Energy Research and Development Division

FINAL PROJECT REPORT

Validated and Transparent Energy Storage Valuation and Optimization Tool

Appendix A: StorageVET™ Use Cases
DISCLAIMER

This report was prepared as the result of work sponsored by the California Energy Commission. It does not necessarily represent the views of the Energy Commission, its employees or the State of California. The Energy Commission, the State of California, its employees, contractors and subcontractors make no warrant, express or implied, and assume no legal liability for the information in this report; nor does any party represent that the uses of this information will not infringe upon privately owned rights. This report has not been approved or disapproved by the California Energy Commission nor has the California Energy Commission passed upon the accuracy or adequacy of the information in this report.
ACKNOWLEDGEMENTS

The Electric Power Research Institute (EPRI) prepared this report.

Principal Investigator
R. Entriken

This report describes research sponsored by the California Energy Commission.
ABSTRACT

This document describes use cases for the StorageVET software. Use cases, in general, are descriptions of how users expect to use this software, and thus they take various user perspectives and describe processes for evaluating storage projects, whether existing or future. Existing projects are evaluated to address certain operational decisions, and future projects are typically evaluated for investments and cost recovery purposes.

Studying both existing and future use of storage projects is important, because much about future evaluations is based on an assumption that some sort of storage project will exist and its operations govern almost all revenue and also operating costs.

Use cases are very valuable in the planning, design, and development phases of software and other projects, because they not only offer a means for incorporating user perspectives and priorities into those phases, but they also set the stage for designing the software functionality that enables these uses for the given priorities.

Keywords: Energy Storage, Use Cases, Grid Services, Analysis, Modeling, Valuation

Please use the following citation for this report:

TABLE OF CONTENTS

Acknowledgements ................................................................................................................................ iii

ABSTRACT .............................................................................................................................................. iv

TABLE OF CONTENTS ....................................................................................................................... vi

EXECUTIVE SUMMARY ...................................................................................................................... 1

CHAPTER 1: Introduction ....................................................................................................................... 2

Approach ................................................................................................................................................. 2
About Use Cases .................................................................................................................................... 2
Locations ................................................................................................................................................. 3
Technologies ........................................................................................................................................... 4
Services .................................................................................................................................................... 5
Scenarios .................................................................................................................................................. 7
Report Overview .................................................................................................................................... 7

CHAPTER 2: Use Case Overview .......................................................................................................... 8

A tors ....................................................................................................................................................... 8

StorageVET ......................................................................................................................................... 8
Regulator ............................................................................................................................................. 8
Investors .............................................................................................................................................. 8
Stakeholders ........................................................................................................................................ 8
Program Administrator ..................................................................................................................... 8
Participant ........................................................................................................................................... 8
Researcher ........................................................................................................................................... 9
Market Designer ................................................................................................................................. 9
Vendor ............................................................................................................................................... 9
Storage/DER Operator ....................................................................................................................... 9
Information Exchange ....................................................................................................................... 9
Process ............................................................................................................................................... 9
CHAPTER 5: Operator Use Cases

Use Case 3.1. Reshape Net Load Profile

Use Case 3.2. Co-Optimize Services Scheduling

Use Case 3.3. Determine Left-Over Storage Capacity

Use Case 3.4. Schedule an Outage
EXECUTIVE SUMMARY

This is an interim report. The final report will have an Executive Summary.
CHAPTER 1:
Introduction

This document describes use cases for the StorageVET software. Use cases, in general, are descriptions of how users expect to use this software, and thus they take various user perspectives and describe processes for evaluating storage projects, whether existing or future. Existing projects are evaluated to address certain operational decisions, and future projects are typically evaluated for investments and cost recovery purposes.

Studying both existing and future use of storage projects is important, because much about future evaluations is based on an assumption that some sort of storage project will exist and its operations govern almost all revenue and also operating costs.

Approach

The following use case descriptions are a result of two workshops in September 2015 and December 2015, conducted by the StorageVET team. In the first workshop the team solicited input about how the users and other storage stakeholders may require decision support from a tool as StorageVET is envisioned to be. Those discussions largely formed the outline and most of the body of this report. The second workshop was used to socialize the content of this document and to discuss how the use cases can be used to address the complex aspects of decision support surrounding storage investment, operations, and system design. One of the important conclusions of this second discussion is that the use cases can be considered as chapters of a hyper-linked story, wherein their order depends on what decision a user needs to explore next.

About Use Cases

Use cases are very valuable in the planning, design, and development phases of software and other projects, because they not only offer a means for incorporating user perspectives and priorities into those phases, but they also set the stage for designing the software functionality that enables these uses for the given priorities.

As a potential StorageVET user, one should strive to understand the use case descriptions, verify that they are accurate from the user’s perspective, and decide how important each one will be for the user and the user’s organization over time. Some will be more important in the early stages of use, such as the planning stage, and others will be more important later, during operations.

As a developer of StorageVET, one should strive not only to understand the use cases, but also to imagine the functionality needed to implement them. The developer should imagine how certain functions, perhaps in slightly different forms, will be reused over and over, which is an indicator of core functions that require added flexibility for good results in multiple contexts. Additionally, the developer should think about processes and precedence of functions, so that the functional modules have the right starting and ending conditions and fit together properly.
The use cases have major context parameters:

- **Location** – Location is specified generally according to the grid connection, whether transmission, distribution, or behind the meter. It also may include some aspects of grid integration, like the surroundings in terms of urban or rural setting, or whether there is a significant local load or generation source. Contextual price information is also an important part of the location.

- **Technology** – Storage technologies can be in the form of electrical and thermal storage, and technologies can also include non-storage benchmarks, like a combustion turbine or demand response resource, that are used to help assess performance.

- **Services** – Storage is very flexible, and there is a long list of services it can provide. These are described later in this introduction.

Finally, the combinations of choices for location, technology, and services comprise what we will call a scenario. Many use cases can be supported by configuring various location, technology, and/or service elements as comparative scenarios and evaluating the costs and benefits.

**Locations**


### Table 1: CPUC Storage Grid Domains / Locations

<table>
<thead>
<tr>
<th>Storage Grid Domains</th>
<th>Regulatory Function</th>
<th>Use-Case Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>(Stand-Alone Energy Storage) Ancillary Services, Peaker, Load Following</td>
</tr>
<tr>
<td></td>
<td>Transmission Reliability (FERC)</td>
<td>Voltage Support</td>
</tr>
<tr>
<td>Distribution-Connected</td>
<td>Distribution Reliability</td>
<td>Substation Energy Storage (Deferral)</td>
</tr>
<tr>
<td></td>
<td>Generation/Market</td>
<td>Distributed Generation + Energy Storage</td>
</tr>
<tr>
<td></td>
<td>Dual-Use (Reliability &amp; Market)</td>
<td>Distributed Peaker</td>
</tr>
<tr>
<td>Behind-the-Meter</td>
<td>Customer-Sited Storage</td>
<td>Bill Mgt / Permanent Load Shifting, Power Quality, Electric Vehicle Charging</td>
</tr>
</tbody>
</table>

Source: CPUC  
http://docs.cpuc.ca.gov/PublishedDocs/Published/G000/M079/K533/79533378.PDF  
Table 1.
The locational information can include additional factors that reflect policy and economics. For instance, the California Independent System Operator (CAISO) publishes wholesale prices for electrical energy and ancillary services (a subset of Services) on a nodal basis for supply and a wider-area (zonal) basis for load. Depending on the grid connection and the rate structure, these prices can vary significantly. Even behind-the-meter storage may be affected by CAISO nodal prices.

The energy and services are heavily dependent on factors that may further help define scenarios, like fuel prices and energy policy. Additionally, they may be affected by emission policies, should certain emission limits be reached or should emission taxes be added to these prices.

**Technologies**

The technology configuration establishes the power and energy capacities of the storage system as well as efficiency and performance characteristics. StorageVET includes a framework of technical parameters that can characterize a wide range of storage technologies including pumped hydro, compressed air energy storage (CAES), static batteries, flow batteries, ultracapacitors, superconducting magnetic energy storage (SMES), thermal storage (rooftop ice or chilled water systems, molten salt, or water heaters), dispatchable hydro, EV batteries, liquid air, and power-to-gas systems.

Non-storage technologies can also be configured to serve as benchmarks. These include combustion turbines, diesel backup generators, and demand response programs.

Figure 1 illustrates the most general case of energy flows considered by the technology configuration framework.
Efficiencies and capacity limitations are key inputs in determining the range of eligible services and net load objectives the system can deliver. Specific storage technologies generally include only a subset of these energy flows. Additional technical parameters not included in this diagram include maximum ramp rate, minimum charge/discharge, degradation characteristics (by cycle or by time), and dependencies on ambient temperature.

**Services**

Services are defined by the StorageVET services list, which is presented in Table 2.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Timing of Decision</th>
<th>Grid Service Category</th>
<th>Grid Services</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 years to months ahead</td>
<td>Resource Adequacy</td>
<td>Resource Adequacy (Generic)</td>
</tr>
<tr>
<td></td>
<td>1–3 days</td>
<td>Energy Scheduling</td>
<td>Day-Ahead Energy Time-Shift</td>
</tr>
<tr>
<td>Service</td>
<td>Horizon</td>
<td>Planning/Operations</td>
<td>Real-Time Energy Time-Shift</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------</td>
<td>-----------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>Ancillary Services</td>
<td>Ahead, or annual contracts</td>
<td>Frequency Regulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Spinning Reserve</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-Spinning Reserve</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Frequency Response/Inertial Response</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flexible Ramping</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Black Start (annual contract)</td>
<td></td>
</tr>
<tr>
<td>Transmission</td>
<td>5–15 years ahead</td>
<td>Transmission Planning</td>
<td>Transmission Capacity Investment Deferral</td>
</tr>
<tr>
<td></td>
<td>Months ahead to real-time</td>
<td>Transmission Operations</td>
<td>Transmission Voltage Investment Deferral</td>
</tr>
<tr>
<td>Distribution</td>
<td>3–10 years ahead</td>
<td>Distribution Planning</td>
<td>Distribution Capacity Investment Deferral (Load Growth)</td>
</tr>
<tr>
<td></td>
<td>Day-ahead to real-time</td>
<td>Distribution Operations</td>
<td>Distribution Capacity Investment Deferral (N-1 Contingency)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Equipment Life Extension</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Distribution Losses Reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Conservation Voltage Reduction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Dynamic Voltage Control</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Backup Power/Microgrid</td>
</tr>
</tbody>
</table>

Each service has an objective for an asset in order to offer a quantifiable benefit with specified minimum criteria for success or value accrual relationships. There are also specific constraints that promote or inhibit the service.

An example source of constraints is the other types of services being offered, and whether a given service has priority over others. Ownership, control, and location of an asset affect the accessible services.

There are different ownership types (utility owned, customer owned, and third-party/IPP), which further affect the objective and constraints of each service. This is because ownership affects tax incentives and impacts, rebate programs, and business models.

Note that ownership is different from control. For instance, a system can be owned by a third party but operated by a utility or another entity serving as scheduling coordinators (in the CAISO market). This is summarized conceptually in Figure 2.
Scenarios

Scenarios are defined mainly as variations of location, technology, and services, but can also include variations of ownership, control, and financing. Certain combinations of locations, technologies, and services are excluded based on logical rules. These rules are to be determined and will be described further in the Functional Requirements Document and in the User’s Manual.

The scenario space is defined in terms of a limited number of scenario variables that have varying values. The number of such variables defines its dimension. Scenarios are represented as individual points in this space.

Report Overview

This report consists mainly of use case descriptions organized by primary actors: regulators, investors, or operators. The next chapter describes the structure of a use case.
CHAPTER 2:  
Use Case Overview

This section describes the major structure and components of a use case. Each use case has a general goal, and the use case structure identifies the actors, describes the information they exchange, and describes the process they go through to accomplish the goal.

The process has a starting condition, steps, and an end result for successful completion. If any part of the process is not successful, there is an exception, which may or may not be recoverable. If an exception has a successful recovery process, then the original process may resume and conclude successfully.

Actors

Actors are associated with roles and physically take the form of people and objects that interact. Actors within a use case are mutually exclusive. In the following descriptions, however, one may notice overlaps. For instance, stakeholders form a very broad group that largely includes most actors besides the Regulator.

Each use case has a Primary Actor — the one who has to make a decision that is supported through the use case process and by the other use case actors, as described in the process.

StorageVET

The common actor in these use cases is the StorageVET software application. Each process typically has one step to run Storage VET.

Regulator

StorageVET is imagined to be a tool used by a utility regulator.

Investors

StorageVET is imagined to be a tool used by an energy storage system (ESS) investor.

Stakeholders

Stakeholder actors are generally part of proceedings to approve storage projects.

Program Administrator

The program administrator is a specific role in a utility or regulatory commission tasked with running an incentive program for energy storage systems.

Participant

A participant receives incentives from an incentive program for building and operating an ESS. Participant behavior, in the form of operational and planning decisions, is affected by the design of the incentive program, and they give qualitative feedback on the design and implementation in terms of ease of understanding, fairness, and implementability.
Researcher
StorageVET is imagined to be a tool for ESS researchers. Researchers are involved in all aspects of ESS design, and they may use StorageVET to help represent the economic viability of a storage project or to help develop a detailed model of a specific storage project.

Market Designer
A market designer is a role that imagines and evaluates electricity market design features. StorageVET may be useful for helping to test the impact that storage may have on electricity markets.

Vendor
A vendor is the role of a person or organization that develops energy storage systems. Vendors may interact directly with a storage development program or indirectly through a utility.

Storage/DER Operator
An operator is a person or organization that operates an ESS or distributed energy resource (DER) device. Operators may use StorageVET to assist in operational decision-making.

Information Exchange
During the use case process, the actors exchange information to accomplish their tasks. As an example, one or more actors may prepare input files for StorageVET, which is run, and then StorageVET may provide its output files to one or more actors.

Within a Use Case subsection on Information Exchange, the exchanges are not necessarily listed in a sequential order. This was done for the convenience of concentrating the description.

Process
The Process subsection describes actions and information exchange by the actors in a logical order that leads to information that supports a decision by the primary actor. A use case process has three parts:

- **Starting Condition** – a set of assumptions and available information used to begin the process
- **Steps** – one or more steps that involve information processing and exchange between the actors
- **End Result** – the set of information used for decision support toward meeting the use case goal

Detailed Functional Process
For purposes of transitioning and linking this Use Case Document with the Functional Requirements Document, each process is elaborated in further detail to the functional level. This functional-level description is duplicated in the Functional Requirements Document, where it becomes the starting point for further analysis and documentation of StorageVET.
Exceptions

Any part of the process may lead to not having a successful completion along a primary process. In this event, the process may undergo a subprocess to handle the exception. If no such subprocess exists, then the use case process ends in failure. Otherwise, the subprocess leads the actor(s) through steps to resolve the unsuccessful part of the primary process and returns to the following step in order to proceed with the primary process.

Decision Support Model Structure

As a step toward formalizing the definition of a StorageVET Use Case, we include a semi formal description of the model structure that supports the decision being made by the principle actor in each use case. The structure has three parts: the decision variables that need to be determined, an objective that the principle actor has in mind during the decision making process, and major sources of uncertainty that impact the decision, represented by scenario variables (see below).

A use case decision support model considers StorageVET as a tool for solving a subproblem of the use case decision. The subproblem requires the processes, simulations, optimization, and heuristic features in StorageVET for one step of the use case process itself.

We assume that all StorageVET parameters have pre-defined ranges — limits on their values — in addition to type definitions, like binary, integer, real, etc.

Sets and Indexes

The parameters and variables require indexing for precise definition, and these indexes take values from defined sets. Both of these need to be described first.

All use cases will include the following sets and indices:

- Parameters – The set of all StorageVET parameter identifiers, indexed by \textit{param}.
- ScenarioIDs – The set of scenario identifiers, indexed by \textit{scenario}.

Decision Variables

The decision variables are decided by the principle actor, like whether to invest in a specific storage project, or how large its capacity should be. These variables are characterized through word descriptions, their numerical range, and their units of measure. Some decision variables may be qualitative, such as selecting a technology or a site for the project.

Decision variables may be treated similar to scenario variables — that is, the user may choose to vary them by using scenario features to explore the effects of alternative decisions on outcomes.

Scenario Variables

In addition to what the primary actor can decide, there are impactful decisions made by others or that are not controllable, like the weather. The values describing these impactful decisions are called scenario variables, and because they are impactful, it is important to account for how they
may vary. As mentioned, a user may choose to treat one of their decision variables as a scenario variable in order to explore the decision space.

This section of the model structure describes those parameters that are most likely to vary over the decision-making horizon and to affect the primary actor’s decision.

All use cases have sources of uncertainty formalized as:

- *ScenarioVariables* — Subset of *Parameters*, indexed by *scenarioVariable*. Indicates which parameters vary across the elements of *ScenarioIDs*. No unit of measure.
- *scenarioValue[scenarioVariable, scenario]* – The value that *scenarioVariable* takes for the given *scenario*. The range and units of measure are defined by the given *scenarioVariable*.

**Objective**

The decision support model objective is a formal version the user’s goal, which appears at the beginning of each use case. It is a quantified version, in terms of the decision variables and the major sources of uncertainty.

Many use cases involve sources of uncertainty and will need to have some statistical representation in the objective to help guide the user’s choice of decision variable values. This statistical representation depends on the preferences of the primary actor and has many variations that are part of the field of *Decision Making under Uncertainty*.  
CHAPTER 3: Regulator Use Cases

The Electric Power Research Institute (EPRI) imagines regulators as important users of StorageVET. The regulatory responsibility may benefit substantially from having a reference model, scenarios, and process for evaluating energy storage systems. Accordingly, the following use cases focus on the development and benchmarking of ESS models and scenarios and then on an ESS evaluation process that is facilitated by these reference components.

Use Case 1.1. Design Scenarios and Model Assumptions

The Regulator creates StorageVET scenarios for purposes of robust investment outcomes and reasonable operating strategies (incentives and projects).

Actors

The principal actor is the Regulator, interacting with Investors and StorageVET.

Regulator

The Regulator uses StorageVET to establish a full and complete set of data for evaluating ESS projects.

Investors

ESS investors have a keen interest in ensuring that the evaluation process is fair and that it can properly represent the types of ESS projects and operating conditions they have in mind.

StorageVET

The StorageVET software brings with it a full definition of the ESS simulation model and process, the necessary inputs needed to complete a valuation process, and the reporting needed to evaluate scenario results.

Information Exchange

Additionally, StorageVET needs to be configured to simulate the desired scenarios. Determining the scenario parameters requires the following communications:

Regulator <-> Investors

StorageVET input scenarios consist of national and local information and StorageVET configuration settings.

Regulator -> StorageVET

StorageVET inputs are fed in and then the output reports are returned.

Process

The following is an iterative process between the Regulator and the Investors to define a set of scenarios for valuation purposes. Each of these iterations is used to refine choices of the scenario variables and the values of these variables to evaluate.
**Starting Condition: Principal Scenario Variables**

Principal scenario variables are chosen mostly on the basis of their potential impact on the value of a future storage project. Examples of variables may involve alternative future economic conditions and fuel prices (as in the AEO 2015 [1]) and alternative future electricity market prices, regulatory policies, and technology evolution paths.

- `displayScenarioVariables()` – provide the user with a list of scenario variables for selection. Variables are selected from the set of data parameters.
- `selectScenarioVariables()` – the user selects scenario variables and then proceeds to identify ranges of values or specific values of each scenario variable.
- `defineScenarioComparison()` – the user defines for each scenario variable the threshold logic for comparisons. This logic is in terms of absolute or one-way differences between like inputs or outputs. This comparison logic is an attribute of the scenario variable.

**Step 1: Specify StorageVET Scenarios Sets**

Scenarios are assembled by the Regulator, in conjunction with Investors, into sets (a set of points in scenario space) that help define full StorageVET input files. More than one set of Scenarios can be defined, in order to define multiple, independent analyses of the same model.

- `displayScenarioSet()` – provide the user with a visualization of scenarios for all of its variables.
- `defineScenarioSet()` – the user chooses from among all combinations of the values of each scenario variable those specific points in this scenario space that constitute a set of Scenarios to be run by StorageVET. The user gives this set a unique identifier. Scenario Sets have creation and modification date attributes and version number attributes that are maintained automatically. Update Scenario Space display to show scenarios as they are added to the set.
- `identifyReferenceScenario()` – the user identifies from among all of the scenarios (subset of points in scenario space) one Reference Scenario, against which the other scenarios will be compared. The Reference Scenario is associated with the set identifier. Use special marking on the Reference Scenario in the Scenario Space display.
- `defineScenarioComparison()` – the user defines for each scenario dimension the threshold logic for comparisons. This logic is in terms of absolute or one-way differences between like inputs or outputs between the Reference Scenario and other Scenarios in the Scenario Set. Each Scenario Set can have different comparison logic, or can use default comparison logic, or can derive logic from the scenario variables. Scenario Set comparison logic is an attribute of the Scenario Set.

**Step 2: Run Scenarios**

StorageVET is configured to simulate the given scenarios.

- `defineScenarioSet()` – the user identifies which Scenario Sets to run.
- `inputModel()` – the user chooses a storage project model. This model is input and populates the model and data with default values.
- `createInput()` – create individual input datasets for each identified Scenario Set.
• **runScenarios()** – StorageVET is run on each Scenario input file and stores all output for comparisons.

**Step 3: Compare Scenarios**

StorageVET results are shared and discussed with the investors. StorageVET is expected to have features specially tuned for visualizing scenario comparisons. Tuning is based on practical considerations, like limited time and funding, and the extent to which each scenario variable impacts storage value.

The Regulator and Investors may iterate on the choices of scenario variables and the choices of the extreme points (high and low values) for these variables.

• **compareScenarios()** – use the Scenario Set comparison logic to flag significant differences in inputs and outputs between the Reference Scenario and other Scenarios in the Scenario Set.

• **displayScenarioComparison()** – display for the user the top-level inputs and outputs flagged by compareScenarios().

• **navigateScenarioComparison()** – the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences.

**End Result: Choices of Specific Scenarios**

The Regulator has a reference scenario and set of specific scenario variables with high and low points for each scenario variable and/or combinations of variables.

• **outputScenarioSet()** – the user selects a Scenario Set and StorageVET creates an output file that fully defines this Scenario Set.

• **inputScenarioSet()** – the user can choose to input one or more Scenario Sets and begin the process at an intermediate step.

**Exceptions**

StorageVET is unable to evaluate a desired scenario variable.

**Decision Support Model Structure**

The Regulator decides which scenarios to use for evaluating a wide range of potential future outcomes and uses for many different storage technologies. In essence, this use case is about choosing the most impactful sources of uncertainty and how to represent them.

In a certain way, sources of uncertainty in this use case are also part of the Regulator decision process in this use case. In other use cases, however, the sources of uncertainty are given and most likely have been defined by this use case.

**Sets and Indexes**

• **Parameters** – The set of all StorageVET parameter identifiers, indexed by param.

• **ScenarioIDs** – The set of scenario identifiers, indexed by scenario.

• **ScenarioVariables** – The set of scenario variable identifiers, indexed by scenarioVariable.

• **ResultingScenarioIDs** – The set of user-chosen scenario variables, indexed by resultingScenario. This set is initialized as part of the Starting Condition.
**Decision Variables**

- **DecisionVariables** – Subset of Parameters, indexed by `decisionVariable`. Indicates which parameters vary across the elements of `ScenarioIDs`. No unit of measure.
- `decisionValue[decisionVariable, resultingScenario]` – The value that `decisionVariable` takes for the given `resultingScenario`. The range and units of measure are defined by the given `param`.

**Scenario Variables**

The Scenario Variables are parameters that are not included in the `DecisionVariables` set, and they are likely the range of choices made by other actors that impact a decision in some other use case where the Regulator is the primary actor. For instance, they may express some range of technologies, services, and locations.

- **ScenarioVariables** — Subset of Parameters, indexed by `scenarioVariable`. Indicates which parameters vary across the elements of `ScenarioIDs`. No unit of measure.
- `scenarioValue[scenarioVariable, scenario]` – The value that `scenarioVariable` takes for the given `scenario`. The range and units of measure are defined by the given `scenarioVariable`.

Because this use case defines the sources of uncertainty for many of the other use cases, especially the Regulator ones, in later use cases one can assume that the following assignments have been executed:

- `ScenarioIDs := ResultingScenarioIDs;`
- `ScenarioVariables := DecisionVariables`
- `scenarioValue[scenarioVariable, resultingScenario] := decisionValue[scenarioVariable, resultingScenario];`

This means that other use cases start with the scenarios defined from this case. The user has still the option to append new scenario variables, perhaps as a means to explore the user’s decision space.

**Objective**

The Regulator’s objective is to identify from the principle scenario variables, given in the Starting Condition and which are formally defined in the `DecisionVariables` set, those that “most impact” Regulatory decisions.

Impacts can be formally defined in terms of the objectives of other Regulator use cases and can be evaluated by the Regulator for all given sources of uncertainty.

**Use Case 1.2. Compare Energy Storage Project Benefits, Costs, and Risks**

*The Regulator compares energy storage filings using StorageVET quantitative outputs, given default inputs and Consistent Evaluation Protocol (CEP) end-use choices.*

A Merchant Investor may wish to conduct a comparative analysis of utility contracts vs. merchant projects.
Actors
The principal actor is the Regulator, interacting with Investors and StorageVET to produce comparisons of alternative Storage Project proposals.

Regulator
The Regulator is conducting a comparison of Storage Project proposals in order to rank them and to make choices about approval for development and approval for cost recovery. The criteria for evaluating these proposals is specified in the California Public Utility Commission’s Consistent Evaluation Protocol (CEP) [3], but may include additional metrics, both quantitative and qualitative, and additional procedural descriptions that note how the valuation method in a proposal may differ from that in StorageVET.

Investors
Investors can be electric utility companies and merchant investors specializing in storage or distributed energy resources. They are proposing projects for regulatory approval and want to learn how their projects compare with the competition.

StorageVET
StorageVET is a vehicle for comparing proposals using a consistent set of scenarios, modeling, and reporting. It may be used directly for comparison or indirectly, relative to an undisclosed proprietary modeling and reporting system.

Information Exchange
Information is exchanged mostly between the Investors and the Regulator.

Investor -> Regulator
StorageVET Settings – Investors describe their proposed storage projects using default StorageVET settings and assumptions and further descriptions of any alternative settings.

CEP Filings – These filings further describe the ESS project, with details about the location, technology, and service capabilities.

Benchmarking – Differences from any proprietary method and the StorageVET results are described. See Use Case 1.4. Model/Data Benchmarking.

Regulator -> Investor
Valuation Scenarios – The Regulator has produced a set of scenarios in the form of StorageVET input files that are used for evaluating Storage Project proposals. See Use Case 1.1. Design Scenarios and Model Assumptions.

Project Rankings – The Regulator produces a comparison ranking for the Storage Project proposals.

Approval/Disapproval – The Regulator approves or disapproves each proposal for development and cost recovery.
Process
This process is once through, with Investors providing proposals and Regulators providing approval or disapproval.

Starting Condition
The Regulator is assumed to have default StorageVET inputs and a set of valuation scenarios. These scenarios may include elements of uncertainty about the future values of specific inputs, such as market prices or loads.

- inputModel() – the user chooses a storage project model. This model is input and populates the model and data with default values.
- inputScenarioSet() – the user chooses one or more Scenario Sets for evaluating the storage project.

Step 1: Regulator Receives Storage Project Proposals from Investors
Investors provide their proposals to the Regulator in the form of non-default StorageVET settings, CEP filings, and benchmarking.

The Regulator assumes that proposals complete the necessary input for StorageVET to evaluate the projects for all valuation scenarios.

There are no StorageVET functions involved in this step.

Step 2: Regulator Configures StorageVET to Accept Inputs from All Filings and Runs Simulations
The Regulator runs StorageVET for all valuation scenarios and Investor proposal information. StorageVET may have the ability to automatically run a large set of Storage Projects across all scenarios.

- displayScenarioSet() – provide the user with a list of storage projects for selection. Projects are selected from the set of that have been submitted by Investors.
- inputScenarioSet() – the user chooses one or more storage project Projects to input. The Storage Project Set identifies new values for data parameters.
- checkForParameterConflicts() – checks the list of scenario parameters and the list of storage project parameters for conflicts or overrides and reports on significant changes.
- createInput() – create individual input datasets for each Storage Project for all given Scenario Sets. This involves a cross product of the Storage Project set and the Scenario set.
- runScenarios() – StorageVET simulates each Storage Project for all given Scenario Sets.

Step 3: StorageVET Presents Quantitative Output in CEP Format, Plus Other Reporting
Using StorageVET, the Regulator produces reports that compare the CEP results and the results of all scenarios. The Regulator may conduct extensive reviews of these quantitative results along with qualitative and benchmark information to further rank Projects.

- compareScenarios() – use the Scenario Set comparison logic to flag significant differences in inputs and outputs between the Storage Projects.
- `displayScenarioComparison()` – display for the user the top-level statistics about inputs and outputs flagged by `compareScenarios()` across the Scenario set. This includes major project scoring metrics as determined by the Regulator.
- `navigateScenarioComparison()` – the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences.

**End Result: Regulator Reports Project Comparison to Utility Investors**

The StorageVET results are provided to the Investors, along with their decision of approval or disapproval.

- `outputScenarioComparison()` – StorageVET creates an output file that reports the Storage Project Comparison across the Scenario set.
- `inputScenarioComparison()` – StorageVET inputs a file that contains the Storage Project Comparison across the Scenario set. This function can be used to restart this use case at an intermediate step.

**Exceptions**

A proposal has invalid or incomplete input for StorageVET.

**Decision Support Model Structure**

The Regulator decides which storage project Projects to support.

**Sets and Indexes**

- `Parameters` – The set of all StorageVET parameter identifiers, indexed by `param`.
- `ScenarioIDs` – The set of scenario identifiers, indexed by `scenario`.
- `Projects` — The set of all storage project proposal identifiers submitted to the Regulator for approval, indexed by `project`.

**Decision Variables**

- `approved(project)` — binary-valued indicator that the Regulator approves the given `project`.

**Scenario Variables**

The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:

- `ScenarioVariables` — Subset of `Parameters`, indexed by `scenarioVariable`. Indicates which parameters vary across the elements of `ScenarioIDs`. No unit of measure.
- `scenarioValue[scenarioVariable, scenario]` – The value that `scenarioVariable` takes for the given `scenario`. The range and units of measure are defined by the given `scenarioVariable`.

**Objective**

The Regulator’s objective is complex, but is known to include several formal quantifications in terms of the CEP reporting requirements, like the Ratepayer Impact Measure and the Total Resource Cost Test.
Use Case 1.3. Design Incentive Program

The Regulator designs parameters for a given incentive program framework.

Actors
The principal actor is the Regulator, interacting with Stakeholders, Program Participants, the Program Administrator, and StorageVET.

Regulator
The Regulator has ranked Projects that pass the Ratepayer Impact Measure (RIM) test but do not pass the Net Benefit Test (NBT), because they have higher costs than benefits, and the Regulator wants to explore a variety of incentives to encourage investors to proceed with their projects.

Stakeholders
Stakeholders want to ensure that their constituents are treated fairly and are interested in the reporting results under the variety of incentive program designs.

Program Administrator
The Program Administrator wants to ensure that the incentive program design is clear and implementable.

Program Participant
Program Participants want to ensure that the incentive program is clear and that they understand the impact it may have on their proposed projects.

StorageVET
StorageVET is used as a tool for evaluating scenarios that are based on alternative incentive program designs.

Information Exchange
Stakeholders <-> Regulator
Together, the Stakeholders and the Regulator work on designs for incentive programs. This is called an incentive program framework. The Stakeholders and the Regulator engage in an iterative process to design a set of scenarios, which are in the form of StorageVET input files, prior to the execution of this use case. Such an analysis to determine an incentive program is typically defined in terms of some threshold of policy achievement. This could be in the form of a benchmark technology or one of the ranked projects, with which other storage projects are intended to achieve parity.

Regulator -> Stakeholders
The Regulator returns to the Stakeholders the incentive parameters, which could be in one or more forms, like a tax credit or cash back.
Process
This incentive design process deeply involves a stakeholder process to define a policy goal, like integration of renewable energy production, the target level of energy storage seen to support that goal, and one or more potential incentive mechanisms for testing. StorageVET is used to assess the relative effectiveness of the mechanisms.

Starting Conditions
Incentive Program Framework – This framework is defined by the specification of one or more parameters in StorageVET that has the effect of changing either the operational decisions or the investment decisions on specific ESS projects. It could take the form of a feed-in tariff, tax rebate or reduction, or direct subsidy.

Targets – The Regulator sets policy targets for the quantity of ESS to approve during a round of project evaluations.

Scenarios – ESS projects should be economically robust, according to the various performance tests, across a number of reasonable future alternative scenarios (see Use Case 1.1. Design Scenarios and Model Assumptions).

Benchmark Technology – This represents a technology for comparison purposes, and it may be one of a set of given and ranked ESS Projects. Projects that perform better than this technology could be promising candidates for approval.

• inputModel() – the user chooses a storage project model. This model is input and populates the model and data with default values.
• inputScenarioSet() – the user chooses one or more Scenario Sets for evaluating incentive programs.
• displayScenarioSet() – provide the user with a visualization of incentive parameter space. Each dimension of this space represents a potential choice for an incentive scenario variable.
• selectScenarioVariables() – the user selects incentive parameters for the incentive scenario variables and then proceeds to identify ranges of values or specific values of each scenario variable. Incentive scenario variables have unique identifiers. Update the Incentive Scenario Space display while it is being selected and further defined by the user.
• defineScenarioComparison() – the user defines for each incentive scenario variable the threshold logic for comparisons. This logic is in terms of absolute or one-way differences between like inputs or outputs. This comparison logic is an attribute of the incentive scenario variable.
• inputScenarioSet() – the user chooses one or more storage project Projects for evaluating incentive programs. The Storage Project Set identifies new values for data parameters. The Storage Project Set also contains benchmark technology projects.
• checkForParameterConflicts() – checks the lists of incentive scenario parameters, scenario parameters, and Storage Project parameters for conflicts or overrides and reports on significant changes.
- `displayScenarioSet()` – provide the user with a visualization of the incentive scenarios for all of its scenario variables, including options to view other scenario variables, like those representing sources of uncertainty.
- `defineScenarioSet()` – the user chooses from among all combinations of the values for each incentive scenario variable those specific points in this incentive scenario space that constitute a set of Scenarios to be run by StorageVET. The user gives this set a unique identifier. Incentive Scenario Sets have creation and modification date attributes and version number attributes that are maintained automatically.
- `defineScenarioComparison()` – the user defines for each incentive scenario variable the threshold logic for comparisons of storage projects to Targets. This logic is in terms of absolute or one-way differences between Targets and the inputs and/or outputs of Incentive Scenarios in the Scenario Set. Target Comparison Logic is an attribute of Scenario Sets.

**Step 1: Construct StorageVET Input Files for a Range of Incentives**

The Regulator constructs StorageVET input files using proposed projects, the benchmark technology, scenarios, and the possible incentive program frameworks.

- `createInput()` – create individual input datasets for all combinations of given Incentive Scenarios, Storage Projects, and Scenario Sets.

**Step 2: Run StorageVET**

The Regulator runs StorageVET for selected combinations of the range of input files, and StorageVET produces comparison outputs that rank the proposed projects according to various criteria used as flexible parameters to define the input files, like scenarios, projects, and ranges of incentive parameter values, especially in terms of the targets.

- `runScenarios()` – StorageVET simulates each Storage Project for all given incentive variables, Storage Projects, and other scenario variables in the Scenario Sets.

**Step 3: Determine Incentive Level(s) That Meet Targets**

The Regulator determines which projects perform above the given targets.

- `compareScenarios()` – use the Incentive Scenario Set comparison logic to flag significant differences in inputs and outputs between the Targets and Incentive Programs.
- `displayScenarioComparison()` – display for the user the top-level inputs and outputs flagged by `compareScenarios()`.
- `navigateScenarioComparison()` – the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences.

**End Result: A Collection of Incentives That Meet Targets**

The Regulator has a collection of Storage Project proposals that are expected to outperform the targets across all scenarios within the incentive program framework.

- `outputScenarioComparison()` – StorageVET creates an output file that reports the Storage Project Comparison of incentive programs.
inputScenarioComparison() – the StorageVET inputs a file that contains the Storage Project Comparison. This function can be used to restart this use case an intermediate step.

Exceptions
No incentive level within the given range is able to bring a proposed ESS project above the target.

Decision Support Model Structure
The Regulator decides on which types of incentive programs to support and what parameter values can support their policy targets.

Sets and Indexes
- Parameters – The set of all StorageVET parameter identifiers, indexed by param.
- ScenarioIDs – The set of scenario identifiers, indexed by scenario.
- IncentiveTypes – The set identifiers of types of incentive programs, indexed by incentive.

Decision Variables
- IncentiveDecisions – Subset of IncentiveTypes, indexed by incentiveDecision. Indicates the Regulators choice of which types of incentive programs to support. No unit of measure.
- incentiveValue[incentiveDecision, param] – The value that param takes for the given incentiveDecision. The range and units of measure are defined by the given param.

Scenario Variables
The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:
- ScenarioVariables – Subset of Parameters, indexed by scenarioVariable. Indicates which parameters vary across the elements of ScenarioIDs. No unit of measure.
- scenarioValue[scenarioVariable, scenario] – The value that scenarioVariable takes for the given scenario. The range and units of measure are defined by the given scenarioVariable.

Objective
The Regulator’s objective is to reach pre-defined policy targets for the development of new storage projects. These targets may be quantified by capacity and duration of likely projects for each location. There may be additional variables for the targets.

The incentiveValue is chosen to meet or exceed the threshold targets subject to passing a variety of Regulator tests, like the Ratepayer Impact Measure and the Total Resource Cost Test.

Use Case 1.4. Model/Data Benchmarking

A principal actor wants to explain differences in simulation results between StorageVET and an alternative reporting framework, or simulation model.

StorageVET is used as a tool for comparing the results of two models of ESS performance: one model is treated as a benchmark, and the other is StorageVET. The challenge is determining the differences between the results of the two models, without disclosing confidential information.
The inputs to the two models are assumed to be consistent and available in StorageVET format. If there are any differences between the inputs, they should be noted and explained for their potential impact on the outputs.

**Actors**
The alternative principal actors are the Regulator, Investors, Researchers, and a Market Designer.

**Regulator**
A Regulator would like to benchmark StorageVET input data with observed data. Observed data is historical and may be used to tune the StorageVET modeling.

**Investors**
An Investor has an alternative model that is preferred over StorageVET for some reason and wants to compare the two modeling systems.

**Researcher**
A Researcher is developing an alternative model that is preferred over StorageVET for some reason and wants to compare the two modeling systems.

**Market Designer**
A Market Designer wants to calibrate StorageVET to a historical period of performance in order to develop a reference case for purposes of comparing alternative market products and services.

**StorageVET**
StorageVET is being used to compare the outputs from its own alternative input scenarios and from alternative modeling systems in order to determine the source of differences in the outputs.

**Information Exchange**

*Principal Actor -> StorageVET*

*Alternative Model Output* – Output from a model for valuing storage that is used as an alternative to StorageVET. Its output is in StorageVET format for easy comparison.

*Wholesale Market Data* – Actual wholesale market settlement data that is used as a reference for calibrating StorageVET data and modeling. Ideally this data is in a format compatible with StorageVET.

*Scenarios* – The set of scenarios over which the StorageVET model and other input data will be benchmarked.

*StorageVET Input* – A complete collection of data in StorageVET format, consisting of scenarios and other data.
StorageVET -> Principal Actor

StorageVET produces model comparison metrics from given output data, across alternative scenarios. These metrics can be used to trace down the sources of differences between the outputs.

Process

This process involves running StorageVET simulations and output comparisons in order to trace down the sources of differences in the output.

Starting Condition

StorageVET is given alternative model output, wholesale market data, scenarios, and StorageVET input for evaluation and comparison. Output of an alternative reporting framework, simulation model, or market structure is produced on the basis of the given scenarios.

- `inputModel()` – the user chooses a storage project model. This model is input and populates the model and data with default values.
- `inputScenarioSet()` – the user chooses one or more Scenario Sets for benchmarking the storage model. The user may wish to adjust the Scenario Comparison Logic for each chosen Scenario Set to best suit the benchmarking analysis. One approach is to begin with large tolerances and to tighten them down in subsequent runs.
- `displayScenarioSet()` – provide the user with a list of Storage Projects for selection. Projects represent the analysis of a storage project with both StorageVET and a proprietary model.
- `inputScenarioSet()` – the user chooses at least two Storage Project proposals to input one from a StorageVET analysis and one from a proprietary analysis. Each Storage Project Set defines a comparison of a single storage project over all Scenario Sets.
- `checkForParameterConflicts()` – checks the list of scenario parameters and the list of Storage Project parameters for conflicts or overrides and reports on significant changes.
- `createInput()` – create individual input datasets for each Storage Project for all given Scenario Sets.

Step 1: Run StorageVET for All Storage Projects and Scenarios

This step produces StorageVET output that can be compared with the output of the alternative model.

- `runScenarios()` – StorageVET simulates each Storage Project Set for all given Scenario Sets.

Step 2: Identify Differences or Changes in Benefits, Costs, and Risks Between StorageVET and Alternative System

StorageVET compares pairs of output files and identifies differences based on preconfigured threshold criteria. These threshold criteria can be simple bounds on differences between analogous output values or logical combinations of output values.

- `compareScenarios()` – use the Scenario Set comparison logic to flag significant differences in inputs and outputs between the benchmark (StorageVET) project and the proprietary analysis in each Storage Project Set.
Step 3: Identify Sources of These Differences

We assume that any significant differences between the inputs of the two modeling systems are identified and well understood.

Knowing the StorageVET model formulation and simulation process and knowing the differences in outputs between StorageVET and the alternative model, the principal actor uses patterns in these differences to attribute to significant differences in the inputs, simulation models, or processing.

- \textit{displayScenarioComparison}() – display for the user the top-level inputs and outputs flagged by \textit{compareScenarios}(). This includes major project scoring metrics as determined by the Regulator.
- \textit{navigateScenarioComparison}() – the user selects input or output comparisons and drills into the upstream calculations to investigate the sources of the differences. The user may wish to tighten down certain Scenario Comparison Logic during the navigation of the comparison, in order to resolve unexplained differences.

End Result: Explain Differences or Changes

The principal actor now explains why the differences exist and why the alternative results are preferable to those from StorageVET.

- \textit{outputScenarioComparison}() – StorageVET creates an output file that reports the Storage Project Comparison.
- \textit{inputScenarioComparison}() – StorageVET inputs a file that contains the Storage Project Comparison. This function can be used to restart this use case at an intermediate step.

Exceptions

A difference between the alternative model outputs and the analogous StorageVET outputs cannot be explained.

Decision Support Model Structure

The Regulator detects differences between the inputs and outputs of two different storage project models and decides which ones are significant, if any.

Sets and Indexes

- \textit{Parameters} – The set of all StorageVET parameter identifiers, indexed by \textit{param}.
- \textit{Variables} – The set of all StorageVET decision variables, indexed by \textit{variable}.
- \textit{Identifiers} – The union of \textit{Parameters} and \textit{Variables}, indexed by \textit{id}.
- \textit{ScenarioIDs} – The set of scenario identifiers, indexed by \textit{scenario}.
- \textit{Models} – The set of models being compared, indexed by \textit{model1} and \textit{model2}. This set includes StorageVET as the benchmark model.

Decision Variables

\textit{thresholdDifferences}[id, \textit{scenario}] – Values that the principle actor decides represent a significant difference when observed between models. Units of measure are defined as those of \textit{id}. The value of this decision variable may be given in the Starting Condition.
significantDifference[id, model1, model2 scenario] – A binary-valued indicator that parameter or variable id differs significantly between model1 and model2 for the given scenario.

**Scenario Variables**

The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:

- **ScenarioVariables** — Subset of Parameters, indexed by scenarioVariable. Indicates which parameters vary across the elements of ScenarioIDs. No unit of measure.
- **scenarioValue[scenarioVariable, scenario]** – The value that scenarioVariable takes for the given scenario. The range and units of measure are defined by the given scenarioVariable.

**Objective**

The Regulator’s objective is subjective in this use case. The root of their decision-making is specifying thresholdDifferences, which may take place as part of the use case process or given as part of the Starting Condition through some sort of external process.

If thresholdDifferences is given in the Starting Condition, then there is no objective. Running StorageVET merely computes and reports on the value of significantDifference.

**Use Case 1.5. Define Benchmark Model for Improving CEP**

The Regulator ensures that the CEP definition is complete, transparent, readily available, well defined, and fit for the purpose.

The Consistent Evaluation Protocol (CEP) [3] contains a set of quantitative input values and blank outputs to be completed by the entity responding to an RFP. CEP information also includes qualitative information that is beyond the scope of this use case. Examples of the categories of CEP information are proposer identification, project duration, efficiency, O&M, and technology.

StorageVET is an open, transparent model that is capable of taking these input values and producing the required output values, and it may include more information than is in the current CEP. For instance, there may be additional important inputs, and StorageVET also contains configurable modeling equations, objective functions, and processes for simulating ESS projects. This complete set of information is necessary for producing well-specified output and may act as a supplement to the CEP.

**Actors**

The principal actor is the Regulator.

**Regulator**

The Regulator is interested in having a set of ESS performance data that is well specified and comparable across a variety of proposed projects. The CEP acts as the vehicle to achieve this, and StorageVET may provide support for further achieving this goal.
**Stakeholders**
Stakeholders work with the Regulator to improve the CEP in a public forum in which the information specified in the CEP is compared with the StorageVET data model, simulation model, and ESS performance reporting.

**StorageVET**
StorageVET is used as a public reference for comparison with the CEP. It serves as a medium for communicating concepts and quantitative information about data, modeling, and analytical processes.

**Information Exchange**
The consistent data is represented in the CEP and StorageVET. It is exchanged seamlessly for the purposes of comparing these two frameworks.

**Regulator <-> StorageVET**

*CEP Scenarios* – These scenarios are predefined by the Regulator and used as a collection of inputs for consistent evaluation of Storage Projects. They are used by the Regulator to run StorageVET.

*StorageVET Outputs and Comparison Metrics* – Outputs and comparison metrics are produced by StorageVET based on the scenarios and predefined comparison processes.

**Regulator <-> Stakeholders**
Regulators and Stakeholders work in a forum to define scenarios (see Use Case 1.1. Design Scenarios and Model Assumptions) and discuss the definition and completeness of the CEP in comparison to StorageVET.

*CEP Scenarios* – Should Stakeholders choose to do so, they can use the CEP scenarios as input for their own alternative analysis.

*StorageVET Outputs and Comparison Metrics* – Regulators and Investors review the StorageVET results and decide together whether any differences between the scenarios are significant and not yet represented properly in the CEP. If so, then the CEP is updated to include these significant input values and/or output placeholders.

**Process**
The Stakeholder forum takes place in advance of this use case.

**Starting Condition: CEP Input Scenarios and Reporting Structure**
Assume StorageVET is accurate. Scenarios are available.

**Step 1: Verify Correspondence Between CEP and StorageVET Features and Assumptions**
Stakeholders review the assumptions supporting the CEP and StorageVET and determine whether they are sufficiently consistent or need to be revised in some way.
Step 2: Verify Completeness of CEP Inputs
The CEP input assumptions and data definitions are reviewed to ensure that any ambiguity does not significantly impact ESS performance across the given scenarios.

Step 3: Document and Verify Input Needed for Given CEP Outputs
Trace the modeling equations, objectives, and decision processes in StorageVET to ensure that sufficient assumptions and input data are available to produce the desired output.

End Result: I/O Influence Diagram and Documentation of the Technical Model and Reporting
The Stakeholder process produces an influence diagram describing the needed inputs and processes for producing desired output.

Exceptions
CEP or StorageVET misses assumptions or has significantly different assumptions.

CEP or StorageVET lacks some inputs.

Decision Support Model Structure
The Regulator decides whether the quantitative aspects of the CEP require further definition.

Sets and Indexes
- Parameters – The set of all StorageVET parameter identifiers, indexed by param.
- ScenarioIDs – The set of scenario identifiers, indexed by scenario.
- StartingCEPParameters – A subset of Parameters that identifies those that correspond directly to given CEP parameters.

Decision Variables
- EndingCEPParameters – A subset of Parameters that identifies those that the Regulator would like to include in the next version of the CEP.

Scenario Variables
The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:
- ScenarioVariables — Subset of Parameters, indexed by scenarioVariable. Indicates which parameters vary across the elements of ScenarioIDs. No unit of measure.
- scenarioValue[scenarioVariable, scenario] – The value that scenarioVariable takes for the given scenario. The range and units of measure are defined by the given scenarioVariable.

Objective
The Regulator’s objective is to define a sufficient set of parameters for the CEP that “most impact” Regulatory decisions.

Impacts can be formally defined in terms of the objectives of other Regulator use cases and can be evaluated by the Regulator for all given sources of uncertainty.
CHAPTER 4: 
Investor Use Cases

This chapter treats the Investor role in general, where an Investor can be a merchant organization or a regulated utility. The goals of these organizations may differ; however, the use of StorageVET is very similar in each use case. The main differences between entities is their primary performance goal, and StorageVET produces performance reports from all relevant perspectives as a means for being useful to all Stakeholders.

Use Case 2.1. Screen Feasible Locations

An Investor has a given planning area and wants to create a short list of specific locations for deeper analysis.

This use case may be implemented through a database query using threshold characteristics for load shapes, load growth, energy price, marginal capacity cost, available capacity, frequency, and energy requirement/duration. Screening through engineering judgment may also be used as a course filter/criterion.

Benchmark technologies can be a CT, demand response, a diesel generator, or one or more default storage technologies.

Actors

The principal actor is the Investor.

Investor

The Investor has a list of potential locations for siting storage projects and would like to rank them according to the likelihood that an ESS solution will be economical.

StorageVET

StorageVET is run in a batch mode to test a limited number of technologies and scenarios across many locations. The results are presented in a comparison view suited for investment decisions.

Information Exchange

The scenarios and technologies used for screening locations are subsets of available datasets, chosen for quick evaluation.

Regulator and/or Analyst -> Investor

Scenarios – The Regulator specifies a collection of scenarios for evaluating and comparing alternative Storage Projects from multiple investors (see Use Case 1.1. Design Scenarios and Model Assumptions). A subset of these is used for screening purposes, with emphasis on extreme scenarios and fast evaluation.
Regulator, System Operator, or Utility -> Investor

Feasible Locations – A list of available locations for installation of new or upgraded storage projects. These locations may have technology dependencies and could number in the hundreds.

StorageVET -> Investor

Benchmark Technologies – StorageVET is expected to include supply-side and demand-side technologies against which storage projects are compared. They become important for this use case.

Process

The process is generally to set up StorageVET to run a large batch of analyses that are combinations of benchmark technologies, scenarios, and a large number of locations. Then the comparison functions in StorageVET are used to compare prospective locations from different viewpoints.

Starting Condition: Feasible Locations, Scenarios, and Benchmark Technologies

We assume that the Investor is provided with all necessary inputs to fully specify the combinations of analyses necessary for screening a large number of locations.

- `inputModel()` – the user chooses a storage project model. This model is input and populates the model and data with default values.
- `inputScenarioSet()` – the user chooses one or more Scenario Sets for evaluating the storage model. The user may wish to adjust the Scenario Comparison Logic for each chosen Scenario Set to best suit the benchmarking analysis. One approach is to begin with large tolerances and to tighten them down in subsequent runs.
- `displayScenarioSet()` – provide the user with a list of technologies for selection.
- `defineScenarioSet()` – the user chooses a limited number of benchmark technologies to evaluate. It is expected that StorageVET will include default generic technologies for this type of evaluation. The user-selected benchmark technologies define a Technology Set that will be used for comparing locations over all Scenario Sets.
- `outputScenarioSet()` – allows the user to output a defined Technology Set.
- `inputScenarioSet()` – allows the user to input a defined Technology Set, in order to initialize this process.
- `displayScenarioSet()` – provide the user with a list of locations for selection.
- `defineScenarioSet()` – the user chooses multiple locations to evaluate, and they become elements of a Location Set that will be used to compare the locations over all Scenario Sets and benchmark technologies.
- `defineScenarioComparison()` – the user defines for each benchmark technology the metrics for comparisons of locations. These metrics can be standard performance tests, like those defined in the CEP, or user definitions. Location Comparison Logic is an attribute of Location Set.
- `outputScenarioSet()` – allows the user to output a defined Location Set.
• `inputScenarioSet()` – allows the user to input a defined Location Set, in order to initialize this process.
• `checkForParameterConflicts()` – checks the scenario parameters, locations, and technologies for conflicts or overrides and reports on significant changes.
• `createInput()` – create individual input datasets for each location for all given Scenario Sets and the Technology Set.

**Step 1: Choose Primary Service(s) for Screening**

The Investor chooses a primary service for screening purposes. This primary service is specifically used to determine the viability of all locations. Secondary services may or may not be run, depending on the time requirement for computing all of the full screening.

• `displayScenarioSet()` – provide the user with a visualization of the storage services that are compatible with those in the Technology Set.
• `defineScenarioSet()` – the user chooses from among all services those to include in this analysis. One service in particular is labeled as the primary service. The user-selected services with the labeled primary services define the Services Set for a given technology.
• `outputScenarioSet()` – allows the user to output a defined Services Set.
• `inputScenarioSet()` – allows the user to input a defined Services Set, in order to initialize this process.

**Step 2: Run StorageVET**

Screen the specific locations with heuristic thresholds that indicate high potential for ESS benefits from the primary service.

• `runScenarios()` – StorageVET simulates each Location for all given benchmark technologies and Scenario Set.

**Step 3: Rank Locations Having Economic Potential**

Use StorageVET to report out the location rankings according to various performance tests.

• `compareScenarios()` – use the Location Set comparison logic to sort location results in the order of a user-chosen performance metric.
• `displayScenarioComparison()` – display for the user the top-level locations flagged by `compareLocations()`.
• `navigateScenarioComparison()` – for a given performance metric ranking, the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences between the locations and technologies, across all scenarios.

**End Result: High Ranked Feasible Locations for Deeper Analysis**

The Investor is provided with a limited set of locations, perhaps less than 20, that have high potential for the benchmark technology providing the given primary service.

• `outputScenarioSet()` – allows the user to output a defined Location Set.
• `outputScenarioComparison()` – StorageVET creates an output file that reports the Location Comparison of benchmark technologies for all scenarios.
• *inputScenarioComparison()* – StorageVET inputs a Location Comparison of benchmark technologies for all scenarios in order to modify or explore it.

**Exceptions**
The primary service cannot be screened.

**Decision Support Model Structure**
The Investor decides on which locations are most viable for supporting a storage project. This use case is a preliminary for Use Case 2.2. Size a Project by Location, Primary Service, and Technology, which assumes that the location is given.

**Sets and Indexes**
- **Parameters** – The set of all StorageVET parameter identifiers, indexed by *param*.
- **ScenarioIDs** – The set of scenario identifiers, indexed by *scenario*.
- **Locations** – The set of locations for prospective storage projects, indexed by *location*.
- **Technologies** – The set of candidate technologies to test at each location, indexed by *tech*.
- **Services** – The set of candidate services to test at each location, indexed by *service*.

**Decision Variables**
- **PrimaryService*[location, technology]*** – The primary service to test at each *location* for the given technology. This is a user-configurable option for this use case.
- **ViableLocations*[location, technology]*** – A binary valued indicator of a viable location for a storage project.

**Scenario Variables**
The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:

- **ScenarioVariables** — Subset of *Parameters*, indexed by *scenarioVariable*. Indicates which parameters vary across the elements of *ScenarioIDs*. No unit of measure.
- **scenarioValue*[scenarioVariable, scenario]*** – The value that *scenarioVariable* takes for the given *scenario*. The range and units of measure are defined by the given *scenarioVariable*.

These may be supplemented by additional preferences of the Investor.

**Objective**
The Investor objective is to rank the potential locations according to their own metric, which may be the Total Resource Cost Test.

**Constraints**
The Investor is subject to the approvals of the Regulator, which requires that projects meet minimum threshold requirements on the various perspective tests. For instance, the Ratepayer Impact Measure needs to be positive. Violation of a constraint for a *[location, technology]* pair excludes from *ViableLocations*. 
Use Case 2.2. Size a Project by Location, Primary Service, and Technology

An Investor wants to design an ESS project that is expected to be most valuable when given the location and a specific technology.

We assume that StorageVET can automatically choose the best services. The user specifies yes/maybe/no for each technology.

StorageVET will have an automated method to determine alternative designs and to test each one for over- or under-build. Further, there is expected to be a filtering for designs (sizes) that are close to “optimal.”

Primary services include T&D deferral.

Actors
The principal actor is the Investor or Vendor.

Investor
The Investor has a limited number of locations for deep analysis of ESS potential. These sites have been screened for high potential value and now need details about the technology and its power output and storage duration, among other parameters.

Vendor
The Vendor sells a specific technology and is interested in determining the best configurations of power and duration to offer for sale. A limited number of high potential value locations are targeted as representing the market for their products.

StorageVET
StorageVET is used to search the power and duration space for the optimal parameter values providing the highest value services across all scenarios for a given location.

Information Exchange
The Investor or Vendor collects information for evaluating ESS technology sizes and ultimately reports back to the Regulator the most promising projects.

Regulator -> Investor
Regulator -> Investor -> Vendor

Procurement Cycle Guidelines – There are multiple procurement cycles occurring on a bi-annual basis.

Minimum Performance Requirements – Potentially viable storage projects should meet or exceed minimum technical performance requirements in order to pass this screening process.

Prudence Criteria – Potentially viable projects should meet or exceed non-technical performance criteria associated with non-technical aspects of the project.
Scenarios – The Regulator provides a collection of scenarios for evaluation and comparison of storage projects.

CEP – The Consistent Evaluation Protocol (CEP) is the CPUC standard evaluation framework [3].

Investor -> Regulator
Vendor -> Investor -> Regulator

CEP Output Values – Required entries in the CEP are determined from running StorageVET and/or other evaluation methodologies.

Prudence Values – These consist of nontechnical values or descriptions supporting regulatory approval.

Process
After data collection, the primary actor runs StorageVET and decides which projects to propose to the Regulator.

Starting Condition: Location, Primary Service, Technologies, Procurement Cycle Guidelines
A location, a primary service, and an optional technology are specified, in accordance with given procurement cycle guidelines.

- inputModel() – the user chooses a storage project model. This model is input and populates the model and data with default values.
- inputScenarioSet() – the user chooses one or more Scenario Sets for evaluating incentive programs.
- inputTechnologySet() – allows the user to input a defined Technology Set, in order to initialize this process.
- inputLocationSet() – allows the user to input a defined Location Set, in order to initialize this process.
- checkForParameterConflicts() – checks the scenario parameters, locations, and technologies for conflicts or overrides and reports on significant changes.
- createInput() – create individual input datasets for each location for all given Scenario Sets and the Technology Set.

Step 1: Assemble StorageVET Input Files
The Investor or Vendor collects all StorageVET inputs for a given location.

- displayScenarioSet() – provide the user with a list of technologies for selection.
- displayTechnology() – provide the user with a display of the parameters for a selected technology, and allow the user to customize the parameter settings. It is expected that StorageVET will include default generic technologies for the user to customize.
- displayScenarioVariables() – provide the user with a list of technology sizing variables for selection. Variables are selected from the set of data parameters of the given technology according to the power and energy capacity of the device, and any other significant sizing parameters.
- `selectScenarioVariables()` – the user selects sizing variables and then proceeds to identify ranges of values or specific values of each sizing variable.
- `defineScenarioSet()` – the user chooses from among all combinations of the values of each sizing variable those specific points in this scenario space that constitute a Sizing Set to be run by StorageVET. The user gives this set a unique identifier. The Sizing Set is an attribute of the technology.
- `defineScenarioComparison()` – the user defines the metrics for comparing sizing scenarios. These metrics can be standard performance tests, like those defined in the CEP, or user definitions. Sizing Comparison Logic is an attribute of the Sizing Set.
- `outputScenarioSet()` – allows the user to output a defined Technology Set, with one element, for later use.
- `defineScenarioSet()` – the user chooses a limited number of custom technologies to evaluate. The user-selected benchmark technologies define a Technology Set that will be used for comparing locations over all Scenario Sets.
- `outputScenarioSet()` – allows the user to output a defined Technology Set.

**Step 2: Run StorageVET**

StorageVET runs a searching study for the best combination of ESS project power and duration for the given location and a given scenario. All scenarios are run independently and services are optimized automatically within the ESS dispatch. The result is a comparison of ESS sizes for evaluation.

The Investor or Vendor selects from the sizing options the one they think performs best across the variety of scenarios.

- `displayScenarioSet()` – provide the user with a list of technologies for selection.
- `defineScenarioSet()` – allow the user to select one technology scenario to evaluate for sizing.
- `runScenarios()` – StorageVET simulates each sizing scenario of the selected technology over the Scenario Set.
- `compareScenarios()` – use the Sizing Set comparison logic to flag significant differences in inputs and outputs between the different sizing scenarios computing summary statistics of the performance metrics across the scenarios in the Scenario Set.
- `displayScenarioComparison()` – display for the user the top-level sizing scenarios flagged by `compareScenarios()`.
- `navigateScenarioComparison()` – the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences between sizing scenarios.
- `modifyScenarioSet()` – allow the user to redefine the elements of the Sizing Set in order to further resolve the size of the storage project. This involves displaying the region of the sizing space having the best performance metrics, and allowing the user to zoom in on that region, if possible, to determine the project size with greater accuracy.
Note that this step is iterative, having the user define a broad range of sizing scenarios and resolve them into a narrow range having the best performance. This can be repeated for a number of technologies.

**Step 3: Create CEP and Other Outputs**

StorageVET automatically generates CEP and other output for the selected ESS project size at the given location.

- **outputScenarioSet()** – the user selects a Sizing Set and StorageVET creates an output file that fully defines this Sizing Set.
- **inputScenarioSet()** – the user can choose to input one or more Sizing Sets and begin the process at an intermediate step.
- **outputScenarioComparison()** – StorageVET creates an output file that reports the Sizing Scenario Comparison.
- **inputScenarioComparison()** – StorageVET inputs a file that contains the Sizing Scenario Comparison. This function can be used to restart this use case at an intermediate step.
- **outputCEP()** – StorageVET creates a standard CEP output file that reports the location, technology, and size of the best Storage Projects. This file may be supplemented with information about the primary service.
- **inputCEP()** – StorageVET inputs a standard CEP file. This function can be used to help restart this use case at an intermediate step.

**End Result: Valid CEP and Other Outputs**

The Investor or Vendor has quantitative CEP and other information needed for producing an ESS proposal for the Regulator.

**Exceptions**

There are no available options close enough to optimal.

**Decision Support Model Structure**

The Investor chooses the best size for a storage project according to its power level and the duration of output at maximum power (energy storage capacity).

**Sets and Indexes**

- **Parameters** – The set of all StorageVET parameter identifiers, indexed by `param`.
- **ScenarioIDs** – The set of scenario identifiers, indexed by `scenario`.
- **PowerLevels** – The set of real-valued power levels to explore for the given storage project, indexed by `power`. The units of measure of a given `power` is kW.
- **Durations** – The set of real-valued durations of output at maximum power level for the given storage project, indexed by `duration`. The unit of measure of a given `duration` is hours.

**Decision Variables**

- **CandidateSizes[power, duration]** – A binary-valued indicator that the given `power` and `duration` for the given storage project meet the optimal sizing criteria.
**Scenario Variables**

The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:

- *ScenarioVariables* — Subset of *Parameters*, indexed by *scenarioVariable*. Indicates which parameters vary across the elements of *ScenarioIDs*. No unit of measure.
- *scenarioValue[scenarioVariable, scenario]* — The value that *scenarioVariable* takes for the given *scenario*. The range and units of measure are defined by the given *scenarioVariable*.

These may be supplemented by additional preferences of the Investor.

**Objective**

The Investor objective is to maximize the Total Resource Value of the *CandidateSizes*.

**Constraints**

The Investor is subject to the approvals of the Regulator, which requires that projects meet minimum threshold requirements on the various perspective tests. For instance, the Ratepayer Impact Measure needs to be positive. Violation of a constraint for a *[power, duration]* pair excludes it from *CandidateSizes*.

**Use Case 2.3. Benchmark a Proprietary Valuation Method**

*An Investor wants to identify and explain significant differences between the Investor’s proprietary results and those of StorageVET.*

This use case is similar to Use Case 1.4. Model/Data Benchmarking, but is more reflective of the perspective of the Investor for satisfying the Investor’s needs, as opposed to the Regulator’s needs.

**Actors**

The principal actor is the Investor.

**Investor**

The Investor is interested in explaining the differences between the Investor’s own proprietary model and StorageVET. There are two goals: (1) to ensure that both models are running correctly, and (2) to ensure that any significant differences can be explained satisfactorily.

**Regulator**

The Regulator is interested in understanding the basis of an Investor’s ESS proposal. If the proposed ESS performance estimates differ significantly from those obtained from StorageVET, then those differences should be explained satisfactorily.

**StorageVET**

StorageVET is used to simulate and evaluate the performance of a proposed ESS and to compare the inputs and outputs of a given proprietary model.
Information Exchange
The Investor collects information needed to evaluate an ESS project.

Regulator -> Investors

**CEP Input Data** – Reference Scenario inputs that do not differ across scenarios.

**CEP Scenarios** – Could include alternative future price curves.

Investor <-> StorageVET

The Investor provides StorageVET input files and receives output files in return. The output is in a form that enables the Investor to identify significant differences in the output and to trace their cause back to significant differences in the input.

Investor -> Regulator

**Comparison Results** – Comparison of proprietary tool results to StorageVET benchmark results.

**Project Recommendations** – Investor recommendations regarding the Investor’s evaluation of the proposed project.

Process

Generally, the Investor runs a proprietary model using inputs consistent with the given StorageVET inputs and produces output that is also in the StorageVET format. StorageVET is then used as a tool for comparing the two sets of inputs and outputs to explain differences.

**Starting Condition: Proprietary Storage Valuation Results, Scenarios, ESS Design**

The Investor has a detailed ESS design and has simulated its performance using a proprietary tool. The inputs and outputs of that tool are in StorageVET format.

- **inputModel()** – the user chooses a storage project model. This model is input and populates the model and data with default values.
- **inputScenarioSet()** – the user chooses one or more Scenario Sets for benchmarking the storage model. The user may wish to adjust the Scenario Comparison Logic for each chosen Scenario Set to best suit the benchmarking analysis. One approach is to begin with large tolerances and to tighten them down in subsequent runs.
- **displayScenarioSet()** – provide the user with a list of Storage Projects for selection. Projects represent the analysis of a storage project with both StorageVET and a proprietary model.
- **inputScenarioSet()** – the user chooses at least two Storage Project proposals to input one from a StorageVET analysis and one from a proprietary analysis. Each Storage Project Set defines a comparison of a single storage project over all Scenario Sets.
- **checkForParameterConflicts()** – checks the list of scenario parameters and the list of Storage Project parameters for conflicts or overrides and reports on significant changes.
- **createInput()** – create individual input datasets for each Storage Project for all given Scenario Sets.

**Step 1: Run StorageVET**

The Investor runs StorageVET on the ESS design for all scenarios.
• **runScenarios()** – StorageVET simulates each Storage Project Set for all given Scenario Sets.

**Step 2: Input Proprietary Results**
The Investor imports the proprietary model results into StorageVET for comparison analysis.

• **inputCEP()** – StorageVET inputs a standard CEP file for each Storage Project in the Storage Project Set that is a proprietary model.

**Step 3: Compare Proprietary Results with StorageVET Results**
StorageVET is used to compare outputs of alternative analysis methods.

• **compareScenarios()** – use the Scenario Set comparison logic to flag significant differences in inputs and outputs between the benchmark (StorageVET) project and the proprietary analysis in each Storage Project Set.

**Step 4: Trace StorageVET Results to Root Differences**
Beginning at the final output results for various metrics, the components are compared for differences, backward through the accounting and simulation processes. By these means, root differences can be traced to certain time periods and scheduling variables.

• **displayScenarioComparison()** – display for the user the top-level inputs and outputs flagged by compareScenarios(). This includes major project scoring metrics as determined by the user.

• **navigateScenarioComparison()** – the user selects input or output comparisons and drills into the upstream calculations to investigate the sources of the differences. The user may wish to tighten down certain Scenario Comparison Logic during the navigation of the comparison, in order to resolve unexplained differences.

**End Result: List of Significant Root Differences and Related Metrics**
The Investor has a list of significant output differences between the proprietary model and StorageVET and a trace of the root cause for these differences down to time periods, service schedules, service valuations, and performance test differences.

• **outputScenarioComparison()** – StorageVET creates an output file that reports the Storage Project Comparison.

• **inputScenarioComparison()** – StorageVET inputs a file that contains the Storage Project Comparison. This function can be used to restart this use case at an intermediate step.

**Exceptions**
The proprietary model does not provide sufficient reporting to trace differences in enough detail.

**Decision Support Model Structure**
The Investor identifies quantitative differences between the inputs and outputs of one or more of their proprietary models and those of StorageVET and decides which ones are significant, if any.
Sets and Indexes

- **Parameters** – The set of all StorageVET parameter identifiers, indexed by `param`.
- **Variables** – The set of all StorageVET decision variables, indexed by `variable`.
- **Identifiers** – The union of Parameters and Variables, indexed by `id`.
- **ScenarioIDs** – The set of scenario identifiers, indexed by `scenario`.
- **Models** – The set of models being compared, indexed by `model1` and `model2`. One of the models is StorageVET.

Decision Variables

- `thresholdDifferences[id, scenario]` – Values that the principle actor decides represent a significant difference when observed between models. Units of measure are defined as those of `id`. The value of this decision variable may be given in the Starting Condition.

- `significantDifference[id, model1, model2 scenario]` – A binary-valued indicator that parameter or variable `id` differs significantly between `model1` and `model2` for the given `scenario`.

Scenario Variables

The Regulator has predefined the sources of uncertainty in Use Case 1.1. Design Scenarios and Model Assumptions, and they are formalized as:

- **ScenarioVariables** — Subset of Parameters, indexed by `scenarioVariable`. Indicates which parameters vary across the elements of ScenarioIDs. No unit of measure.

- `scenarioValue[scenarioVariable, scenario]` – The value that `scenarioVariable` takes for the given `scenario`. The range and units of measure are defined by the given `scenarioVariable`.

These may be supplemented by additional preferences of the Investor.

Objective

The Investor objective is subjective in this use case. The root of their decision-making is specifying `thresholdDifferences`, which may take place as part of the use case process or given as part of the Starting Condition through some sort of external process. The Regulator may also have specified thresholds that need to be used in this analysis.

If `thresholdDifferences` is given in the Starting Condition, then there is no objective. Running StorageVET merely computes and reports on the value of `significantDifference`. 
CHAPTER 5: Operator Use Cases

The use cases in this chapter describe the various ways that an ESS operator might utilize StorageVET as a decision support system in ESS operations. Some are essential to evaluating ESS benefits, while others are more forward looking.

While the expected near-term use of StorageVET is for evaluating Storage Project proposals from the perspectives of Investors and Regulators, ESS operations form the important basis for generating revenues and other benefits from an ESS. Accordingly, a certain level of functionality for simulating ESS operations must be present in order to fully represent ESS benefits.

The more forward-looking use cases are included here in order to help Stakeholders and the StorageVET project team imagine these future uses and prepare the initial software architecture for later enhancements in support of operational uses.

The time horizon of ESS operations is typically on the order of days, because of the limited ability of small ESS resources (durations of hours) to be economically effective over longer horizons.

Use Case 3.1. Reshape Net Load Profile

An ESS Operator, at an aggregation point, wants to reshape net load over a set future time period.

Scheduling an ESS can be achieved through financial incentives and through constraints on the way it operates. In this use case, an operational constraint is added to the dispatch formulation to represent an external limitation on the ESS power input and output. This external limitation represents a requirement or limit on an aspect of the power system that is not internal to the ESS, and in this example it is called an aggregation point, at which there is a power flow criterion. The ESS should operate in such a way as to avoid violating this criterion.

Actors
The principal actor is the Operator.

Operator
The Operator wants to use an ESS to avoid violating a power flow criterion at a given aggregation point.

Forecaster
The Forecaster provides estimates of the net power flow at the aggregation point. This information can vary over time. The Forecaster may also provide time-varying limitations on the power flow at the aggregation point.
StorageVET

StorageVET schedules the ESS to charge and discharge over time in such a way as to avoid violations of the aggregation point criteria.

Information Exchange

The forecast information and ESS status and capabilities are used to simulate operations over a short period, for which the forecast remains valid.

Forecaster -> Operator

Net Load Shapes – A collection of time series of the net load at a given aggregation point that are used as starting points for the exercise to reshape them. This forecast could be probabilistic in that it consists of multiple versions having different relative likelihoods of occurring.

Residual Scenarios – Aspects of the system conditions at the aggregation point, aside from the net load, that may have bearing on a decision about how to schedule the ESS. These residual scenarios may also be probabilistic. For instance, they may include alternative price futures.

Operator -> StorageVET

Scenarios – Combinations of net load shapes and residual scenarios comprising full input scenarios for StorageVET.

Power Flow Criteria – Limitations on the aggregated power flow to which the ESS contributes.

StorageVET -> Operator

ESS Performance – Report of ESS performance over the study period, part of which is compliance with any load shape restrictions.

ESS Schedule – The operating schedule for the ESS.

Process

The Operator collects all needed scheduling information for StorageVET and reviews the output to decide how to schedule the ESS.

Starting Condition: ESS, Location, Net Load Shapes, and Probabilistic Scenarios

The Operator starts with all necessary scheduling information.

- `inputModel()` – the user chooses a storage project model. This model is input and populates the model and data with default values.
- `inputScenarioSet()` – the user chooses one or more Scenario Sets for evaluating the storage project.
- `inputScenarioSet()` – allows the user to input a defined Location Set, order to initialize this process. This Location Set has only one element representing the location of this storage project.
- `inputScenarioSet()` – allows the user to input a defined Technology Set, in order to initialize this process. The Technology Set has only one element representing the technology of this storage project.
• *inputScenarioSet()* – allows the user to input a defined Services Set, in order to initialize this process.

**Step 1: Determine Target Net Load Shape**
The target net load shape could come from information provided by the Forecaster, or it could be the residual capability of the ESS for providing additional services (see Use Case 3.3. Determine Left-Over Storage Capacity).

• *modifyScenarioSet()* – The load shape service is labeled as the primary service. The user may choose from among other services those to include in this analysis. The user-selected services with the labeled primary services define the Services Set for a given technology.

• *inputScenarioSet()* – allow the user to input data for a target load shape as a Schedule Scenario. This load shape is sufficient to represent the operating horizon, and the input may also be done automatically by using an attribute of the Load Shape Service.

**Step 2: Run StorageVET on All Scenarios**
The Operator runs StorageVET for all load profile and other combinations of factors important to the near-term dispatch.

• *runScenarios()* – StorageVET simulates each the given location, technology, and services over all elements of the Scenario Set.

**Step 3: Report Performance**
StorageVET reports metrics about net load shaping, violations of the net load criteria at the aggregation point, and possibly mitigation strategies.

• *compareScenarios()* – use the Scenario Set comparison logic to sort scenario results in the order of a user-chosen performance metric. Include high-level statistics of the performance across scenarios.

• *displayScenarioComparison()* – display for the user the top-level performance statistics flagged by *compareScenarios()*.

• *navigateScenarioComparison()* – for a given performance metric ranking, the user selects input or output performance metrics and drills down into the upstream calculations to investigate the sources of the differences between scenarios.

• *selectScenario()* – allow the user to select a storage schedule from among those computed. The storage schedule is a Schedule Scenario.

• *compareScenarios()* – StorageVET recomputes the performance metrics for each scenario using the selected storage schedule and it uses the Scenario Set comparison logic to sort scenario results in the order of a user-chosen performance metric. Include high-level statistics of the performance across scenarios.

Note that the last two functions *selectScenario()* and *compareScenarios()* allow the user to explore the decision space for the best schedule to meet the target load shape. This could be iterative through user interaction, or StorageVET may explore all combinations of schedules and scenarios to automatically select a schedule.
End Result: Reshaped Net Load That Approximates Target Shape

The Operator has an operating schedule for the near-term horizon that is effective in avoiding violations of criteria at the aggregation point.

- outputScenarioSet() – StorageVET creates an output file with the operating schedule of the storage project.
- inputScenarioSet() – StorageVET inputs an operating schedule of the storage project. This may be used to begin this process at an intermediate step.

Exceptions

Net load cannot be reshaped to meet the target net load shape.

Decision Support Model Structure

The Operator decides how to dispatch the ESS project over the coming operating horizon in order to not exceed a load shape or to track it as closely as possible.

Sets and Indexes

- Parameters – The set of all StorageVET parameter identifiers, indexed by param.
- ScenarioIDs – The set of scenario identifiers, indexed by scenario.
- Periods – The set of time periods over which the Operator will schedule the given ESS, indexed by period. Note: Each time period can have it’s a different duration of time.
- Services – The set of service capabilities of the given ESS, indexed by service.
- energy – An element of Services representing the energy flow.

Parameters

- TargetShape[period] – The target net load shape at the aggregation point. The unit of measure is kW-period.
- Penalty – A real-valued penalty cost for the objective.

Decision Variables

- Schedule[service, period] – The ESS schedule for service in period of time. The unit of measure depends on the service.
- NetLoad[period] – The net load schedule for energy service at the aggregation point in period of time. The unit of measure is kW-period.

Scenario Variables

Over an operating horizon of days, the value of the Schedule typically impacted most by weather and market phenomena. If the net load shape is affected particularly by a distributed resource, like PV, then the forecast error in its energy production is an important source of uncertainty.

Objective

The Operator objective depends on how the net load shape should be tracked. If net load is not to exceed the given target shape, then the objective is a penalty function on any level that NetLoad exceeds TargetShape.

$$\text{minimize } \text{SUM}(\text{period}, \text{Penalty} \cdot (\text{max}(0, \text{Schedule} - \text{TargetShape})))$$
If net load is to track the given target shape, then the objective is a penalty function on any level that Schedule deviates from NetLoad.

\[ \text{minimize } \sum_{\text{period}} \left( \text{Penalty} \times \max(0, \text{Schedule} - \text{TargetShape}) \right) \]

**Use Case 3.2. Co-Optimize Services Scheduling**

*The Storage Operator uses day-ahead forecasts to determine services to offer for the coming day.*

Scheduling an ESS can be achieved through financial incentives and through constraints on the way it operates. In this use case, financial incentives are used to guide scheduling over the near-term operating horizon. StorageVET is able to decide automatically for each time period which services bring the highest value, according to the chosen incentive (objective function).

**Actors**
The principal actor is the ESS Operator.

**ESS Operator**
The ESS Operator is interested in determining a near-term operating schedule that uses the full ESS capability to maximize revenue from a selected number of services.

**Scheduler**
The Scheduler is responsible for scheduling all system resources, including the ESS.

**ESS**
The energy storage system represents itself in terms of its capabilities and current operating state.

**StorageVET**
StorageVET simulates operations of the ESS over the near-term scheduling horizon.

**Information Exchange**
The Operator collects needed inputs, runs StorageVET, and uses the output for guidance on how to schedule services over the coming day.

**CAISO -> Scheduler or ESS Operator**
*Market Data* – Time series prices for energy and ancillary services.

*Unit-Specific Data* – State of charge (SOC), should the storage be optimized by the CAISO.

**ESS Operator <-> StorageVET**
*Day-Ahead Forecast Inputs* – Multiple sets of StorageVET input files for evaluation.

*Top-Ranking Services* – A list of the highest-value services of the future study period.

**ESS Operator -> Scheduler**
*Service Offers* – Offers for market services in the appropriate market format.
Scheduler <-> CAISO
Service Offers – Submission and approval of proposed schedules.

CAISO Settlements – Settlement files for the ESS performance. Used to validate and tune StorageVET.

Process
The Operator uses StorageVET to decide the ESS near-term operating schedule.

Starting Condition: Storage Parameters and Day-Ahead Forecast Inputs
The ESS operating state, and market forecasts, which may include uncertainty, are given.

• inputModel() – the user chooses a storage project model. This model is input and populates the model and data with default values.
• inputScenarioSet() – the user chooses one or more Scenario Sets for evaluating the storage project.
• inputScenarioSet() – allows the user to input a defined Location Set, order to initialize this process. This Location Set has only one element representing the location of this storage project.
• inputScenarioSet() – allows the user to input a defined Technology Set, in order to initialize this process. The Technology Set has only one element representing the technology of this storage project.
• inputScenarioSet() – allows the user to input a defined Services Set, in order to initialize this process.

Step 1: Set Service Priorities
The ESS Operator decides the service priorities for the operating horizon and specifies them in StorageVET.

• modifyScenarioSet() – The user choose the primary service from among the price-dispatched services. The user may choose from among other price-dispatched services those to include in this analysis. The user-selected services with the labeled primary service define the Services Set for a given technology.

Step 2: Run StorageVET
The Operator runs StorageVET for a given site and screens for high-value services.

• runScenarios() – StorageVET simulates each the given location, technology, and services over all elements of the Scenario Set.

Step 3: Report Service Values and Schedules
StorageVET reports value of alternative services and the concomitant ESS schedules.

• compareScenarios() – use the Scenario Set comparison logic to sort scenario results in the order of a user-chosen performance metric. Include high-level statistics of the performance across scenarios.
displayScenarioComparison() – display for the user the top-level performance statistics flagged by compareScenarios().

navigateScenarioComparison() – for a given performance metric ranking, the user selects input or output performance metrics and drills down into the upstream calculations to investigate the sources of the differences between scenarios.

selectScenario() – allow the user to select a storage schedule from among those computed.

compareScenarios() – StorageVET recomputes the performance metrics for each scenario using the selected storage schedule and it uses the Scenario Set comparison logic to sort scenario results in the order of a user-chosen performance metric. Include high-level statistics of the performance across scenarios.

Step 4: Convert to Market Offers
The Operator converts the StorageVET ESS schedules into market offers and submits them to the Scheduler.

outputScenarioSet() – StorageVET creates an output file with the operating schedule of the storage project.

inputScenarioSet() – StorageVET inputs an operating schedule of the storage project. This may be used to begin this process at an intermediate step.

End Result: Successful Transmission of Market Offers
The Scheduler has valid ESS market offers that achieve the Operator’s goals.

Exceptions
There is significant error in the forecast error ESS status.

Decision Support Model Structure
The Operator decides how to dispatch multiple services provided by the ESS project over the coming operating horizon in order to maximize profit.

Sets and Indexes

- Parameters – The set of all StorageVET parameter identifiers, indexed by param.
- ScenarioIDs – The set of scenario identifiers, indexed by scenario.
- Periods – The set of time periods over which the Operator will schedule the given ESS, indexed by period. Note: Each time period can have it's a different duration of time.
- Services – The set of service capabilities of the given ESS, indexed by service.

Decision Variables

- Schedule[service, period] – The ESS schedule for service in period of time. The unit of measure depends on the service.

Scenario Variables

Over an operating horizon of days, the value of the Schedule typically impacted most by weather and market phenomena. The Operator can formalized the specification of uncertainty using:
• **ScenarioVariables** — Subset of **Parameters**, indexed by **scenarioVariable**. Indicates which parameters vary across the elements of **ScenarioIDs**. No unit of measure.
• **scenarioValue[scenarioVariable, scenario]** – The value that **scenarioVariable** takes for the given **scenario**. The range and units of measure are defined by the given **scenarioVariable**.

**Objective**
The Operator objective depends on how the net load shape should be tracked. If net load is not to exceed the given target shape, then the objective is a penalty function on any level that **NetLoad** exceeds **TargetShape**,

\[
\text{maximize } \sum_{\text{period}} \text{Penalty} \cdot \max(0, \text{Schedule} - \text{TargetShape}).
\]

If net load is to track the given target shape, then the objective is a penalty function on any level that **Schedule** deviates from **NetLoad**,

\[
\text{minimize } \sum_{\text{period}} \text{Penalty} \cdot \max(0, \text{Schedule} - \text{TargetShape}).
\]

**Use Case 3.3. Determine Left-Over Storage Capacity**

*Given primary obligations, determine what capability and availability remain for secondary uses.*

StorageVET is given ESS capabilities and a schedule of primary service obligations and determines a schedule of residual ESS capability that can be used for added near-term value. Primary obligations could be providing power quality or backup power (see Use Case 3.5. Power Quality and Use Case 3.6. Backup Power).

**Actors**
The principal actor is the Operator.

**Operator**
The Operator has an ESS schedule of primary service obligations and wants to further investigate the use of the ESS for additional services.

**StorageVET**
StorageVET reduces the ESS capability according to a given schedule of operations, including service obligations. It produces a schedule of residual ESS capabilities.

**ESS**
The energy storage system is represented by a time schedule of its capabilities.

**Information Exchange**
The Operator collects ESS capabilities and obligations to determine what remains useful for additional services.

**ESS -> StorageVET**
*Starting ESS State* – Time series of the future state of charge and available power input and output levels.
**StorageVET** \(\rightarrow\) **Operator**

*Updated ESS State* – Updated time series of the future state of charge and available power input and output after the primary service has been scheduled.

**Process**

The Operator collects the ESS status and obligations and determines the remaining capability.

**Starting Condition: Starting ESS State, Primary Service, Scenarios**

The ESS has an existing state and schedule of service obligations. Future scenarios represent a range of conditions over which the service obligations need to be provided and may have associated probabilities. The remaining capability, likewise, may have a range of values with associated probabilities.

- **inputModel()** – the user chooses a storage project model. This model is input and populates the model and data with default values.
- **inputScenarioSet()** – the user chooses one or more Scenario Sets for evaluating the storage project.
- **inputScenarioSet()** – allows the user to input a defined Location Set, order to initialize this process. This Location Set has only one element representing the location of this storage project.
- **inputScenarioSet()** – allows the user to input a defined Technology Set, in order to initialize this process. The Technology Set has only one element representing the technology of this storage project.
- **inputScenarioSet()** – allows the user to input a defined Services Set, in order to initialize this process.

**Step 1: Run StorageVET for All Scenarios to Meet the Primary Obligation**

The Operator runs StorageVET for the batch of scenarios to determine the range or distribution of remaining ESS capability.

- **runScenarios()** – StorageVET simulates each the given location, technology, and services over all elements of the Scenario Set.

Note that this step may not be needed if the scenarios have already been run and the schedules saved as output. In which case, the user may use **inputScenarioSet()**.

**Step 2: Determine the Updated ESS Availability and Capability**

The Operator decides from ESS reporting the remaining ESS availability and capability.

- **computeAvailability()** – StorageVET uses the storage schedule to determine the necessary required power and energy at the start and end of the operating horizon that is needed to achieve the given performance. The remaining power and energy in the system is available for other uses. This analysis is conducted across all scenarios, which may be probabilistic, and StorageVET computes performance metric statistics across scenarios.
• `compareScenarios()` – use the Scenario Set comparison logic to sort availability results for the scenarios in the order of a user-chosen performance metric. Include high-level statistics of the performance across scenarios.

• `displayScenarioComparison()` – display for the user the top-level performance statistics flagged by `compareScenarios()`.

• `navigateScenarioComparison()` – for a given performance metric ranking, the user selects input or output performance metrics and drills down into the upstream calculations to investigate the sources of the differences across scenarios.

**Step 3: Report Updated ESS State**

This is an equivalent ESS state after the primary service has been scheduled.

• `outputScenarioSet()` – StorageVET creates an output file with the available power and energy over the operating interval in the form of a single Schedule Scenario within a Scenario Set.

• `inputScenarioSet()` – StorageVET inputs the available power and energy over the operating schedule in the form of a single Schedule Scenario within a Scenario Set. This may be used to begin this process at an intermediate step.

Note that storage availability has not been used explicitly in the Use Cases. Instead, it has been assumed that this information is part of the technology model.

**End Result: Updated ESS State**

The Operator has a set of parameters to define a new and dynamically changing ESS.

**Exceptions**

Primary service obligation exceeds current ESS capability and availability.

**Decision Support Model Structure**

The Operator needs to determine the resulting ESS state, given a starting state and a schedule of services.

**Sets and Indexes**

• `ScenarioIDs` – The set of scenario identifiers, indexed by `scenario`.

• `Periods` – The set of time periods over which the Operator will schedule the given ESS, indexed by `period`. Note: Each time period can have it’s a different duration of time.

• `Services` – The set of service capabilities of the given ESS, indexed by `service`.

• `StateVariables` – The set state variables of the given ESS, indexed by `stateVariable`. These include the state of charge and availability for all services.

**Parameters**

• `StartState[stateVariable, period]` – The value for `stateVariable` in `period` of time. The unit of measure depends on the `stateVariable`.

• `Schedule[service, period]` – The ESS schedule for `service` in `period` of time. The unit of measure depends on the `service`.
Service availability may be specified according to upper and lower limits for provision of the service and these limits may have probability distributions. More detail is beyond the scope of this description of model structure.

**Decision Variables**
- \( \text{EndState}[\text{stateVariable}, \text{period}] \) – The value for \( \text{stateVariable} \) in \( \text{period} \) of time. The unit of measure depends on the \( \text{stateVariable} \).

**Scenario Variables**
Over an operating horizon of days, the value of the \( \text{Schedule} \) is typically impacted most by weather and market phenomena. While the start \( \text{StartState} \) and \( \text{Schedule} \) are given, it may be that \( \text{Schedule} \) has an added \( \text{scenario} \) index to represent uncertainty in the actual operations. In this case, the \( \text{EndState} \), too, would require a \( \text{scenario} \) index.

**Objective**
The Operator has no specific objective in this use case. However, it may turn out that determining the \( \text{EndState} \) is less of a rote calculation and depends on Operator preferences, which may then become part of the objective.

**Use Case 3.4. Schedule an Outage**
*Determine a period of time in the near future during which it is most economical to schedule an outage.*

This use case supports the Operator in making decisions about ESS maintenance. The Operator has in mind some near-term requirement (time window) for conducting maintenance on all or part of the ESS. The maintenance may require a partial or full derating of the ESS capabilities. This requirement is evaluated over the scheduling horizon, and various options for scheduling maintenance are ranked and compared by StorageVET.

The maintenance horizon may be extended to weeks, compared to the operating horizon of days.

**Actors**
The principal actor is the Operator.

**Operator**
The Operator wants to conduct maintenance and suggests parameters, like duration, and options and exclusions.

**StorageVET**
StorageVET ranks suggestions for maintenance options according to performance tests determined by the Operator.

**Information Exchange**
The Operator collects information about the maintenance requirement, scenarios, performance metrics, and ESS performance in order to decide when to conduct maintenance.
ESS -> StorageVET

Starting ESS State – Time series of the future state of charge and available power input and output levels.

StorageVET -> Operator

Outage Windows – Prioritized list of time periods of sufficient duration for conducting maintenance.

Process

The Operator runs StorageVET and compares options for a ranked set of maintenance windows.

Starting Condition: ESS State, Primary Service, Outage Scenarios, Operating Scenarios

The Operator has all needed input to evaluate the ESS performance over the maintenance horizon. This can include full and partial deratings of the ESS capabilities.

- **inputModel()** – the user chooses a storage project model. This model is input and populates the model and data with default values.
- **inputScenarioSet()** – the user chooses one or more Uncertainty Scenario Sets for evaluating the storage project.
- **inputScenarioSet()** – allows the user to input a defined Location Set, order to initialize this process. This Location Set has only one element representing the location of this storage project.
- **inputScenarioSet()** – allows the user to input a defined Technology Set, in order to initialize this process. The Technology Set has only one element representing the technology of this storage project.
- **inputScenarioSet()** – allows the user to input a defined Services Set, in order to initialize this process.

- **inputScenarioSet()** – StorageVET inputs the available power and energy over the operating schedule. This may be a large selection of schedules that includes a starting schedule of the base case availability.

Step 1: Run StorageVET for All Scenarios

The Operator runs StorageVET on a batch of scenarios and compares the ranked performance of various ESS maintenance schedules.

- **displayScenarioSet()** – provide the user with a visualization of availability scenarios.
- **defineScenarioSet()** – the user chooses from among all availability schedules those they wish to evaluate. The user gives this set a unique identifier. Update schedule display to show schedules as they are added to the set. The selected schedules define the Availability Set.
- **identifyReferenceScenario()** – the user identifies from among all schedules one base case schedule, which represents the starting condition of the storage project. The Base Schedule is associated with the set identifier. Use special marking on the Base Schedule in the display.
- **defineScenarioComparison()** – the user defines performance metrics for comparing maintenance schedules. These metrics can be standard performance tests, like those defined
in the CEP, or user definitions. Availability Comparison Logic is an attribute of the Availability Set. The point of Maintenance screening is to find schedules that perform most poorly for most operating metrics. Therefore, the Availability Comparison Logic is the opposite of other operating use cases.

- `runScenarios()` – StorageVET simulates the cross product of the Availability Set and Uncertainty Scenario Set.

### Step 2: Determine Low-Value Operating Windows

Time periods (windows) for operations having low economic value are candidates for conducting maintenance. These windows should be filtered using maintenance outage constraints. The periods that meet the constraints pass this step.

- `compareScenarios()` – use the Availability Set comparison logic to flag significant differences in inputs or outputs between the Base Schedule and other Schedules in the Availability Set.
- `displayScenarioComparison()` – display for the user the top-level inputs and outputs flagged by `compareScenarios()`.
- `navigateScenarioComparison()` – the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences.

### Step 3: Run StorageVET

StorageVET is run with degraded service capabilities for each chosen outage window and scenario.

- `runScenarios()` – StorageVET simulates each the given location, technology, and services over the cross product of the Availability Set and Uncertainty Scenario Set.

### Step 4: Rank Outage Windows by Opportunity Cost

StorageVET creates a ranked list of maintenance schedules for the Operator to evaluate over the performance metrics and scenarios.

- `compareScenarios()` – use the Availability Set comparison logic to flag significant differences in inputs or outputs between the Base Schedule and other Schedules in the Availability Set.
- `displayScenarioComparison()` – display for the user the top-level inputs and outputs flagged by `compareScenarios()`.
- `navigateScenarioComparison()` – the user selects input or output comparisons and drills down into the upstream calculations to investigate the sources of the differences.

### End Result: Outage Windows

The Operator has a prioritized list of maintenance schedules.

- `outputScenarioSet()` – StorageVET creates an output file with the available power and energy over the operating interval that corresponds to the user-selected maintenance schedule.

### Exceptions

The Operator cannot meet primary obligations and conduct an important outage.
Decision Support Model Structure

The Operator decides on a period of time over which to schedule an outage.

Sets and Indexes

- **Parameters** – The set of all StorageVET parameter identifiers, indexed by `param`.
- **ScenarioIDs** – The set of scenario identifiers, indexed by `scenario`.
- **Periods** – The set of time periods over which the Operator will schedule the given ESS, indexed by `period`, `startPeriod`, and `endPeriod`. Note: Each time period can have a different duration of time.
- **Services** – The set of service capabilities of the given ESS, indexed by `service`.
- **StateVariables** – The set state variables of the given ESS, indexed by `stateVariable`. These include the state of charge and availability for all services.

Parameters

- `StartState[stateVariable, period]` – The value for `stateVariable` in `period` of time. The unit of measure depends on the `stateVariable`.
- `Schedule[service, period]` – The ESS schedule for `service` in `period` of time. The unit of measure depends on the `service`.
- `Penalty[stateVariable, period1, period2]` – A real-valued cost applied to differences between `StartState[stateVariable, period1]` and `StartState[stateVariable, period2]`, where the maintenance begins at `startPeriod` of time and ends at `endPeriod` of time.

Decision Variables

- `Outage[service, period]` – A binary-valued indicator that an outage of `service` occurs in `period` of time.
- `Derating[service, period]` – A percentage-valued indicator in [0,100] indicating the level of the outage in terms of `derating` service in `period` of time.

Scenario Variables

Over an operating horizon of days, the value of the `Schedule` typically impacted most by weather and market phenomena. While the start `StartState` and `Schedule` are given, it may be that `Schedule` has an added `scenario` index to represent uncertainty in the actual operations.

Objective

The Operator objective is to minimize the cost of the outage, which is in terms of the difference in profit lost over the maintenance period, plus any penalties for the state deviating between the `startPeriod` of time and ends at `endPeriod` of time.

Use Case 3.5. Power Quality

The Operator uses ESS to maintain power quality according to a given service agreement.

The ESS is used to provide power quality (PQ) services, and the requirements for these services can vary over time, depending on how the external system is operating and other conditions, like weather. StorageVET is used to schedule the ESS for this service.
Actors
The principal actor is the Operator.

Operator
The Operator wants to schedule the ESS for power quality service.

StorageVET
StorageVET provides a power quality service schedule for the ESS.

ESS
The energy storage system has a starting state and forecasted capability over the scheduling horizon.

Information Exchange
The Operator collects all of the needed information on status, capability, power quality needs, and ESS power quality schedule in order to finalize the schedule.

ESS -> StorageVET
ESS State – Time series of the future state of charge and available power input and output levels.

StorageVET -> Operator
PQ Schedule – A time series of reserved energy and power to use for PQ service.

Process
The Operator collects StorageVET input, runs StorageVET, and evaluates the output for deciding the PQ schedule.

Starting Condition: ESS State, PQ Service Agreement and Schedule, Non-Performance Penalties, PQ Event Scenarios
The Operator has all information to run StorageVET for scheduling PQ service, including the event scenarios and likely arrival statistics.

- inputScenarioSet() – StorageVET inputs the available power and energy over the operating schedule. This may be used to begin this process at an intermediate step.
- inputModel() – the user chooses a storage project model. This model is input and populates the model and data with default values.
- inputScenarioSet() – the user chooses one or more Scenario Sets for evaluating the storage project.
- inputScenarioSet() – allows the user to input a defined Location Set, order to initialize this process. This Location Set has only one element representing the location of this storage project.
- inputScenarioSet() – allows the user to input a defined Technology Set, in order to initialize this process. The Technology Set has only one element representing the technology of this storage project.
• inputScenarioSet() – allows the user to input a defined Services Set, in order to initialize this process. This includes the PQ Service Schedule.
• inputScenarioSet() – the user chooses one or more PQ Scenario Sets for evaluating the storage project.

**Step 1: Determine Power Deratings for PQ Service Schedule and All PQ Event Scenarios**

The Operator uses a pre-calculation to derate the ESS for PQ service. This calculation may be part of StorageVET functionality.

• computeAvailability() – StorageVET uses the PQ Service Schedule to determine the necessary required power and energy at the start and end of the operating horizon that is needed to achieve the given performance. The remaining power and energy in the system is available for other use cases. This analysis is conducted across all PQ Scenario Sets, which are probabilistic, and operating scenarios defined by the user. StorageVET computes statistically the power deratings that are expected to be sufficient for providing PQ service over the operating horizon. This takes the form of an Availability Schedule, with probabilistic representation of availability for each scheduling interval.

**Step 2: Run StorageVET**

Run for all PQ event scenarios to determine their impact on SOC.

• runScenarios() – StorageVET simulates each the given location, technology, and services over the cross product of the PQ Scenario Sets and Scenario Set.
• compareScenarios() – use the PQ Scenario Set comparison logic to sort results in the order of a user-chosen performance metric. Include high-level statistics of the performance across all scenarios.
• displayScenarioComparison() – display for the user the top-level performance statistics flagged by compareScenarios().
• navigateScenarioComparison() – for a given performance metric ranking, the user selects input or output performance metrics and drills down into the upstream calculations to investigate the sources of the differences across scenarios.

**End Result: PQ Schedule**

A schedule of energy and power reserve held for providing PQ service. This also includes a schedule of PQ service value.

• outputScenarioSet() – StorageVET creates an output file with the operating schedule of the storage project.
• outputScenarioSet() – StorageVET creates an output file with the available power and energy over the operating interval.

**Exceptions**

The ESS state is not able to meet the PQ service agreement and schedule.
Decision Support Model Structure

The Operator decides how to schedule power quality service over the operating decision horizon of a few days.

Sets and Indexes

- **Parameters** – The set of all StorageVET parameter identifiers, indexed by `param`.
- **ScenarioIDs** – The set of scenario identifiers, indexed by `scenario`.
- **Periods** – The set of time periods over which the Operator will schedule the given ESS, indexed by `period`, `startPeriod`, and `endPeriod`. Note: Each time period can have its a different duration of time.
- **Services** – The set of service capabilities of the given ESS, indexed by `service`.
  - `power_quality` – The service of power quality.
- **StateVariables** – The set state variables of the given ESS, indexed by `stateVariable`. These include the state of charge and availability for all services.

Parameters

- **StartState[stateVariable, period]** – The ESS value for `stateVariable` in `period` of time. The unit of measure depends on the `stateVariable`.
- **Schedule[service, period]** – The minimum level of ESS `service` in `period` of time, where `service` is not `power_quality`. The unit of measure depends on the `service`.
- **Penalty[service, period]** – The penalty for not providing sufficient `service` in `period` of time.

Decision Variables

- **Schedule[power_quality, period]** – The schedule for `power_quality` service in `period` of time.

Scenario Variables

Over an operating horizon of days, the value of the **Schedule** typically impacted most by weather and market phenomena. While the start **StartState** and **Schedule** are given, it may be that **Schedule** has an added `scenario` index to represent uncertainty in the actual operations.

The arrival of a power quality event is also a source of uncertainty and is formalized as:

- **EventArrival[period]** – A probability that a power quality event will arrive in period of time.
- **EventDerating[service, period]** – A percentage-valued indicator in [0,100] indicating of derating of `service` in `period` of time due to a power quality event.

Objective

The Operator objective is to minimize the opportunity cost of providing `power_quality` service, plus any `Penalty` for deviation from the **Schedule**.

**Use Case 3.6. Backup Power**

*The Operator ensures that domestic power is available according to a given service agreement.*

The ESS is used to provide backup power services, and the requirements for these services can vary over time, depending on how the external system is operating and other conditions, like weather. StorageVET is used to schedule the ESS for this service.
Actors
The principal actor is the Operator.

Forecaster
The Forecaster provides estimates of the arrival of outages that require backup power and the quantities of power and energy needed to successfully provide backup power. This information varies over time.

Operator
The Operator wants to schedule the ESS for backup power service.

StorageVET
StorageVET provides a backup power service schedule for the ESS.

ESS
The energy storage system has a starting state and forecasted capability over the scheduling horizon.

Information Exchange
The Operator collects all of the needed status, capability, and power outage event arrivals in order to finalize the schedule.

Forecaster -> Operator
The Forecaster provides statistical information about outage arrivals over the scheduling horizon. This could be in the form of an outage arrival distribution.

ESS -> StorageVET
ESS State – Time series of the future state of charge and available power input and output levels.

StorageVET -> Operator
Backup Power Schedule – A time series of reserved energy and power to use for backup service.

Customer -> Operator
The Customer provides the ESS operator with a service agreement that specifies the minimum number of outage hours, loss of load probability, and penalties for lack of service. This agreement may also include statistics in terms of SAIFI (System Average Interruption Frequency Index), SAIDI (System Average Interruption Duration Index), and so on.

Process
The Operator collects StorageVET inputs to evaluate backup power service levels, given forecasts of future events, and decides on a backup power service schedule.

Starting Condition: Service Agreement, Non-Performance Penalties, Outage Arrival Distribution
The Operator has all needed StorageVET inputs and a service agreement.
• *inputScenarioSet()* – StorageVET inputs the available power and energy over the operating schedule. This may be used to begin this process at an intermediate step.
• *inputModel()* – the user chooses a storage project model. This model is input and populates the model and data with default values.
• *inputScenarioSet()* – the user chooses one or more Scenario Sets for evaluating the storage project.
• *inputScenarioSet()* – allows the user to input a defined Location Set, order to initialize this process. This Location Set has only one element representing the location of this storage project.
• *inputScenarioSet()* – allows the user to input a defined Technology Set, in order to initialize this process. The Technology Set has only one element representing the technology of this storage project.
• *inputScenarioSet()* – allows the user to input a defined Services Set, in order to in order to initialize this process. This includes the Backup Service Schedule.
• *inputScenarioSet()* – the user chooses one or more Backup Scenario Sets for evaluating the storage project. These contain information about the service agreement, non-performance penalties, and outage arrival distribution.

**Step 1: Determine Outage Scenarios**
The Operator uses StorageVET to define a set of power outage scenarios using their arrival distributions.

• *computeAvailability()* – StorageVET uses the Backup Service Schedule to determine the necessary required power and energy at the start and end of the operating horizon that is needed to achieve the given performance. The remaining power and energy in the system is available for other use cases. This analysis is conducted across all Backup Scenario Sets, which are probabilistic, and operating scenarios defined by the user. StorageVET computes statistically the power deratings and energy requirements that are expected to be sufficient for providing Backup Power service over the operating horizon. This takes the form of an Availability Schedule, with a probabilistic representation of availability for each scheduling interval.

**Step 2: Run StorageVET**
The Operator runs StorageVET for all power outage event scenarios to determine their impacts and to determine the backup power service performance metrics.

• *runScenarios()* – StorageVET simulates each the given location, technology, and services over the cross product of the Backup Scenario Sets and Scenario Set.
• *compareScenarios()* – use the Backup Scenario Set comparison logic to sort results in the order of a user-chosen performance metric. Include high-level statistics of the performance across all scenarios.
• *displayScenarioComparison()* – display for the user the top-level performance statistics flagged by *compareScenarios()*.
• **navigateScenarioComparison()** – for a given performance metric ranking, the user selects input or output performance metrics and drills down into the upstream calculations to investigate the sources of the differences across scenarios.

**End Result: Reserve Schedule**

The Operator has a reserve schedule of energy and power to use for backup power, and a schedule of backup service value.

• **outputScenarioSet()** – StorageVET creates an output file with the operating schedule of the storage project.
• **outputScenarioSet()** – StorageVET creates an output file with the available power and energy over the operating interval.

**Exceptions**

The Operator is not able to use the ESS to mitigate the expected outages while meeting the service agreement, and as a result the schedule of backup power service value goes negative over the scheduling horizon.

**Decision Support Model Structure**

The Operator decides how to schedule backup service over the operating decision horizon of a few days.

**Sets and Indexes**

- **Parameters** – The set of all StorageVET parameter identifiers, indexed by `param`.
- **ScenarioIDs** – The set of scenario identifiers, indexed by `scenario`.
- **Periods** – The set of time periods over which the Operator will schedule the given ESS, indexed by `period`, `startPeriod`, and `endPeriod`. Note: Each time period can have it’s a different duration of time.
- **Services** – The set of service capabilities of the given ESS, indexed by `service`.
  - `backup_power` – The service of power quality.
- **StateVariables** – The set state variables of the given ESS, indexed by `stateVariable`. These include the state of charge and availability for all services.

**Parameters**

- `StartState[stateVariable, period]` – The ESS value for `stateVariable` in `period` of time. The unit of measure depends on the `stateVariable`.
- `Schedule[service, period]` – The minimum level of ESS `service` in `period` of time, where `service` is not `backup_power`. The unit of measure depends on the `service`.
- `Penalty[service, period]` – The penalty for not providing sufficient `service` in `period` of time.

**Decision Variables**

- `Schedule[backup_power, period]` – The schedule for `power_quality` service in `period` of time.
Scenario Variables

Over an operating horizon of days, the value of the Schedule typically impacted most by weather and market phenomena. While the start StartState and Schedule are given, it may be that Schedule has an added scenario index to represent uncertainty in the actual operations.

The arrival of a backup power event is also a source of uncertainty and is formalized as:

- EventArrival[period] – A probability that a backup power event will arrive in period of time.
- EventDerating[service, period] – A percentage-valued indicator in [0,100] indicating of derating of service in period of time due to a power quality event.

Objective

The Operator objective is to minimize the opportunity cost of providing backup_power service, plus any Penalty for deviation from the Schedule.

Use Case 3.7. Calibrate Degradation Model

The Operator wants a set of ESS degradation model parameters that accurately represent the observed ESS degradation.

The Operator uses StorageVET to calibrate an ESS degradation model.

Actors

The principal actor is the Operator.

Operator

The Operator wants to calibrate an ESS degradation model to help determine its ongoing capabilities or to help schedule maintenance.

StorageVET

StorageVET compares time series of ESS status and use with the parameters of an ESS degradation model in order to improve the parameter settings to more accurately represent the actual ESS status now and in the future.

Information Exchange

The Operator collects time series information about ESS use and capabilities and alternative parameter settings and their performance in representing the ESS degradation.

Operator <-> StorageVET

The Operator provides StorageVET with time series of observed battery capability and use and receives back a comparison of the actual and modeled capability for that use.

ESS Performance Schedules – A time series of ESS actual performance capabilities between maintenance periods.

ESS Use Schedules – Time series of actual ESS operating schedules.
Degradation Model Parameters – Starting values for the ESS degradation model parameters, and improved values that result from this use case.

Process
The Operator collects historical ESS information and compares actual performance with a StorageVET model of battery degradation.

Starting Condition
The Operator has historical time series of battery performance and use, along with any maintenance and expected changes in performance.

- inputModel() – the user chooses a storage project model. This model is input and populates the model and data with default values.
- displayScenarioSet() – provide the user with a list of historical storage schedules for selection. Schedules represent the actual performance of a storage project.
- defineScenarioSet() – the user chooses historical storage schedules to define the Storage Schedule Set. The user gives this set a unique identifier. Storage Scenario Sets have creation and modification date attributes and version number attributes that are maintained automatically.
- defineScenarioComparison() – the user defines for each schedule scenario the threshold logic for comparisons. This logic is in terms of absolute or one-way differences between like inputs or outputs between the given scenario and other scenarios in the Schedule Scenario Set. Inputs will be parameters that affect degradation and outputs will be performance measures and statistics for power output, and energy output, and state of charge. Each Scenario Set can have different comparison logic, or can use default comparison logic, or can derive logic from the scenario variables. Schedule Scenario Set comparison logic is an attribute of the Scenario Set.
- inputScenarioSet() – the user selects a set of schedules of past battery performance. Each Schedule Set defines the actual or simulated operation of the storage project corresponding to storage.
- outputScenarioSet() – the user selects a Schedule Scenario Set and StorageVET creates an output file that fully defines this Schedule Scenario Set.
- inputScenarioSet() – the user can choose to input one or more Schedule Scenario Sets and begin the process at an intermediate step.
- inputScenarioSet() – StorageVET inputs the actual or simulated available power and energy over the operating schedules defined in the Schedule Scenario Set. This may be used to begin this process at an intermediate step.
- checkForParameterConflicts() – checks all Scenario Sets and the list of storage degradation parameters for conflicts or overrides and reports on significant changes.

Step 1: Set Up StorageVET Scenarios for All Periods of Operations Between Maintenance
The Operator creates StorageVET input files from each post-maintenance hour to the next maintenance period, including the starting ESS performance.

- createInput() – create individual input datasets for each schedule.
Step 2: Run StorageVET

The Operator runs StorageVET with the starting status and given use schedules in order to produce performance metrics about degradation.

- runScenarios() – StorageVET simulates each schedule in the Schedule Scenario Sets.

Step 3: Run StorageVET Degradation Parameter Analysis

The Operator runs a separate function in StorageVET that fits degradation parameters to best reproduce the actual degradation of the ESS compared to the simulated degradation schedules.

- compareScenarios() – use the Scenario Set comparison logic to flag significant differences in inputs and outputs between the simulated (StorageVET) schedules and the historical schedules in each Schedule Scenario Set.
- displayScenarioComparison() – display for the user the top-level inputs and outputs flagged by compareScenarios(). This includes major project scoring metrics as determined by the user.
- navigateScenarioComparison() – the user selects input or output comparisons and drills into the upstream calculations to investigate the sources of the differences. The user may wish to tighten down certain Schedule Scenario Comparison Logic during the navigation of the comparison, in order to resolve unexplained differences.
- outputScenarioComparison() – StorageVET creates an output file that reports the Schedule Scenario Comparison.
- inputScenarioComparison() – StorageVET inputs a file that contains the Storage Project Comparison. This function can be used to restart this use case at an intermediate step.

Note that parameter optimization may be done automatically by using the above functionality to minimize a least squares differences between actual and simulated schedules, but this requires the ability to compute gradients of comparison performance with respect to each degradation parameter.

End Result: Improved Degradation Model Behavior

The Operator has a new and improved set of ESS degradation parameters.

- outputScenarioSet() – allow the user to save the values of the degradation parameters for the given technology as a Scenario Set having a single Scenario element.
- inputScenarioSet() – the user can choose to input degradation parameters as a Scenario Set having a single Scenario element and begin the process at an intermediate step.

Exceptions

The StorageVET parameters cannot reproduce the ESS degradation in a satisfactory manner.

Decision Support Model Structure

The Operator wants to calibrate parameters of the ESS degradation model using known system states and measured capabilities.

Sets and Indexes

- Parameters – The set of all StorageVET parameter identifiers, indexed by param.
• **DegradationParameters** – A subset of *Parameters* that are part of the degradation model, indexed by `degradationParam`.

• **ScenarioIDs** – The set of scenario identifiers, indexed by `scenario`.

• **Periods** – The set of time periods over which the Operator will schedule the given ESS, indexed by `period`, `startPeriod`, and `endPeriod`. Note: Each time period can have it’s a different duration of time.

• **Services** – The set of service capabilities of the given ESS, indexed by `service`.

• **StateVariables** – A subset of *Parameters* that are state variables of the given ESS, indexed by `stateVariable`. These include the state of charge and availability for all services, especially in terms of how the degradation model affects system state.

**Parameters**

• `State[stateVariable, period]` – The observed ESS state for `stateVariable` in `period` of time. The unit of measure depends on the `stateVariable`.

• `Schedule[service, period]` – The minimum level of ESS `service` in `period` of time, where `service` is not `backup_power`. The unit of measure depends on the `service`.

**Decision Variables**


• `DegradationDerate[service, period]` – A percentage-valued indicator in [0, 100] indicating derating of `service` in `period` of time due to a power quality event. This variable is a function if `DegradationInput` and `Schedule`.

**Scenario Variables**

The observed state may have errors in measurement, which means that `State` could have the `scenario` index added. The `Schedule` may also have errors and need the `scenario` index.

**Objective**

The Operator objective is to minimize the sum of mean-square differences between the `DegradationDerate` from the actual `State`.
CHAPTER 6
Use Case Ranking

This chapter describes priorities for development of the Use Cases from the perspective of the StorageVET project team, who think more in terms of resource requirements and difficulty, and the StorageVET stakeholders, who think more in terms of usefulness.

Ranking by the StorageVET Project Team

Table 3 has a ranking by the EPRI project team according to priorities of meeting the project goals and tasks that must be completed early in the development process. This ranking was conducted informally among the StorageVET project team members according to the immediate industry needs, and according to which use cases provide foundational capabilities needed for other use cases to function properly.

<table>
<thead>
<tr>
<th>Regulator</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.</td>
<td>Design Scenarios and Model Assumptions</td>
<td>Very Important</td>
</tr>
<tr>
<td>1.2.</td>
<td>Compare ES Project Benefits, Costs, &amp; Risks</td>
<td>Very Important</td>
</tr>
<tr>
<td>1.3.</td>
<td>Design Incentives</td>
<td>Potentially Important</td>
</tr>
<tr>
<td>1.4.</td>
<td>Model / Data Benchmarking</td>
<td>Potentially Important</td>
</tr>
<tr>
<td>1.5.</td>
<td>Define Benchmark Model for Improving CEP</td>
<td>Potentially Important</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Investor</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.</td>
<td>Screen Feasible Locations</td>
<td>Very Important</td>
</tr>
<tr>
<td>2.2.</td>
<td>Size a Project by Location, Primary Service, and Technology</td>
<td>Very Important</td>
</tr>
<tr>
<td>2.3.</td>
<td>Benchmark a Proprietary Valuation Method</td>
<td>Potentially Important</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operator</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.</td>
<td>Reshape Net Load Profile</td>
<td>Very Important</td>
</tr>
<tr>
<td>3.2.</td>
<td>Co-Optimize Services Scheduling</td>
<td>Very Important</td>
</tr>
<tr>
<td>3.3.</td>
<td>Determine Left-Over Storage Capacity</td>
<td>Very Important</td>
</tr>
<tr>
<td>3.4.</td>
<td>Schedule an Outage</td>
<td>Potentially Important</td>
</tr>
<tr>
<td>3.5.</td>
<td>Backup Power</td>
<td>Very Important</td>
</tr>
<tr>
<td>3.6.</td>
<td>Power Quality</td>
<td>Potentially Important</td>
</tr>
<tr>
<td>3.7.</td>
<td>Calibrate Degradation Model</td>
<td>Potentially Important</td>
</tr>
</tbody>
</table>

The project team is currently conducting a stakeholder survey, using this document as background material, to identify the priorities of the potential user community and other technical experts.
Ranking by StorageVET Stakeholders

EPRI conducted a survey of stakeholders to determine their priorities for Use Cases, Services, and Technologies. The respondents identified the key purpose they identified for a public energy storage valuation tool.

- To validate internal modeling regarding the economic feasibility of energy storage projects to financial institutions and clients.
- To compare results to proprietary economic models for validation.
- To understand the drivers and key applications of Energy Storage Systems from utilities and market perspective.
- To help design systems that better meet customer requirements.
- Provide a common view and consistent methodology of the value of ES within a use case and specific location.

**Figure 1: Survey Respondents by Type**

![Survey Respondents by Type](image)

**STAKEHOLDERS**

- Researchers: 21%
- Project Developers: 17%
- Consultants: 21%
- Utilities: 4%
- Manufacturers: 37%

Use Case Rankings

- **Investor** – Size a Project by Location, Primary Service, Technology and Screen Feasible Locations, and Benchmark a Proprietary Valuation Method.
- **Operators** – Reshape Net Load Profile Co-Optimize Services Scheduling, and Determine Leftover Storage Capacity.
Data is based on users who self-identified as having a 4 or greater out of 5 as far as involvement in creating or using storage valuation models. Battery technology is clearly preferred, followed by thermal storage and flywheels.
Services Rankings

Figure 3: Top Priority Services

Data is based on users who self-identified as having a 4 or greater out of 5 as far as involvement in creating or using storage valuation models

**Data Source Ranking**

- CAISO OASIS data  26%
- Weather Data  15%
- Green Button Load Data  14%
- OPEN EI  11%
References


APPENDIX A
TABLE OF ACRONYMS

AEO – Annual Energy Outlook
CAES – compressed air energy storage
CAISO – California Independent System Operator
CPUC – California Public Utilities Commission
CEP – Consistent Evaluation Protocol
DER – distributed energy resource
DRP – distribution resources plan
ESS – energy storage system
FERC – Federal Energy Regulatory Commission
IPP – independent power producer
PQ – power quality
SOC – state of charge
T&D – transmission and distribution
TOU – time of use
APPENDIX B: 
STORAGEVET INPUT INFORMATION

StorageVET input data reflects multiple levels of information over a future operations or planning period.

National Information

National-level information describes things like fuel prices, economic conditions, financial parameters, national policies, and any other inputs that are not much affected by location. National information should be internally consistent—for example, relatively high fuel prices and interest rates are not usually associated with sustainable economic conditions. A good source of U.S. national information, with multiple important scenarios, is the Energy Information Administration (EIA) Annual Energy Outlook (AEO) [1]. The 2015 version of the AEO has a reference scenario and sensitivity scenarios that investigate future economic growth and alternative natural gas and oil prices.

Location-Specific Information

Location-level information describes electricity prices, incentive programs, connection charges, and capital costs.

Electricity Market Information

Descriptions of electricity market products and price information are available commercially and from planning processes of the California Energy Commission and CAISO. Products include the following:

Energy – Energy is traded multiple times per day, for different commitment periods. A common product used in valuation is the day-ahead energy (DAE), which is traded about one day prior to the start of the commitment period (usually midnight) for a 24-hours duration.

Ramping – The net load changes over short durations of hours, and these changes, up and down, can be significant to the extent that they affect resource adequacy and grid reliability. The CAISO is considering new services provided generally by controllable resources, of which storage can be a significant provider.

Ancillary Services – These services are reservations of power transfer capability for minimum duration times. A list was provided in Table 2.

Capacity – Often defined as the ability of storage to transfer power for a minimum duration of 4 hours during peak load periods. The peak period is typically during summer months during hours of highest net load (14:00 – 18:00).

Avoided T&D Costs

Because energy storage can reduce net load during peak demand periods, capacity requirements for transmission and distribution (T&D) infrastructure can be reduced. In some
cases, operational expenses can also be reduced by energy storage. For example, storage may ameliorate congestion on certain transmission paths, reducing losses. All quantifiable avoided costs, whether capital or operational, are considered economic benefits of energy storage.

**StorageVET Configuration**

StorageVET is designed to answer the following general questions. Some use cases may only consider a subset of these:

- Is a storage system with given technical characteristics capable of modifying a load such that the resulting net load meets certain requirements?
- Can a storage system with given technical characteristics provide ancillary services with specific requirements?
- Can a storage system with given technical characteristics successfully avoid capital and/or operational costs that would otherwise be incurred?
- How should an energy storage system be dispatched in order to maximize the combined benefit of available services, net energy revenue, and avoided costs?

Depending on which of these questions are relevant to a particular use case, the following data and configuration inputs may apply. In general, the user can either select available data and configurations embedded in the tool or enter custom data and configuration parameters.

**Location and Operational Parameters**

- Define location (for example, transmission, distribution, or behind the meter)
- Define operational restrictions or predetermined dispatch schedules
- Service selections and dispatch objectives
- Define pass/fail objectives for desired net load required to successfully achieve avoided costs
- Select services of interest and describe energy cost/revenue schedules
- Technology Configuration (some quantities are potentially dependent on ambient temperature or state of charge)
- Specify charge/discharge capacity and ramping characteristics
- Specify energy storage capacity or duration of discharge at maximum power
- Specify charge/discharge/self discharge efficiency losses
- Characterize degradation effects of the storage system (may be in terms of cycle life, total mileage, or time)
- Quantify the cost of consumables (for example, fuel heat rate for a CAES or CT benchmark system).

**Operational Limitations**

Describe time-dependent operation limitations, which may result from statutory, technical, or operational reasons. These may be quantified in terms of min/max charge/discharge power, or required states of charge at specific times.
Economic Perspective

Select the economic perspective for which valuation is desired. Quantify key economic parameters such as discount rate or fuel prices.