by CTS.

2.3.2 Conformance with EMIX

This specification uses models and artifacts simplified from and in the style of OASIS Energy Market Information Exchange [EMIX] to communicate product definitions, quantities, and prices. EMIX provides a succinct way to indicate how prices, quantities, or both vary over time.

The EMIX product definition, as included in the Transactive Resource, is implied in CTS 1.0. Future CTS specifications may include market context from EMIX and EnergyInterop, as well as other information on products and markets including market terms.

2.3.3 Conformance with WS-Calendar Streams

The WS-Calendar specifications[6] express sequences and enables negotiation of schedules in a manner that is semantically compatible with human schedules, i.e., [iCalendar]. The WS-Calendar Platform Independent Model (PIM) [WsCalendar-PIM] defines common semantics for the specifications. WS-Calendar is the standard under the NIST Smart Grid Roadmap for all such communication.

WS-Calendar is used to describe products whose value changes with time of delivery, and again into Energy Interoperation, which uses Transactive Resources.

This specification bases its representation of single intervals on Schedule Signals and Streams [Streams], a WsCalendar-PIM conforming specification for expressing consecutive occurrences of schedules or products.

CTS 1.0 transacts a single interval at a time, expressed as a single-interval Stream. Energy systems supported by CTS-based markets may express their requirements and capabilities over time using multi-interval Streams or in separate single-interval Streams.

2.3.4 Compatibility with Facilities Smart Grid Information Model

The Facilities Smart Grid Information Model [FSGIM] was developed to define the power capabilities and requirements of these systems over time. FSGIM uses the semantics of WS-Calendar and EMIX to construct its information models for [power] use over time. These sequences of [power] requirements are referred to as load curves. Load curves can potentially be relocated in time, perhaps delaying or accelerating the start time to get a more advantageous price for [power]. These load curves are the basis upon which a TE Agent would base its market decisions.

The Architecture of EML-CTS is premised on distinct physical systems being able to interoperate by coordinating their production and consumption of **[power]**[7] irrespective of their ownership, motivations, or internal mechanisms. This specification defines messages and interactions of that interoperation.

---

[6] See Section 1.5 Normative References

[7] See Section 1.1.

CTS transactions are semantically aligned with FSGIM load requests. CTS 1.0 uses Streams to express single-interval tenders in anticipation of the use of Streams in FSGIM-conformant communications.

2.4 Common Transactive Services Architecture

The implied CTS architecture is drawn from and is a subset and simplification of the architecture presented in [EnergyInterop]. Specifically, the Energy Interoperation architecture uses the Service-Oriented Architecture (SOA) model which has become the consensus view for energy.

The Energy Mashup Lab uses the Actor Model, which can be implemented in SOA with a few lightweight Service Operations. The Lab adapted the SOA model of Energy Interoperation into an Actor-to-Actor model that requires fewer and lighter weight messages.

The Actor Model names a style of system integration used for high scalability and resilience.[8] The Actor Model uses a small number of simple messages to coordinate behavior among simple agents termed Actors. Note that Actors need not be actually simple; Actors are unable to present complexity because the messages are so simple.

Simple messages are an essential aspect of actor architectures. The Common Transactive Services are a lightweight profile of the OASIS Energy Interoperation specification. All CTS messages are simple, and make no assumptions about the systems behind the messages.

Just as the market participants present simple messages, so too, does the market. The internals of a market contain a market engine to match tenders, and to declare contracts. The rules used to match tenders could be nearly instant order book, or periodic double auction, or some other model. This complexity is hidden. The market receives tenders and announces contracts. Only the simple messages of CTS are used.

All interactions described in CTS are as defined in [EnergyInterop]. That specification describes interactions between pairs of actors, and, in a deployment, relationships are established among actors. Actors may perform in pairwise chains of actors.

All interactions and actors below are described as if for Actors in electrical energy markets. For use in other transactive energy markets, or even transactive resource markets, only the product or resources would be changed.

An actor takes on a role, for example a business role as a Party. In the UML model, *PartyId* and *CounterPartyId* inherit from *ActorId* which in turn inherits from class *UidType*.

2.4.1 Sides in Tenders and Transactions

At any moment, a Party has a *position* which represents the cumulative amount of power (or other product) that an actor has previously transacted for that time interval.

A Party can take one of two Sides in a given Transaction:

- Buy, or

---

[8] See C. Hewitt, "Actor Model of Computation: Scalable Robust Information Systems," arxiv.org, 2010, or C. Hewitt, "A Universal Modular Actor Formalism for Artificial Intelligence," ICJA, 1973, or many other references

- Sell

A Party selling [power] relative to its current position takes the Sell Side of the Transaction. A Party buying [power] relative to its current position takes the Buy Side of the Transaction.

From the perspective of the market, there is no distinction between a party selling additional power and party selling from its previously acquired position. An Actor representing a generator generally takes the Sell side of a transaction. An Actor representing a consumer generally takes the Buy side of a transaction. A generator may take the Buy Side of a Transaction in order to reduce its own generation, in response either to changes in physical or market conditions or to reflect reflecting other commitments made by the actor. A consumer may choose to sell from its current position if its plans change, or if it receives an attractive price. A power storage system actor may choose to buy or sell from interval to interval, consistent with its operating and financial goals.

We do not specify how the [power] is delivered. For example, a long distance transfer might be implemented with the seller selling power to its local grid and the buyer buying power from its local grid, with financial reconciliation producing the same result as a direct sale and deliver.

2.4.2 Semantic Composition

The semantics and interactions of CTS are selected from and derived from [EnergyInterop].

Energy Interoperation incorporates two other standards, [EMIX] and [WS-Calendar], and uses an early Streams definition.

- EMIX describes price and product for electricity markets.
- WS-Calendar communicates schedules and sequences of operations. This specification uses the [Streams] optimization which is a standalone specification, rather than part of Energy Interoperation 1.0.
- Energy Interoperation uses the vocabulary and information models defined by those specifications to describe the services that it provides. The payload for each Energy Interoperation service references a product defined using [EMIX]. EMIX schedules and sequences are defined using [WS-Calendar]. Any additional schedule-related information required by [EnergyInterop] is expressed using [WS-Calendar].
- Since [EnergyInterop] was published, a semantically equivalent but simpler [Streams] specification was developed in the OASIS WS-Calendar Technical Committee[9]. CTS uses that simpler [Streams] specification.

In effect, CTS is a profile of Energy Interoperation but with simplified information models and defines only payloads, not the messaging.

CTS 1.0 supports a single product market; in other words, product definitions in CTS 1.0 are implicit. Most markets will be better served with multiple products, for example, a 1-hour market for power alongside a 5-minute market for power.

Future development of CTS may include a discoverable market description profiled from the EMIX Market Context. The EMIX Market Context defines market rules and catalogs the

[9] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-calendar

products tradeable in the market. Future versions of CTS will support multiple markets, hence multiple products.

All terms used in this specification are as defined in their respective specifications.

2.5 Products and Instruments

A CTS Product is a specific resource packaged in a specific quantity for a specific duration. An example is 10 kW of power over the duration of an hour. An actor wanting to buy or sell 100 kW of power during an hour would tender 10 units of that product.

The CTS architecture transacts in power products at specific times, referred to as Instruments. Tenders become Contracts when a tender to buy a product at a specific time, and a tender to sell a product at that same specific time are matched. These are two tenders for the same instrument

3 Services and Operations

This section re-iterates terms and simpler models from [Energy Interop]. That specification is normative.

For each service operation there is an actor that *invokes* the service operation and one that *provides* the service. These roles are indicated by the table headings *Service Consumer* for the actor or role that consumes or invokes the service operation named in the *Operation* column and *Service Provider* for the actor or role that provides or implements the service operation as named in the *Operation* column.

This terminology is used through all service definitions presented in this specification.

The column labeled *Response Operation* lists the name of the service operation invoked as a response. Most operations have a response, excepting primarily those operations that broadcast messages. The roles of *Service Consumer* and *Service Provider* are reversed for the *Response Operation*.

For transactive services any party may receive tenders (priced offers) of service and possibly make tenders (priced offers) of service.

Any party using transactive energy services may own generation or distributed generation or reduce or increase energy from previously transacted energy amounts. These activities are not identified in transactive services. The dispatch of these resources and the use of energy by a party are influenced by tenders between Parties that may result in new Transactions and changes in operations.

The next section describes the roles in a transactive approach tendering and prices are used by parties to discover and negotiate transactions that respect the preferences of each party and energy usage, generation, storage and controllability directly available to each party.

3.1 Structure of Common Transactive Services and Operations

The Common Transactive Services presented in this specification are only

- Transactive Services—for implementing transactions and tenders

We include UML definitions for the standard payloads for service requests, rather than the service, communication, or other characteristics. In Section 5 we describe standard serialization for the CTS standard payloads; additional bindings may be used by conforming implementations.

3.2 Naming of Services and Operations

The naming of services and operations follows a pattern. Services are named starting with the letters ***Ei*** following the Upper Camel Case convention. Operations in each service use one or more of the following patterns. The first listed is a fragment of the name of the initial service operation; the second is a fragment of the name of the response message which acknowledges receipt, describes errors, and may pass information back to the invoker of the first operation.

Create—Created An object is created and sent to the other Party

Cancel—Canceled A previously created request is canceled

For example, to construct an operation name for the EiEvent service, "Ei" is concatenated with the name fragment (verb) as listed. For example, an operation to cancel an outstanding Tender is called *EiCancelTender*.

3.3 Payloads and Messages

We define only the payloads; the particular networking technique and message structure is determined by the applications sending and receiving CTS payloads.

3.4 Description of the Services and Operations

The sections below provide the following for each service:

- Service description
- Table of operations
- Interaction patterns for the service operations in graphic form
- Information model using [UML] for key artifacts used by the service
- Operation payloads using [UML] for each operation

3.5 Responses

In a service interaction, responses may need to be tracked to determine if the transaction is successful or not. This may be complicated by the fact that any given transaction may involve the transmission of one or more information objects.

A Response returns the success or failure of the entire operation, with possible detail included if there is more than one bundled operation (e.g. EiCancelTender with multiple tenders).

It is MANDATORY to return errors in responses. It is MANDATORY in CTS to return successes in responses.[10]

The class diagram reflects the generic response in CTS 1.0.

The description of EiResponseType is from Energy Interoperation, changing only the cardinality of responseDescription (to zero, that is, not passed).

[10] This contrasts with Energy Interoperation, where it is not mandatory to return any responses if the entire EiCancelTender service operation was completed successfully. The pattern in Energy Interoperation is to return those that have failed (required) and those that succeeded (optional).

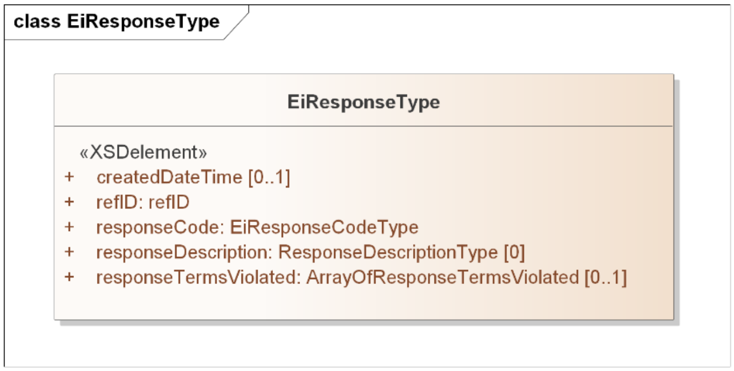

Figure 3-1: Example of generic error response for a service operation

The attributes of EiResponseType are in the following table.

Table 3-1: EiResponse Attributes

Attribute	Meaning	Notes
Created DateTime	Optional timestamp indicating the date and time when this EiResponse was created	
RefId	Reference ID which identifies the artifact or message element to which this is a response. refId serves as a correlation ID[11].	
Response Code	The Response Code indicates success or failure of the operation requested. The Response Description is unconstrained text, perhaps for use in a user interface. The code ranges are those used for HTTP response codes,[12] specifically 1xx: Informational - Request received, continuing process 2xx: Success - The action was successfully received, understood, and accepted 3xx: Pending - Further action must be taken in order to complete the request 4xx: Requester Error - The request contains bad syntax or cannot be fulfilled 5xx: Responder Error - The responder failed to fulfill an apparently valid request	
Response Description	The Response Description is in the model but profiled to be cardinality 0..0.	Not present in CTS 1.0 payloads

[11] As an example of the *Correlation Pattern* for messages

[12] See e.g. https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Response Terms Violated	The Terms Violated by the request to which this is a response. Conforming CTS 1.0 implementations SHALL omit this attribute.	Market Terms and Market Context may be implemented in a future release.

There is no exhaustive list of all possible Response Codes. The Response Codes are intended to enable even the smallest device to interpret Response. This specification uses a pattern consisting of a 3-digit code, with the most significant digit sufficient to interpret success or failure. This pattern is intended to support that smallest device, while still supporting more nuanced messages that may be developed.

While the only value after the leading digit the Response Code defined in Energy Interoperation is 00, conforming specifications may extend these codes to define more fine-grained response codes. These should extend the pattern above, though. A response code such as 403 should always be within the realm of Requester Error.

EML-CTS uses error code 200 for success.

4 Transactive Services

Transactive Services define and support the lifecycle of transactions from initial Tender to final settlement. The phases described in [Energy Interop] are

- Registration—to enable further phases. (Not part of CTS)
- Pre-Transaction —binding tenders for transactions. (Part of CTS)
- Transaction Services—execution and management of transactions. (Part of CTS)
- Post-Transaction—settlement, energy used or demanded, payment, position. (Not part of CTS)

For transactive services, the roles are **Parties** and **Counterparties**.

The terminology of this section is that of business agreements: tenders and transaction. The Service descriptions and payloads are simplified and updated from those defined in Energy Interoperation.

4.1 Pre-Transaction Services

Pre-transaction services are those between parties that may prepare for a transaction. The service in CTS is EiTender with two service operations.

Tenders and transactions are artifacts based on **[EMIX]** artifacts suitably flattened and simplified, and which contain schedules and prices in varying degrees of specificity or concreteness.

Table 4-1: Pre-Transaction Tender Services

Service	Operation	Response	Service Consumer	Service Provider	Notes
EiTender	EiCreateTender	EiCreatedTender	Party	Party	Create and send Tender
EiTender	EiCancelTender	EiCanceledTender	Party	Party	Cancel one or more Tenders

4.1.1 Interaction Pattern for the EiTender Service

Figure 4-1 presents the [UML] sequence diagram for the EiTender Service. Note that EiDistributeTender is not part of CTS 1.0, but is being considered for a future release.

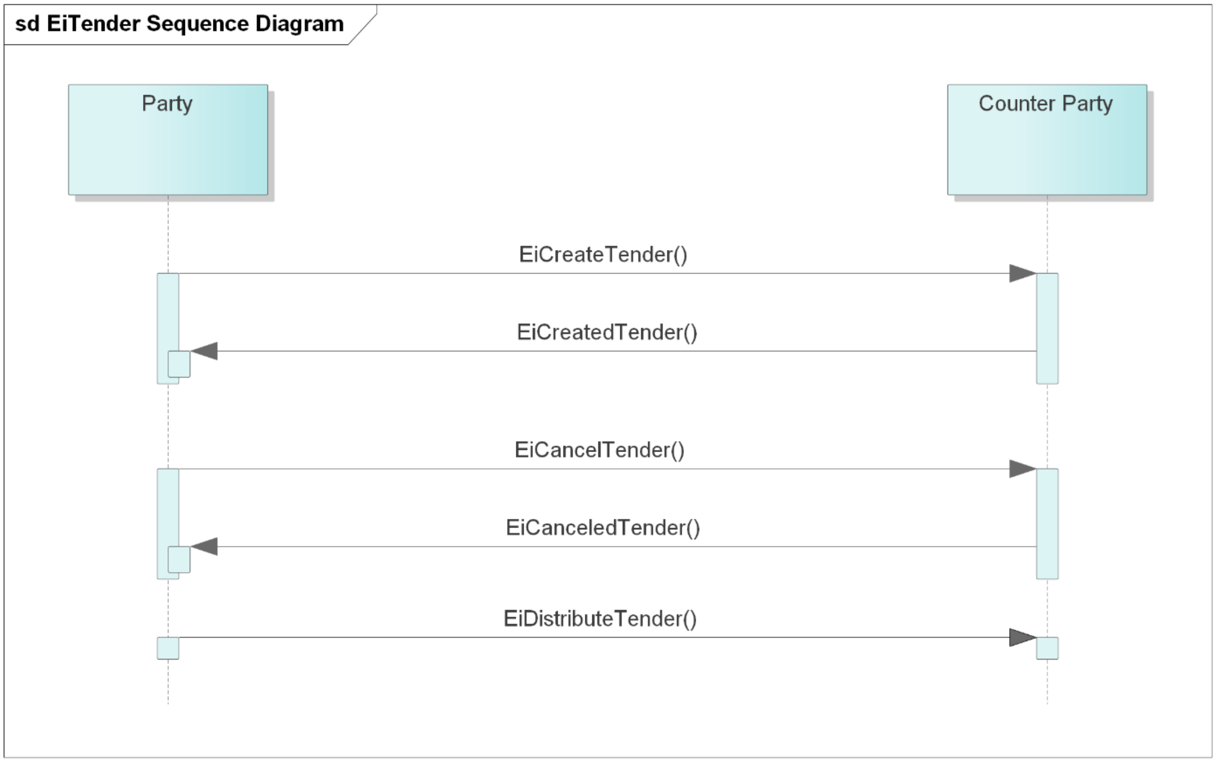

Figure 4-1: UML Sequence Diagram for the EiTender Service

4.1.2 Information Model for the EiTender Services

The information model for the EiTender Service artifacts follows that of **[EMIX]**, but flattened and with product definition implied by the implementation.

Time interval, price, and quantity are key elements for a product; the other aspects of product definition (e.g. energy and units) are implicit as described in Section 2.4.2.

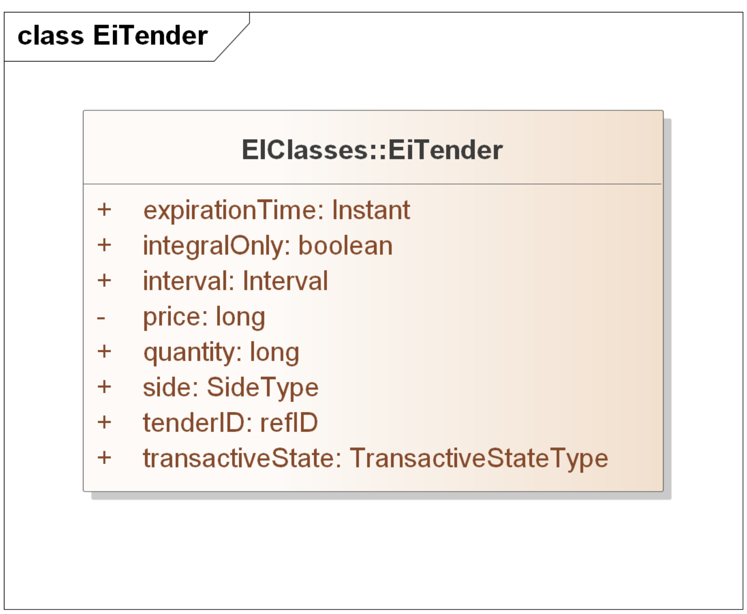

Figure 4-2: Class EiTenderType

The attributes of EiTender are shown in the following table.

Table 4-2: EiResponse Attributes

Attribute	Meaning	Notes
Expiration Time	The date and time after which this Tender is no longer valid.	
Integral Only	All of the Tender must be bought or sold at once; no partial sale or purchase	In EML-CTS set to False
Interval	The time interval for the product being offered	
Price	The unit price for the product being offered	Total price is the product of price and quantity
Quantity	The quantity of the product being offered	Total price is the product of price and quantity
Side	Whether the tender is to buy or to sell the product	
Tender ID	An ID for this tender	
Transactive State	The transactive state of this payload (tender)	See below

Transactive State is a concept from EMIX; it describes the state of an object. For CTS 1.0, only states *tender* and *transaction* are used.

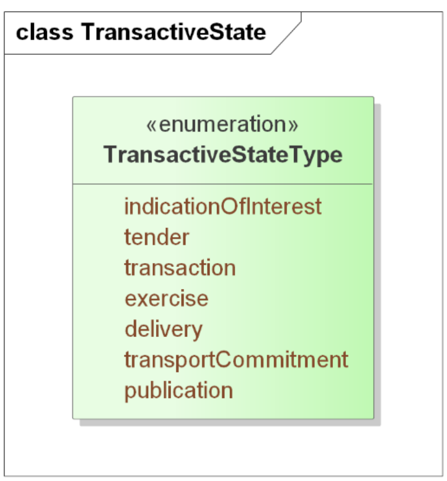

Figure 3-3: Enumeration TransactiveStateType

4.1.3 Operation Payloads for the EiTender Service

The **[UML]** class diagram describes the payloads for the EiTender service operations.

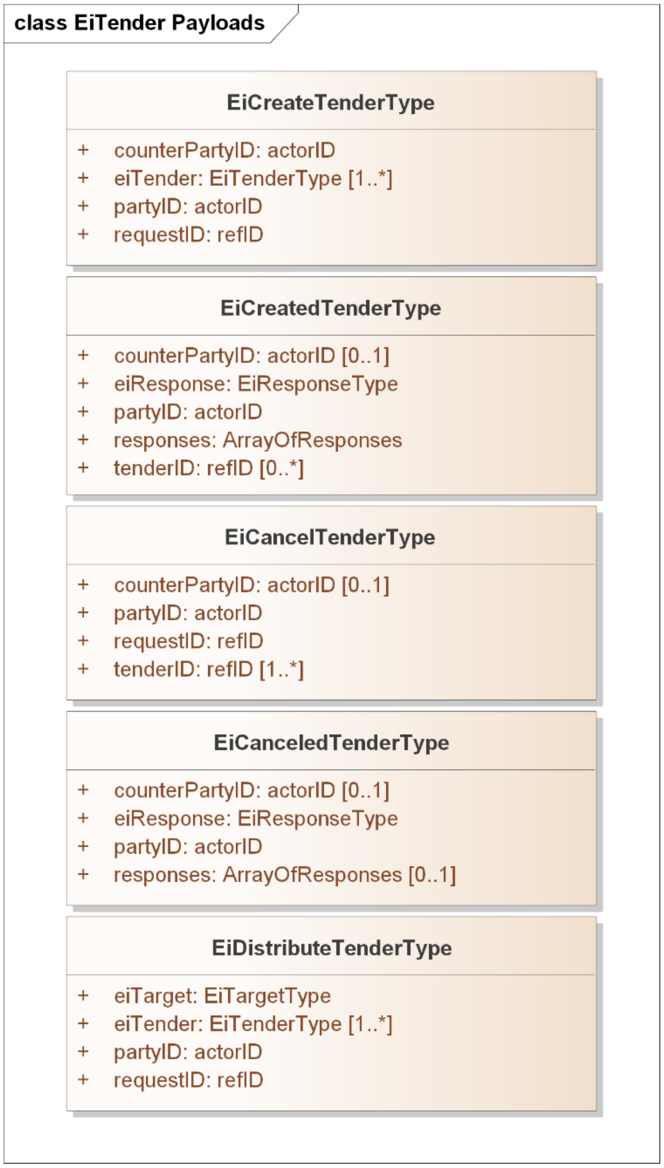

Figure 4-4: UML Class Diagram for the Operation Payloads for the EiTender Service

4.2 Transaction Management Services

The service operations in this section manage the exchange of transactions. The context is implied, or may in the future be made explicit with a Market Context reference (see Section 2.4.2).Note that canceling or modifying transactions are not included in either CTS or Energy

Interoperation. Following the approach in distributed agreement protocols[13], a compensating transaction SHOULD be created as needed to compensate for any effects.[14]

Table 4-3: Transaction Management Service

Service	Operation	Response	Service Consumer	Service Provider	Notes
EiTransaction	EiCreateTransaction	EiCreatedTransaction	Party	Party	Create and send Transaction

4.2.1 Interaction Pattern for the EiTransaction Service

This is the [UML] sequence diagram for the EiTransaction Service:

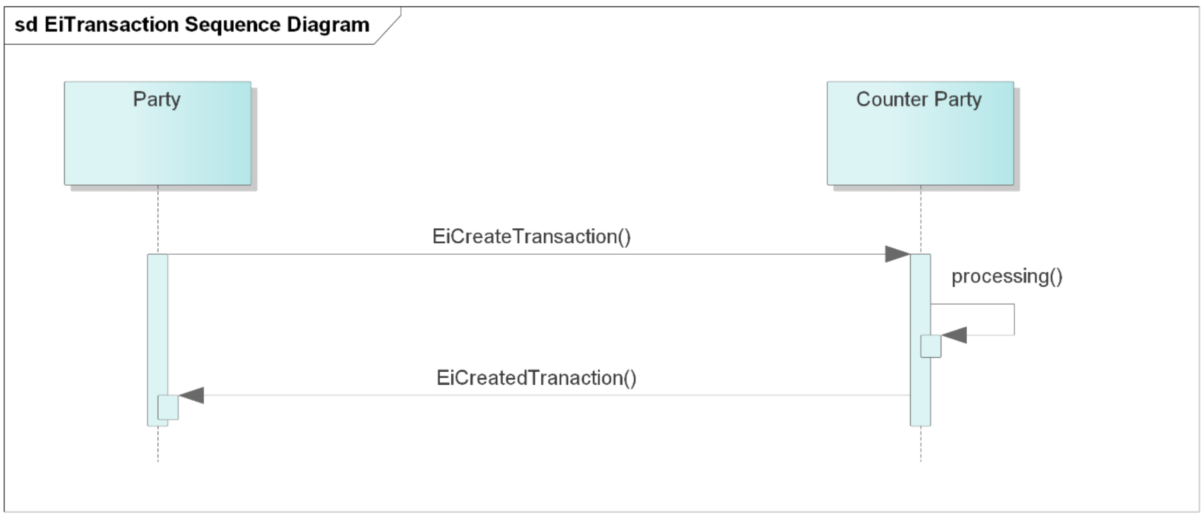

Figure 4-5: UML Sequence Diagram for the EiTransaction Service

4.2.2 Information Model for the EiTransaction Service

Transactions are derived from **[EMIX]** artifacts including a Stream with time, quantity, and price. Flattening similar to that in EiTender is used.

Although an EiTransaction object includes the original EiTender, the EiTransaction carries its own Transactive State.

[13] See, e.g., WS-Transaction and WS-BusinessActivity.

[14] This is consistent with the way that distributed agreement protocols such as [WS-BusinessActivity] manage compensation rather than cancelation.

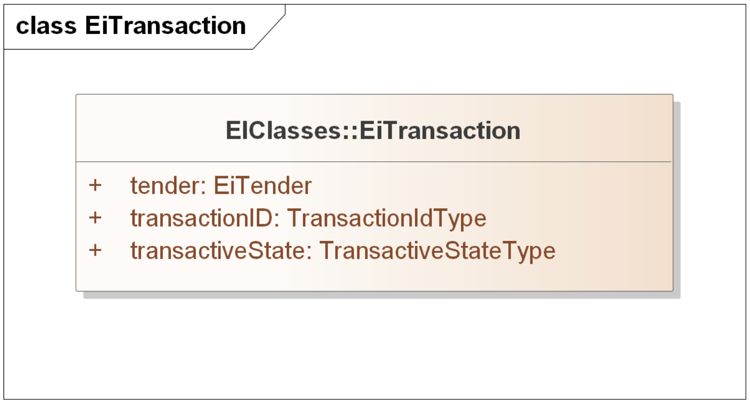

Figure 4-6: UML Class Diagram of EiTransaction

The attributes of EiTransaction are shown in the following table.

Table 4-4: EiTransaction Attributes

Attribute	Meaning	Notes
Tender	The EiTender that led to this Transaction.	The ID, quantity and price may differ from that originally tendered due to market actions.
Transaction ID	An ID for this Transaction	The contained Tender has its own TenderID
Transactive State	The transactive state of this payload (transaction)	See Figure 3-3: Enumeration TransactiveStateType

4.2.3 Operation Payloads for the EiTransaction Service

The **[UML]** class diagram describes the payloads for the EiTransaction service operations.

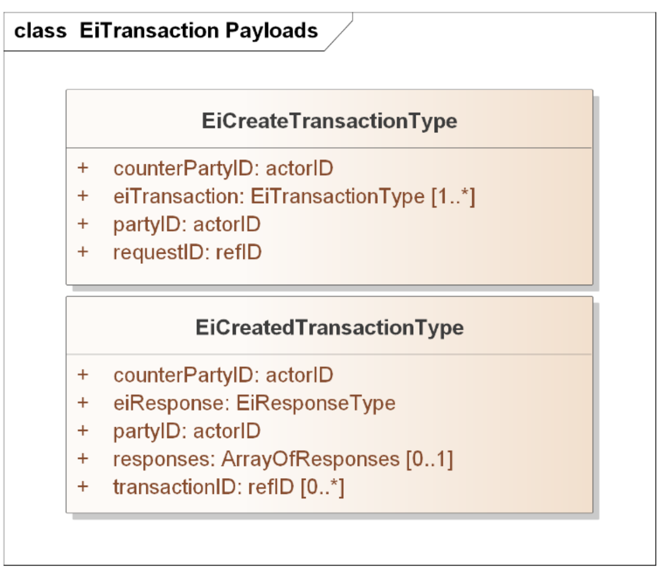

Figure 4-7: UML Class Diagram of EiTransaction Service Operation Payloads

4.3 Comparison of Transactive Payloads

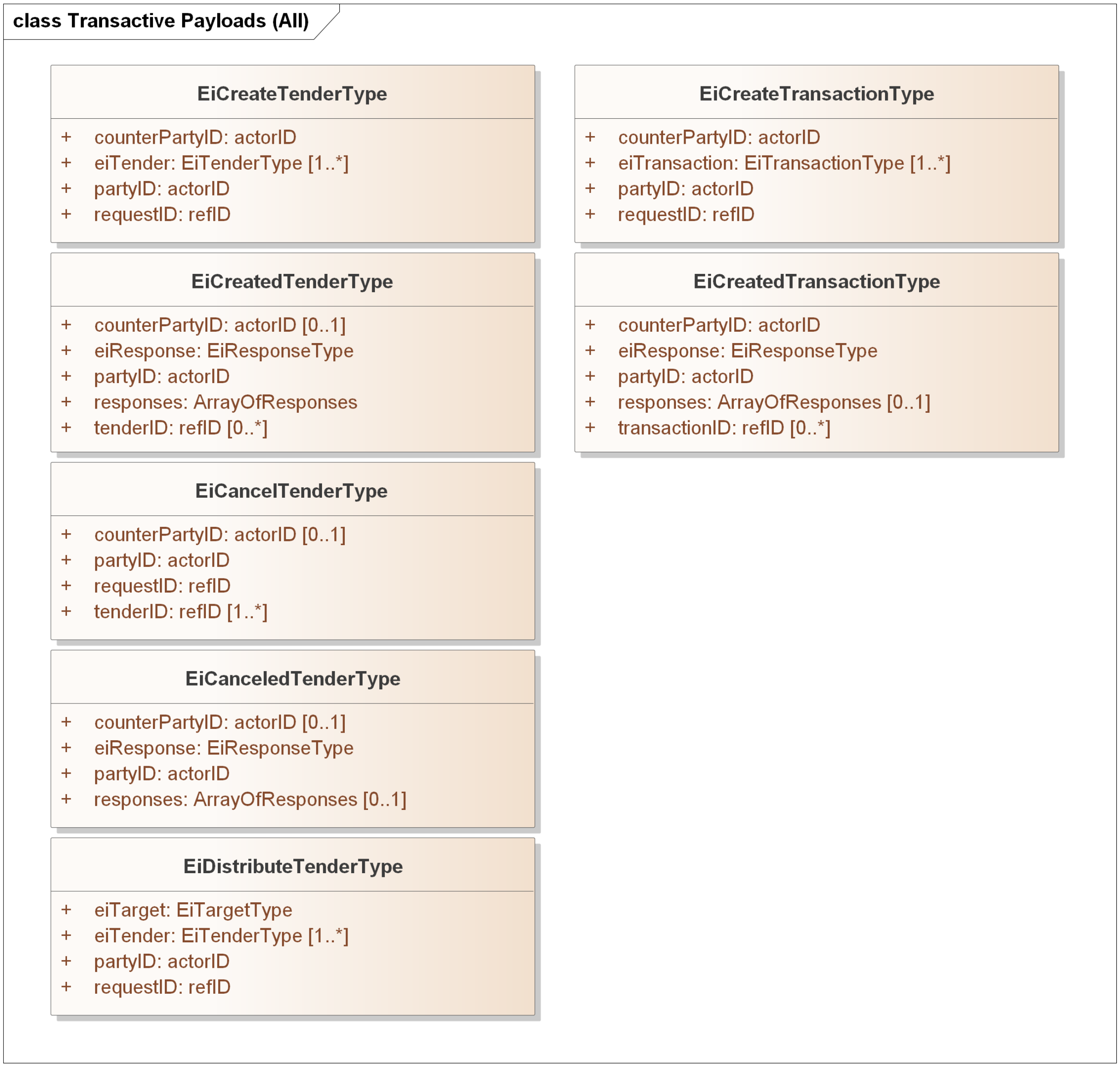

Figure 4-8: UML Diagram comparing all Transactive Payloads

5 Bindings

Payloads and interaction patterns are described in **[UML]** in Section 1 above. This section contains bindings for the payloads in three encoding schemes:

- JSON **[JSON]**
- XML Schema **[XSD]**
- FIX Simple Binary Encoding **[SBE]**

5.1 JSON

TODO—JSON Schema available

5.2 XML Schema

TODO—XML Schema available

5.2.1 XML Namespaces

5.3 Simple Binary Encoding

TODO—Work in progress

6 Conformance and Processing Rules for Common Transactive Services

TODO update in progress

7 Conformance Statements

The Common Transactive Services conform to the base standards as follows.

This section includes conformance statements for the Common Transactive Systems to

- OASIS Energy Interoperation 1.0
- OASIS WS-Calendar MIN
- OASIS WS-Calendar Schedule Streams and Signals

Appendix A. Acknowledgments

This specification began with William Cox leading the Common Transactive Services team in the 2015-2016 NIST Transactive Energy Challenge to define the initial structure of the CTS [CTS2016].

Others picked up and used that work, culminating in a contract from NIST with TC9, Inc and Cox Software Architects LLC to develop agents to support co-simulation of bilateral markets with GridLAB-D™ input models in the NIST Cyberphysical systems modelling platform. That contract required all work to be open source from day one, and all work to be done in the open. TC9 opted to perform the work in the open repositories of The Energy Mashup Lab. NIST has incorporated that code into their TE Market simulation model.

The initial draft of CTS 1.0 (this specification) was based on clarifications and simplifications discovered building the internal services and APIs of that project. The Lab has continued to refine that work through and with the NJIT Capstone Projects.

All work continues in the open GitHub repositories, and all code is licensed under an Apache 2.0 license.

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

- NIST, the National Institute of Standards and Technology, including
    - David Holmberg
    - Thomas Roth
- Members of the WS-Calendar, Energy Market Information Exchange, and Energy Interoperation TCs (see acknowledgement in the respective specifications)
- Members of the NIST Transactive Energy Challenge Common Transactive Services work group (see acknowledgement in the respective specification and paper)
- New Jersey Institute of Technology and the NJIT Capstone program where The Energy Mashup Lab worked with faculty, staff, and teams of Seniors and Masters students, specifically
    - Professor Osama Eljabiri
    - Capstone Executive Team members for each term listed below
    - Team Members Fall 2020: Omair Abdul, Omar Janouk, Matthew Molinari
    - Team members Summer 2020: Indira A. Akkiraju, Josiah Nieves, Alex Shepherd
    - Team members Spring 2020: Matt Amato, Dhruvinkumar Desai, Anupam Saini, Justin Schuster
    - Team members Fall 2019: Rajeev Chanchlan, Jasper Sam David, Mounica Gona, Dhrumil Shah, Karan Shah
- The Energy Mashup Lab, its officers and associates
    - Toby Considine
    - William Cox
    - David A Cohen

Appendix B. Background and Development history

The Common Transactive Services (CTS) are a lightweight profile of the OASIS Energy Interoperation Standard [EnergyInterop].

The Energy Interoperation Technical Committee was formed to define the necessary interactions between Smart Grids and their end nodes, including Smart Buildings, Enterprises, Industry, Homes, and Vehicles. The specification defines data and communication models that enable standard exchange of signals for dynamic pricing, reliability, and emergencies. Energy Interoperation supports market-based balancing of energy supply and demand while increasing fluidity of contracts.

In 2015, the US National Institute for Standards and Technology (NIST) began the Transactive Energy Modeling and Simulation Challenge (TE Challenge). A report delivered to the TE Challenge and a paper delivered to the Transactive Energy Systems Conference **[TESC2016]** defined a minimal subset of Energy Interoperation, which became known as the common transactive services. The report further showed commonality between the messages of existing TE systems, including several not based on Energy Interoperation.

The Energy Mashup Lab has created an open source implementation using the Common Transactive Services called EML-CTS[15], which has in turn helped us to further simplify the original description of CTS and led to this evolved specification.

The EML-CTS v1.0 system uses CTS message payloads expressed in JSON for all market communications. The Lab plans to contribute this specification to the OASIS Energy Interoperation Technical Committee as the basis for work on a standard lightweight specification for The Common Transactive Services.

[15] https://github.com/EnergyMashupLab/eml-cts

Appendix C. Glossary

No definition in this glossary supplants normative definitions in this or referenced specifications. They are here merely to provide a guidepost for readers at to terms and their special uses. Implementers will want to be familiar with all referenced standards.

Actor is an architectural component that interacts with other actors. Actors may take on roles, e.g. as a Party in a transaction.

Agreement is broad context that incorporates market context. Agreement definitions are out of scope in the Common Transactive Services.

EMIX: As used in this document, EMIX objects are descriptions applied to a WS-Calendar Sequence. EMIX defines Resource capabilities, used in tenders to match capabilities to need, and in Products, used in tenders and in specific performance and execution calls. Please note that CTS uses more recent WS-Calendar specifications than that used in EMIX, and that the product definition in CTS 1.0 is implicit.

Party or Transactive Party is a role that an actor may take. In the EML-CTS implementation, the Local Market Agent (LMA) is not a party, but the Transactive Energy User Agent (TEUA) is a party and represents its Energy Manager.

Resource (as defined in EMIX[16]): A Resource is something that can describe its capabilities in a Tender into a market. How those Capabilities vary over time is defined by application of the Capability Description to a WS-Calendar Sequence. See [EMIX].

Stream: A set of contiguous intervals of the same size. See [Streams]

Tender: A tender is an offering for a Transaction. See Transaction.

Transaction: A binding commitment between parties entered into under an agreement.

---

[16] See http://docs.oasis-open.org/emix/emix/v1.0/cs02/emix-v1.0-cs02.html#_Toc319594576

Appendix D. Revision History

Revision	Date	Editor	Changes Made
CTS 1.0 Draft of 2020-10-28	2020-10-28	William Cox	First published document. Evolved from OASIS Energy Interoperation Standard, CTS reports and papers, and the EML-CTS project