transition

Moving to a smart future

Select and Dispatch Tool: Final Report

July 2023

Author: TNEI

Scottish & Southern Electricity Networks

tnei

A specialist energy consultancy

Final Report

Select and Dispatch Tool

SSEN

15330-10-R1
31 July 2023

CLIENT'S DISCRETION

Scottish & Southern Electricity Networks

tneigroup.com

Quality Assurance

TNEI Services Ltd, TNEI Africa (PTY) Ltd and TNEI Ireland Ltd operate an Integrated Management System and is registered with The British Assessment Bureau as being compliant with ISO 9001 (Quality), ISO 14001 (Environmental) and ISO 45001 (Health and Safety).

Disclaimer

This document is issued for the sole use of the Customer as detailed on the front page of this document to whom the document is addressed and who entered into a written agreement with TNEI. All other use of this document is strictly prohibited and no other person or entity is permitted to use this report unless it has otherwise been agreed in writing by TNEI. This document must be read in its entirety and statements made within may be based on assumptions or the best information available at the time of producing the document and these may be subject to material change with either actual amounts differing substantially from those used in this document or other assumptions changing significantly. TNEI hereby expressly disclaims any and all liability for the consequences of any such changes. TNEI also accept no liability or responsibility for the consequences of this document being relied upon or being used for anything other than the specific purpose for which it is intended, or containing any error or omission which is due to an error or omission in data used in the document that has been provided by a third party.

This document is protected by copyright and may only be reproduced and circulated in accordance with the Document Classification and associated conditions stipulated or referred to in this document and/or in TNEI's written agreement with the Customer. No part of this document may be disclosed in any public offering memorandum, prospectus or stock exchange listing, circular or announcement without the express and prior written consent of TNEI. A Document Classification permitting the Customer to redistribute this document shall not thereby imply that TNEI has any liability to any recipient other than the Customer.

Any information provided by third parties that is included in this report has not been independently verified by TNEI and as such TNEI accept no responsibility for its accuracy and completeness. The Customer should take appropriate steps to verify this information before placing any reliance on it.

Document Control

Revision	Status	Prepared by	Checked by	Approved by	Date
R0	FIRST ISSUE	OP, MM, SS, GM	SS, GM	GM	24/07/2023
R1	CLIENT COMMENTS	OP, MM, SS, GM	SS, GM	GM	31/07/2023

TNEI Services Ltd
Company Registration Number: 03891836 **VAT Registration Number: 239 0146 20**

Registered Address Bainbridge House 86-90 London Road Manchester M1 2PW Tel: +44 (0)161 233 4800	7th Floor West One Forth Banks Newcastle upon Tyne NE1 3PA Tel: +44 (0)191 211 1400	7th Floor 80 St. Vincent Street Glasgow G2 5UB Tel: +44 (0)141 428 3180

TNEI Ireland Ltd
Registered Address: 104 Lower Baggot Street, Dublin 2, DO2 Y940
Company Registration Number: 662195 **VAT Registration Number: 3662952IH**

Unit S12, Synergy Centre
TU Dublin Tallaght Campus
Tallaght
D24 A386
Tel: +353 (0)190 36445

TNEI Africa (Pty) Ltd
Registered: Mazars House, Rialto Rd, Grand Moorings Precinct,7441 Century City, South Africa
Company Number: 2016/088929/07

Unit 514 Tyger Lake
Niagara Rd & Tyger Falls Blvd
Bellville, Cape Town
South Africa, 7530

Executive Summary

Introduction

This report provides an overview of the Select and Dispatch (S&D) tool, developed by TNEI under project TRANSITION led by Scottish and Southern Electricity Networks (SSEN) and partnered Distribution Network Operators (DNOs).

TRANSITION is a five-year Network Innovation Competition (NIC) funded project exploring the market and technology elements of flexibility within the electricity system. By developing two IT systems (a Neutral Market Facilitation (NMF) Platform and an associated Whole System Coordinator (WSC) tool), TRANSITION enables the advertisement of flexibility needs and running of a series of flexible events including Distribution System Operator (DSO)-Procured Services and DSO-Enabled services. These IT systems operate in conjunction with a forecasting tool and a power system analysis engine to facilitate decentralised flexibility services on the distribution network. The focal point of the project is a sequence of large-scale physical trials of several flexible services on SSEN's Oxfordshire network, coordinated by the novel flexibility platform and run by SSEN. The TRANSITION project also informs and collaborates closely with the ENA Open Networks programme, within which both SSEN and ENWL are heavily active.

The aim of TNEI's scope of work for the S&D tool was to develop a decision support tool for procuring flexibility services and dispatching awarded contracts to mitigate future constraints at various time horizons. S&D was used by SSEN with various sets of off-the-shelf and bespoke software, such as the forecasting solution from Sia Partners, Opus One Solutions' NMF, and DIgSILENT's PowerFactory software for Power Systems Analysis (PSA). Together, these solutions form the end-to-end process for procuring and dispatching flexibility.

This document covers the functionality of the S&D tool, a solution overview complementing the initial design document constraints influencing the decision-making, support and development throughout trials, and valuable insights and recommendations from the project's implementation.

Solution Overview

The S&D tool is specifically designed to facilitate operational strategic decision-making in the procurement and dispatch of flexibility contracts. It evaluates bids from flexibility providers based on their market value and effectiveness in mitigating thermal constraints in the distribution network. The tool incorporates various qualifying techno-economic constraints, such as provider capacity limits, total contract value limits, and service independence, to generate an optimal solution for resolving the constraints.

The core functionality of the S&D tool is powered by an optimisation solver model developed for the project. This solver selects the best combination of contracts to meet requirements while minimising costs. The system architecture is hierarchically designed, with the solver as the central component, supported by surrounding code and functionality. The overall process flow within the S&D tool is outlined in a corresponding figure.

Learnings and Recommendations

Throughout the course of the project, several valuable lessons were learned, including:

- **Design and development:** The modular approach taken to development was found to be very helpful, although there were some rare and unforeseen issues that arose due to interactions between different bits of the tool. However, it may have been beneficial to plan from the outset for using a more agile, iterative approach to design and development, embracing all the principles and practices of DevOps through every stage.

- **Testing:** Tests were developed in parallel to each module to ensure they behaved as expected. However, some time for testing was lost due to the design of the tool taking longer than anticipated to finalise. This also affected User Acceptance Testing, with some issues only discovered very late. Moreover, less refined requirements proved to be challenging to explicitly test due to the absence of some specific details around the desired criteria.
- **Integration:** The file system interface was critical to both S&D and the PSA system and was a focus during the design phase. However, there were still some slight differences in assumptions and approaches made by developers, which led to some issues during testing. The watchdog system operated well, however there was some duplication of effort due to S&D and PSA having separate watchdogs. It would have also been beneficial to spend more time evaluating different optimisation solver options (in terms of accuracy, reliability, and computation time).
- **User Experience:** A bespoke Graphical User Interface (GUI) was developed for users of the tool, however, after design, there was relatively limited time available for end-users to test this and provide feedback, and some very good suggestions could not be incorporated. Issues and bugs within the GUI proved to be difficult to identify as this was reliant on manual use. A thorough manual testing framework, or even an automated framework, could have been useful. A further significant challenge was that, while the underlying optimisation approach was completely transparent, there was sufficient complexity that it was viewed by users as a black-box, with insufficient outputs within the tool to explain every decision recommended by the tool.
- **Ways of Working:** Both SSEN and TNEI provided multi-disciplinary teams, and roles and responsibilities were, in general, clearly defined. Some of these roles were harder to separate in practice: for example, the TNEI team ended up requiring a detailed knowledge of how PSA worked, while at the outset of the project it was expected that all that mattered was the interface. An additional role with responsibility for the entire suite of integrated tools (including S&D, PSA, and the swivel-chair role) could have proved useful – considered as a Solution Architect. The overlapping interaction of the design and development phases required upkeep of a dynamic design document, which would be the primary responsibility of the Solution Architect. This role would involve maintaining a holistic overview of the changing interfaces, assumptions and functional processes involved across the entire system.
- **Select and dispatch for decision support:** Some of the more complex practical details about how flexibility services are procured and dispatched – such as maximum dispatch durations and the inability to partially accept responses - were not reflected in the S&D tool's design. This would need further development in the future but would require a more onerous type of optimisation. In addition, more thought is required about how to deal with cases where there is insufficient flexibility availability. For the purposes of the trials, TNEI and SSEN have used the concept of "dummy flexibility", but for business-as-usual implementation of a more realistic alternative will be required.

Recommendations that could improve future development efforts include:

- Adopt a blended agile methodology across design and development.
- Define and deliver a Minimum Viable Product (MVP) prior to live deployment.
- Incorporate dedicated time for modular and systematic testing.
- Produce a software-orientated requirements set which are collaboratively built from the RTC requirements produced by SSEN and which would help define specific comparable outcomes of the functional and non-functional processes of the tool.

- Implement programmatic methodologies for interfacing systems.
- Provide automated solution interpretation for complex optimization processing.

By implementing these lessons learned, future developments can benefit from increased collaboration, faster development cycles, better testing processes, clearer requirements, improved interfacing, and enhanced understanding of complex optimisation mechanisms.

Contents

FIGURES

TABLES

1 Introduction

This report provides an overview of the Select and Dispatch (S&D) tool, developed by TNEI under project TRANSITION led by Scottish and Southern Electricity Networks (SSEN) and partnered Distribution Network Operators (DNOs).

TRANSITION is a five-year Network Innovation Competition (NIC) funded project exploring the market and technology elements of flexibility within the electricity system. By developing two IT systems (a Neutral Market Facilitation (NMF) Platform and an associated Whole System Coordinator (WSC) tool), TRANSITION enables the advertisement of flexibility needs and running of a series of flexible events including Distribution System Operator (DSO)-Procured Services and DSO-Enabled services. These IT systems operate in conjunction with a forecasting tool and a power system analysis engine to facilitate decentralised flexibility services on the distribution network. The focal point of the project is a sequence of large-scale physical trials of several flexible services on SSEN's Oxfordshire network, coordinated by the novel flexibility platform and run by SSEN. The TRANSITION project also informs and collaborates closely with the ENA Open Networks programme, within which both SSEN and ENWL are heavily active.

The aim of TNEI's scope of work for the S&D tool was to develop a decision support tool for procuring flexibility services and dispatching awarded contracts to mitigate future constraints at various time horizons. S&D was used by SSEN with various sets of off-the-shelf and bespoke software, such as the forecasting solution from Sia Partners, Opus One Solutions' NMF, and DIgSILENT's PowerFactory software for Power Systems Analysis (PSA). Together, these solutions form the end-to-end process for procuring and dispatching flexibility.

This document covers the functionality of the S&D tool, a solution overview complementing the initial design document constraints influencing the decision-making, support and development throughout trials, and valuable insights and recommendations from the project's implementation.

2 Context

This section outlines the rational for developing the S&D tool, including the overall objectives and any design constraints or limitations.

2.1 Flexibility Markets

The future landscape of power systems operations is indeed expected to face significant challenges as the energy system transitions towards net-zero. Integration of intermittent renewables, increasing commercial load peaks, and the rise of digitisation are transforming the way power systems function, leading to a more dynamic and complex environment. This new landscape requires development of more automated decision-making processes and platforms to ensure system balancing is supported. Many of these challenges can be met with flexibility however, this presents additional challenges when delivering these solutions effectively and at scale.

Integrating intermittent renewable energy sources such as wind and solar creates a power system with high variability. These sources are dependent on weather conditions, which are inherently unpredictable. This adds a layer of complexity to the task of balancing supply and demand in real-time. Moreover, rising load peaks present a challenge in Britain's power systems.

Tools such as S&D will enable power system operators to dynamically manage and optimise the dispatch of flexible resources to meet fluctuating demand and mitigate the challenges posed by intermittent renewables. By leveraging advanced algorithms and real-time data, these tools can precisely and swiftly allocate available flexible resources, such as energy storage systems or demand response programs, to address load peaks efficiently. This ensures that demand is met in a cost-effective manner, reducing the need for costly infrastructure upgrades, or relying solely on traditional power generation.

Efficient management of interconnected data sources within prescriptive algorithms which provide clear instructions for more informed and effective decision-making will be fundamental. In essence, the future operability of power systems hinges on the integration of automated decision-support platforms, within standardised flexibility market regimes.

DNOs, including SSEN, are playing a crucial role in standardising flexibility services within the power system. Recognising the importance of flexibility to accommodate the evolving energy landscape, DNOs are actively working towards establishing consistent frameworks and protocols that enable effective utilisation of flexible resources. SSEN, as one of the leading DNOs, is at the forefront of these efforts. By actively engaging with industry stakeholders, regulators, and market participants, they continue to develop standardised mechanisms. This drive is essential to streamline the procurement and deployment of flexible assets, ensuring their optimal utilisation for enhancing system resilience, reducing network constraints, and facilitating the transition towards a sustainable and reliable energy future.

The flexibility markets being trialled in this project includes four services which can be broadly defined by the network scenarios and time windows within which they resolve constraints.

Table 1: Flexibility Service Characteristics

Service	Time Window	Network Scenario
Sustain Peak Management (SPM)	3pm to 7pm	Intact
Sustain Export Peak Management (SEPM)	10am to 2pm	Intact
Secure Constraint Management (SCM)	12am to 12am	N-1 (single asset outage, due to maintenance or failure)
Dynamic Constraint Management (DCM)	12am to 12am	N-1-1 (single asset outage, followed by a second asset outage) / N-2 (outage of two assets)

Each Service except for DCM is procured in two market windows: week ahead, and day ahead. In practice, responses are gathered and process a week before the event, then a day before. Week ahead contracts will procure 80% of the identified requirement to ensure there is still liquidity at the day-ahead stage, and to account for the uncertainty within forecasts. As the event date approaches it is expected that the forecast will improve, enabling the tool to determine how much, if any, remaining flexibility requirement is needed.

The specific purpose and reasoning for these services will not be explained in this report, more details can be found in the reports issued by the TRANSITION team[1].

2.2 Key Solution Requirements

The Select and Dispatch (S&D) tool is designed to support operational decision-making in the procurement and dispatch of flexibility contracts. It gathers bids from flexibility providers, evaluating them based on their market value and efficacy in alleviating future thermal constraints in the distribution network. The S&D tool considers other qualifying techno-economic constraints, such as the maximum capacity of providers, the limits of total contract value, and service independence, which prevents service layering by providers, to produce an optimal solution for resolving the constraint. The primary high-level requirements, are as follows:

- **Data Management**: The S&D tool captures and records data inputs, such as flexibility requirements generated by Power Systems Analysis (PSA) across various network scenarios and the flexibility bids from the market which are inputted by the DSO swivel chair user[2].
- **Data Processing/Calculations**: The tool is also capable of computing and delivering crucial outputs: flexibility requests to meet future network constraints, requirements optimal combinations of flexibility responses (determined via a bounded linear programming problem) to fulfil a flexibility request, and the requisite active power dispatch of available contracts to mitigate a future constraint at different time horizons. These calculations have been designed to incorporate linearised sensitivity factors, provided by SSEN's Power Systems Analysis (PSA) platform. These are discussed later in section 3.
- **Integration**: The S&D tool is built with the capacity to interface with supporting systems, such as with SSEN's self-scripted PSA, and the Neutral Market Facilitator (NMF). The tool is designed

[1] TRANSITION project reports can be found on the project library: https://ssen-transition.com/library/
[2] The DSO swivel chair role was the main user of the S&D tool and was also responsible for the manual interface between S&D and the NMF.

to align with the NMF platform's information delivery and acceptance, a process that relies on manual data entry by the DSO swivel chair user.

- **User Interface**: The S&D tool's user interface is designed with simplicity and efficiency in mind, enabling users to smoothly navigate through the stages of the select and dispatch process. Efficient data entry supports the flow of diverse datasets, and a file-based data provision method facilitates multiple entries/exports.

These requirements constitute to forming the minimum viable product of the S&D tool and have informed the design and development of the underlying processes.

2.3 Technical and Commercial Limitations

This section discusses some of the most important constraints imposed on the design and development of the S&D tool.

2.3.1 Layering of services

Early on in the process of designing the S&D tool, SSEN highlighted that they intended for the services to be capable of being layered with each other. In principle, Sustain is intended for managing the network in its intact state, Secure is intended for managing the network in an N-1 condition (either a contingency or for maintenance), and Dynamic is intended for managing the network in an N-1-1 condition.

However, in practice, when the time came to dispatch services, SSEN intended to use any services available to resolve any network issues, irrespective of the network configuration. This means that Dynamic, although nominally for resolving issues under an N-1-1 condition, could still be used to resolve issues under intact or N-1 conditions. This may be necessary due to the lead-times associated with the different services: Sustain is dispatched at a lead-time of 12-hours and Secure is dispatched at a lead-time of 4 hours, and Dynamic has a lead-time of 30 minutes. Therefore, SSEN may wish to layer Dynamic services on top of these other services closer to real-time in the event that the forecast available at 12-hours and 4-hours led to an underestimate of the required flexibility.

As well as layering between services, it is also possible for services at different voltage levels to be substituted for each other. For example, a service provided at a lower voltage level of the system could in theory meet requirements at all of the voltage levels above that.

The logic that underpins how different services may be layered with or supported by each other was not defined within the initial requirements at the outset of the project. Instead, SSEN developed this logic in parallel with the creation of the High-Level Design and Low-Level Design of the S&D tool by TNEI. Some of the detail of this logic proved to be very nuanced and, at points, different members of the SSEN and TNEI teams had subtly but critically different understandings of the design intent. Elements of the approach were still being finalised during active development of the tool, after the Low-Level Design had been nominally finalised.

The ultimate approach reflected in the deployed version of the tool was that there is no recognition of the possible layering and substitution of services at the start of procurement, but that this is increasingly accounted for as the process is stepped-through:

- At the *request creation* stage, a unique request is made for each voltage level – not acknowledging the ability of lower voltage level services to also meet higher voltage level requirements – and for every network configuration (with a one-to-one mapping of network configurations to the types of services).

- At the *response selection* stage, the tool acknowledges the ability of services at different voltage levels to substitute for each other. This means that the total MW volume of responses selected could be lower (and potentially much lower) than the total MW of requests created if there are requirements at multiple voltage levels.
- At the *dispatch contracts* stage, the tool acknowledges the ability of different services to stand in for each other irrespective of the network configuration.

2.3.2 Partial acceptance and dispatch duration

The TNEI team decided early in the design phase to represent the underlying decision-making problems as constrained optimisation problems, due to some of the complexities involved which made it hard to frame the problem as a simple ranking exercise. For simplicity and tractability, a decision was made not to implement an Integer or Mixed Integer optimisation – these are generally much harder to solve, meaning slower computational times, different (and perhaps less reliable) solvers, etc.

As the design progressed, some aspects of the commercial design were identified which might be inconsistent with this and would ideally require a Mixed Integer formulation. One such issue was the limitation on dispatch durations within the Flexibility Services Agreement (FSA), which mean that any flexibility services provider can be dispatched for a maximum duration, which is defined with their individual FSA. Formulating this within the optimisation model would have required an integer approach, with binary variables denoting whether or not a response is being dispatched in each hour. This would be required for BAU adoption of the tool, however, SSEN advised that this could be managed by the DSO swivel chair users of the tool for the delivery of the technical trials.

Another limitation was the consideration of response capacity as a *continuous* variable – the logic of the optimisation within S&D means that costs are considered based on the ability to *partially* accept a response. For example, if a response is provided with a capacity of 5 MW, the logic of the tool would be able to consider the option of accepting just 3 MW of this. However, in practice, SSEN wished to treat selection as a binary decision: responses are either rejected entirely, or they are fully selected with the full amount. This was adopted for consistency with the NMF platform that was designed for the earlier TRANSITION trials, which S&D would need to interface with. Reflecting partial acceptance within the S&D tool would have required integer variables.

2.3.3 Linearised sensitivity factors

An acknowledged design constraint of the whole system (PSA and S&D) is in the sensitivity factors, which are a representation of a flexibility assets impact on a network asset. Upon request, the PSA system calculates sensitivity factors for a set of flexibility-providing assets based on their offered amounts, providing these for the constrained assets within the network scenario. PSA returns its representation of this sensitivity as a single signed floating-point number, which S&D interprets as the gradient of a linearised version of a power flow, with an intercept of zero. S&D is agnostic to exactly how this calculation is achieved within PSA. While the TNEI and SSEN teams expect this linearised assumption will perform well on most occasions, there is some potential drawbacks that arise from not fully considering the non-linear behaviour of power flow in reality.

The example given in Figure 1 is a hypothetical representation of the true constraint impact for increasing amounts of offered power from an asset alongside its linearised estimation. In this example the gradient of the linear estimation is the sensitivity factor used by the tool.

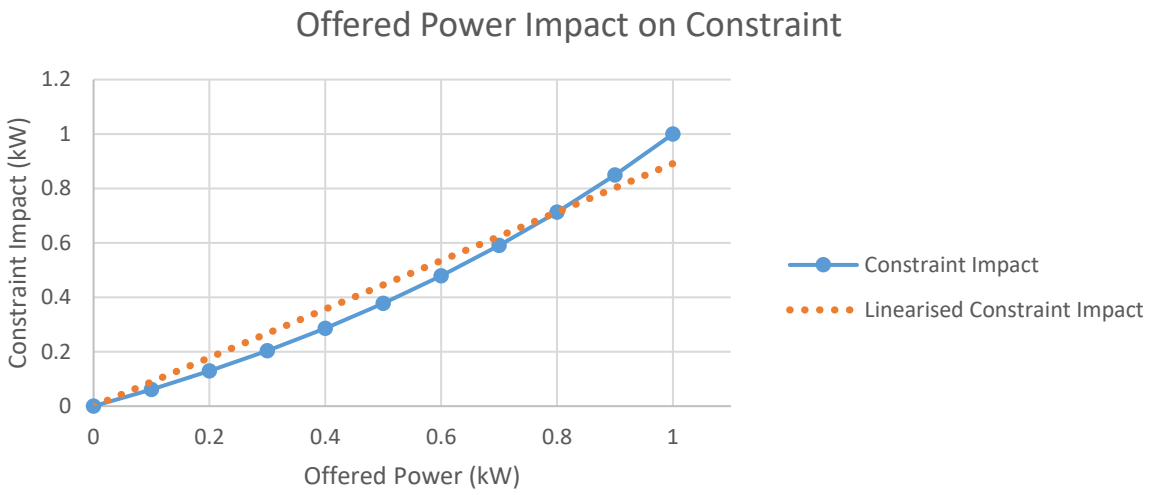

Figure 1: Example of linearised sensitivity factor representation

This linear representation was deemed sufficient within the scope of the technical trials; however, it introduces some limitations that should be considered in future work:

- Each step of the selection and dispatch process is validated by PSA, ensuring S&D will not return a solution that falls short of the requirement. Any inaccuracy introduced from the linearisation of sensitivity factors will be accounted for as part of this iterative validation process. However, these potentially avoidable iterations will likely introduce performance bottle necks and so a pragmatic trade-off between computation time and validation needs would need to be considered.
- Whilst the error between the linearised and true value of the sensitivity factor is typically small, this can impact the optimisation, which may be unable to identify a potentially better solution which exploits the complex relationships between assets.
- Furthermore, the accuracy of the linearisation is very dependent on the size of the provider's response.

In future work, the representation of sensitivity factors could be improved but this will introduce additional complexity into the process. Options might include:

- PSA returns more granular data describing the sensitivity factors in steps which may then be interpolated,
- PSA returns non-linear sensitivity factors using a form which better represents sensitivity factor response curves.

2.3.4 File-based interface between PSA and S&D

The interaction format between PSA and S&D was predetermined as part of the system specification. To achieve a clear and easy-to-follow process flow, the communication between PSA and S&D takes place through a file-based approach. With the small size of the files being shared, SSEN adopted this approach for its anticipated advantages, including the visibility of the data being transferred and the ease of sharing data with other parties for testing and bug fixing. In this communication method, the platforms exchange information using Excel files, with the contents interpreted and processed based on the file name.

Since this form of communication is a non-standard process, it requires the development of interfacing code capable of monitoring and interpreting these files. The PSA and S&D developers implemented this system separately, considering their solutions' dependence on the system architecture.

The key technical limitations that arose from this approach include:

- **Limited data transfer capabilities:** file-based communication has limitations on the flexibility and volume of data that can be effectively exchanged, posing challenges through development and testing. Excel files are explicitly limited by volume, which can lead to data loss and process corruption.
- **Lack of real-time event handling:** file-based communication relies on periodic polling and lacks the ability to handle events or notifications in real-time.
- **Dependency on file structures:** the interface relies on predefined and strict Excel file structures, making any changes to the internal structure or naming requires modifications in the interfacing code and potentially leading to compatibility issues.
- **Error handling and reliability:** file-based interfaces introduce challenges in error handling and ensuring reliable data transmission, such as file corruption, incomplete transfers, or concurrent access.
- **Synchronisation and coordination:** if implemented as multiple threads or separate programs, ensuring proper synchronisation and coordination between the systems can be overcomplicated, requiring careful management of shared resources to avoid race conditions or data inconsistencies.

2.3.5 IT Environment

The Azure Platform as a Service solutions was used by SSEN colleagues to host and deploy the PSA and S&D tools. Some limitations of this were realised during the development phase of the project. Firstly, the absence of parallel environments hindered fluent deployment. Ideally, the project would have benefited from two separate environments: staging and production, so that one could be dedicated to staging new changes and features before pushing them to the final production environment. The lack of an additional environment had implications for end users if the staged features or changes did not perform as expected. Detecting and rectifying these issues would have been more efficient within a dedicated staging environment.

Additionally, the lack of redundancy posed a risk in the event of an outage or technical issue with the environment. With no backup environment available, any disruption to the primary environment could result in service interruptions or downtime.

Fundamentally, the limitations arose because the chosen solution was deemed overly complicated for the needs of S&D and PSA. Alternative deployment approaches could have been considered, such as on-premises deployment, private cloud deployment, or containerisation and orchestration. These options would have offered more streamlined and simplified solutions, potentially mitigating the limitations encountered with the original IT environment.

By opting for a less complex integration strategy, the project could have potentially avoided the delays, reduced the pressure on developers, and provided a more reliable system environment.

3 Solution Overview

This section describes how the S&D Tool was designed and developed to meet the aims outlined in Section 2.

3.1 Design

Most primary functions of the S&D tool use an optimisation solver developed for this project. This solver selects the best set of given contracts which resolves a set of requirements whilst minimising the total cost, it is described further in Section 3.3. The solver is considered the core of the S&D tool, so the system architecture was designed hierarchically with the solver at the centre and the supporting code and functionality built around it. This is illustrated in Figure 2, which outlines the flow of processes within the S&D tool.

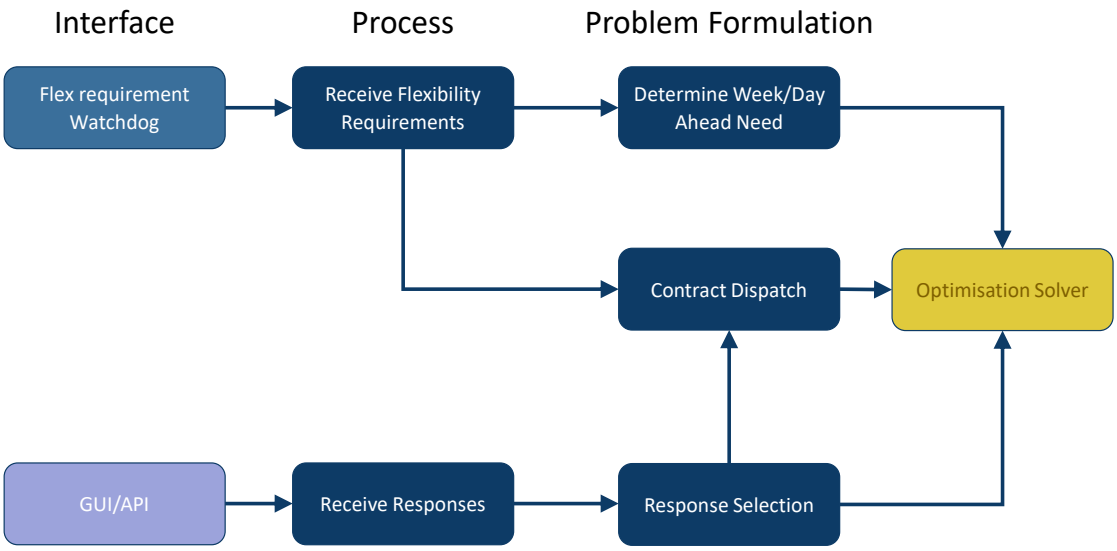

Figure 2: Overview of process flow

As the diagram shows, there are three types of problem the optimisation solver needs to resolve: determining week and day ahead requests, selection of responses, and dispatching contracts. In all cases, the solver is used sequentially to solve for each type of requirement which is covered by SPM, SEPM, SCM, then DCM. Depending on the problem being solved the previous solution when resolving one requirement type might be considered when resolving the next type.

There are three problems the solver is used to solve:

- Determining week/day ahead requests
    - This process is conducted to determine the remaining flexibility required in an auction when applicable contracts already exist and are available to resolve them partially or fully. This process will occur under two circumstances:
        - After receiving Flexibility requirements from PSA which are identified as contributing to either week ahead or day ahead request, the tool will check if any applicable contracts exist and solve for any it finds.
        - After selecting week ahead contracts. The tool will determine how much flexibility is required for the upcoming day ahead request for this requirement.

- Selection is based on all requirements with the same type – all SPM contracts for all SPM requirements.

- Selection of Responses
  - Active responses are used to solve the request they originally responded to. The responses selected in this process become contracts at the end of the auction.
  - This considers both availability price and utilisation price.
  - In case of week ahead requests, this is resolved for 80% of the maximum required capacity. Day ahead responses look to resolve the remaining, updated constraint.
  - Responses are scaled using linearised sensitivity factors, as a means of simplifying the interaction between provider and constraint.
- Dispatching of contracts
  - After confirming a final set of contracts from auction close, the tool determines a set of dispatches whenever it receives new flexibility requirements.
  - Uses all contracts to resolve SPM requirements first, then uses the selected dispatches and remaining contracts to resolve the next requirement type.

3.2 Architecture

The tool is implemented as a single python application/executable. The tool has two parallel processes, one launches the file system watchdog to monitor for flexibility requirements from PSA, and a second process manages the GUI, which is implemented as a web application, as well as the REST structured API[3], which the GUI communicates through. The application has a single SQL database which stores all information about the tool's state, and it can be accessed by all threads of system.

The application was designed so that there is a single instance of the file system watchdog which waits for only flexibility requirement files from PSA. Upon receiving a flexibility requirement, the watchdog will initiate appropriate processes which may go on to requesting and receiving sensitivity factors and response results from PSA synchronously using a new watchdog instance. This means the tool will only ever process flexibility requirement one at a time in chronological order.

However, for processing requests the flask service will create new threads as required, meaning that the tool is capable of handling as many client requests as the hosting computer can handle. Requests from flask will trigger the contract selection process. This process will also require using a separate watchdog instance which requests and receives sensitivity factors and response results synchronously.

The web-application based GUI was implemented using React, a modern front-end JavaScript library which programmatically generates HTML markup, styling, and logic within code. This allowed for fast prototyping and development of the final GUI using a mixture of prefabricated and custom-built components. The GUI uses the server's API to send and receive JSON structured data. It is pre-compiled with each release of the tool and distributed to the client's internet browser through the flask service when they request a page.

Figure 3 describes this architecture. The essential components of the tool are categorised by colour, and that key is maintained throughout this section.

[3] A REST API is a networking API which conforms to the REST architecture. It is a stateless and uniform interface which communicates in standardised data formats (JSON, XML, HTTP). https://en.wikipedia.org/wiki/Representational_state_transfer

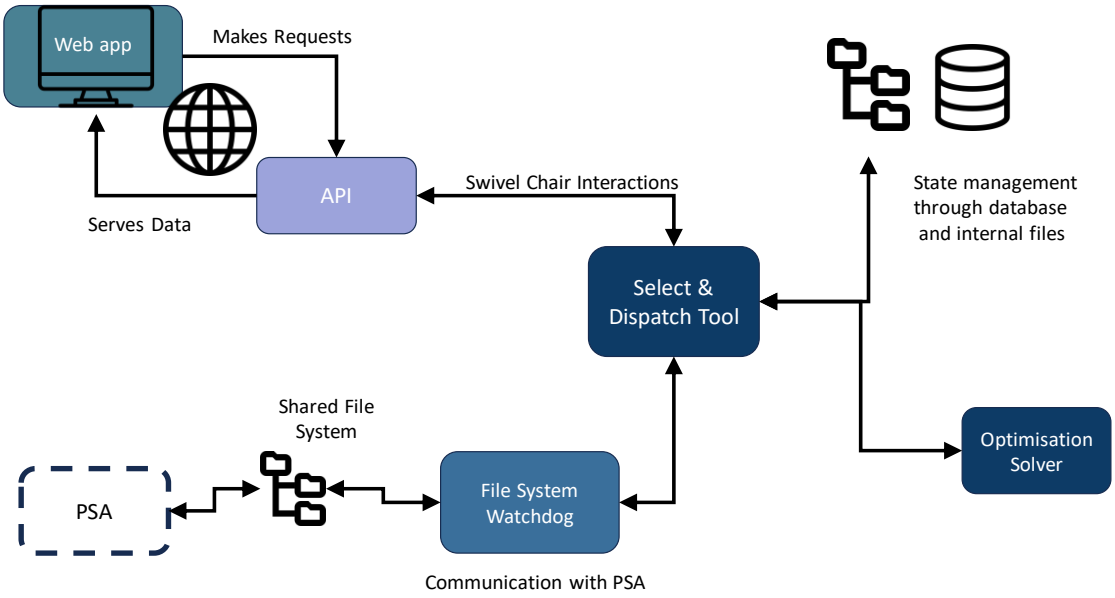

Figure 3: Block Diagram of S&D tool structure

3.3 Optimisation Solver

In this section, the key functional processes outlined in Section 3.1 are described in detail.

3.3.1 Calculating flexibility requests

The first stage in the S&D process is driven by input data received from PSA. This data describes the anticipated thermal overloading of network branch ratings, which has been formulated based on demand forecasting data sourced from the SIA platform and topological data retrieved from NeRDA. A simplified implementation of the optimisation solver is used to calculate necessary flexibility to appease these future constraints.

For each individual network branch, whether a line, cable or a transformer, flexibility requirements are assembled according to the relevant network scenario. (Network scenarios are indicative of the operational configuration of the power network at a given point in time.) Requirements are subsequently categorised into distinct time windows which correspond to a particular flexibility service. Formulation of flexibility requests, within each service and location grouping, simply considers the maximum requirement throughout the window. This maximum is scaled accordingly, accounting for provider reliability. This is illustrated in Figure 4 below.

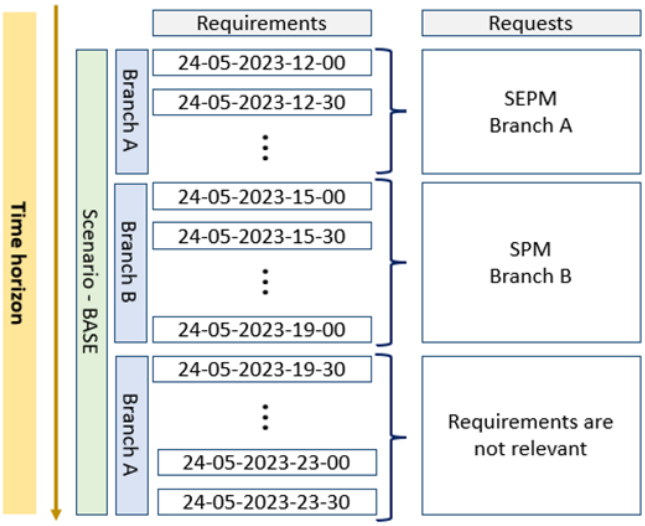

Figure 4: Translation of requirements to requests

3.3.2 Selecting and dispatching flexible contracts through an optimisation solver

Linear optimisation (or linear programming) is a mathematical method used to find the best outcome in a mathematical model where the objective function and constraints are represented by linear relationships. The goal of this optimisation process within the S&D tool is to find the most cost-effective combination of flexibility contracts that will meet a given flexibility requirement. The objective function in this scenario is therefore the minimisation of the total cost of procured flexibility contracts. However, this selection process must also respect several constraints, each representing a technical or commercial limitation within the system.

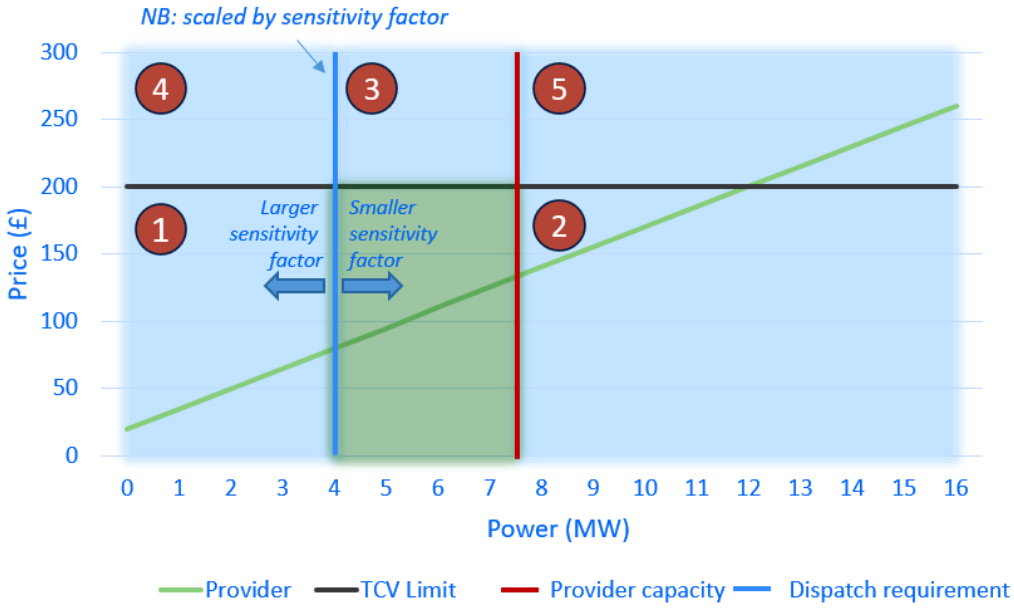

Figure 5: Optimisation solution space and infeasible regions

Figure 5 illustrates a simplified example of the problem. The graph demonstrates the interactions between different constraints and the objective function, shown in green. Considering a single flexibility provider with a fixed availability price (y-intercept) and variable utilisation cost (slope), we can see these dispatch options pass through both feasible and infeasible regions of the decision space. Blue regions indicate that these options would violate one or more of the constraints.

The minimum dispatch is also dependent on the linearised sensitivity factors of the contributing provider. The sensitivity factor is inversely proportional to the degree of radial splitting between the provider and the network constraint. These constraints are further described Figure 6. .

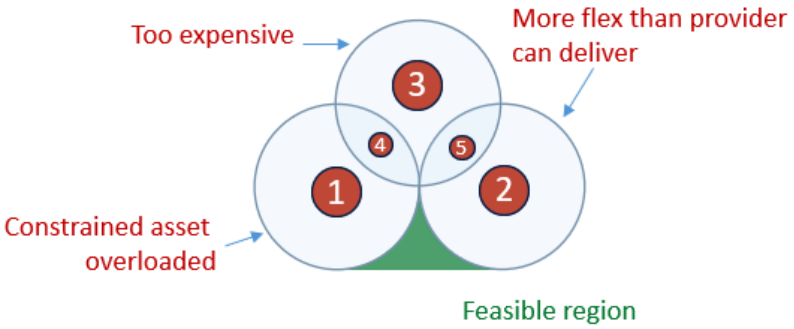

Figure 6: Interaction of constraints

In practice, these constraints represent physical limits (for instance, an asset cannot exceed its maximum capacity without risking failure), commercial limits (a service has a regulatory-imposed maximum total contract value), or system needs (such as a minimum delivery to mitigate a constraint). By considering these constraints within the linear optimisation problem, the S&D tool can find the optimal selection and dispatch of flexibility contracts that satisfy the requirement while also adhering to all specified constraints.

The highlighted green region contains the optimal solution for this problem, but other options exist in this space that satisfy the specified constraints. In this example, we simply scan across the objective function, within the solution space, to find the optimal solution, shown as the blue point in Figure 7.

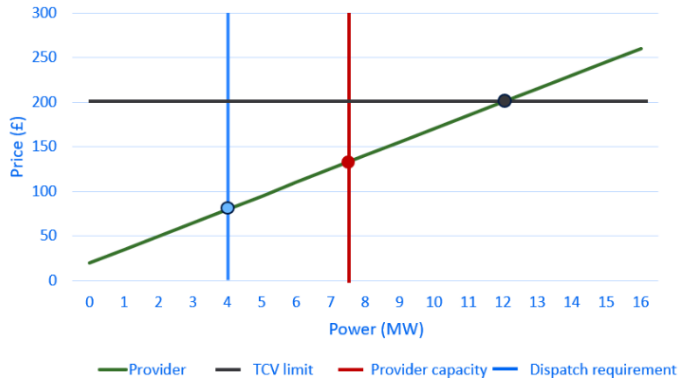

Figure 7: Optimal solution

It is important to note the optimal solutions identified may vary slightly from what is technically optimal when considering the true complex interactions of active power flexibility dispatch, particularly due to the complex non-linear apparent power sensitivities to active power injection. Furthermore, as discussed earlier, the constraints are more complex than indicated in this example. The constraint regions in the diagram are parametric, indexed by several sets which characterise the mathematical problem. For example, the dispatch requirement for a target constraint is parameterised by network scenario and date/time.

This relatively simple example is used only to demonstrate the inner workings of the solver. The real-world problem will exhibit multiple dimensions of complexity and interactions as more responses and requirements are added to the problem. It is these complex interactions which a simpler ranking approach would be unable to resolve and, therefore, necessitates the use of the implemented mathematical optimiser.

Implementation of optimisation algorithms within the space of continuous feature sets presents some data handling issues. From the experience of this project, the most prominent cases arise due to rounding. Rounding is typically used to simplify the optimisation problem, however, can lead to unintended consequences such as omitting solutions from the feasible region, or negligible responses being considered towards a solution. These cases should be handled with care when handling non-integer optimisation problems.

3.4 Back-end Interface

The S&D tool requires an external interface with only the PSA tool. As summarised in section 2.3.4, PSA and S&D communicate by monitoring a shared file system. Both tools implemented their own software for reading, writing, and monitoring this space, and this is referred to as a filesystem watchdog. The specification of this interface was planned during early development and adapted collaboratively if required. Both systems send and receive Microsoft Excel files, and the filename indicates important information such as the type of request, the relative analysis time of the request, as well as meta-information including the origin of file (S&D/PSA) and the process by which it was created (manual or automatic – MAN/AUT). The contents of the file varied depending on the type of file being delivered.

Through this interface, S&D and PSA are engaged in two related but unconnected data flows; PSA sending S&D flexibility requirements, and S&D requesting Sensitivity factor and candidate selection validation calculations from PSA.

PSA sends requirements at a predefined time frequency. S&D's watchdog monitors the file system for these files and then passes them to the tool's processing function which interprets and produces requests based on the requirements.

The S&D tool will communicate with PSA to obtain sensitivity factors and when attempting to resolve constraints with the available responses and contracts. Both steps require PSA to perform a power flow analysis and return results. S&D will send all the relevant request files and then wait for a response from PSA. This loop continues until S&D has completed its process by resolving the constraint, or until 10 iterations have been reached in which case the 'best' selection is taken.

3.5 User Interface

This section provides a general overview of the S&D tool GUI. The user interface has a home page, main menu, and a page for each stage in the Select & Dispatch process (publish requests, select responses, and dispatch), and also contains information relevant to the current market gate. On each page data is presented to the user in a table view, which can be filtered and queried as required. Finally, an explore data page allows users to access and query all aspects of the tool.

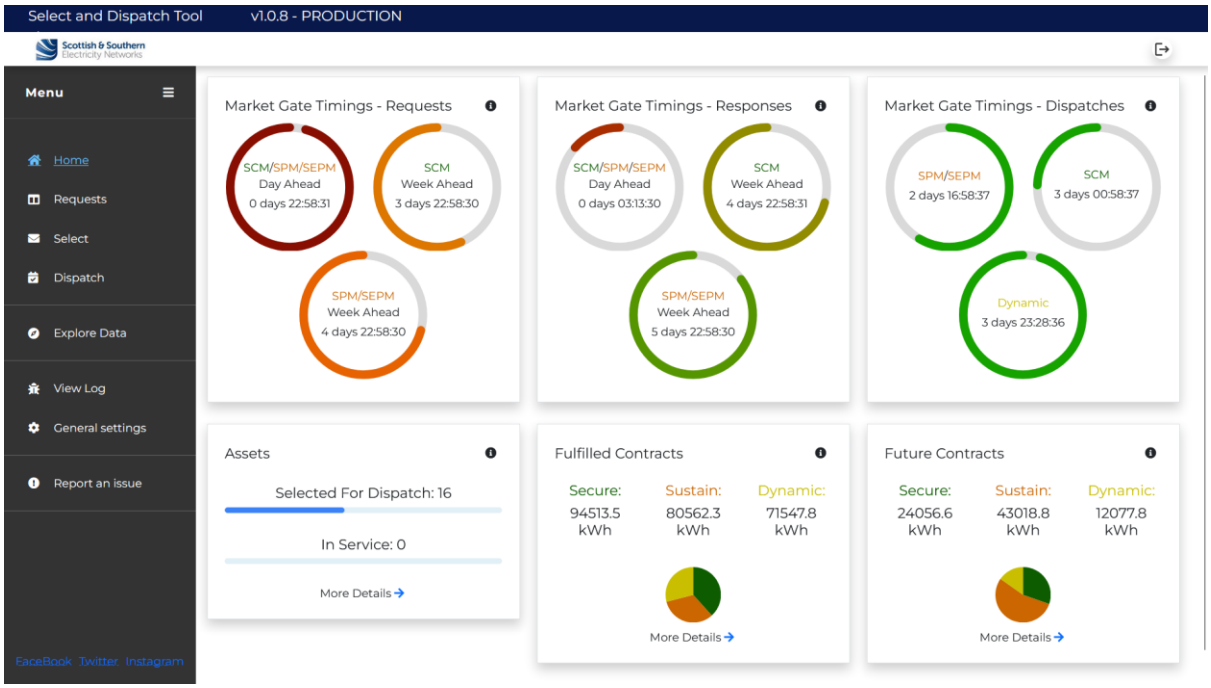

Figure 8: Homepage of tool

The S&D tool home page is shown in Figure 8 and provides an overview of the tools current state and provides market information required by users to co-ordinate the market and dispatches. It also provides a summary of the total energy which has and will be provided by current and future contracts, grouped into cards by stage and market timing. Each card also includes a help button on the top left. Clicking this shows a description of the card's contents and how they should be interpreted.

The "Requests" page shown in Figure 9 presents all upcoming constraints which have been identified by PSA in the "Upcoming Constraints" tab, and the "Upcoming Requests" tab describes every request which has been generated for the constraints. The export requests button downloads a truncated and formatted version of the upcoming requests, designed specifically for the DSO swivel chair user who is transferring requests to a neutral market facilitator (NMF) to create an auction.

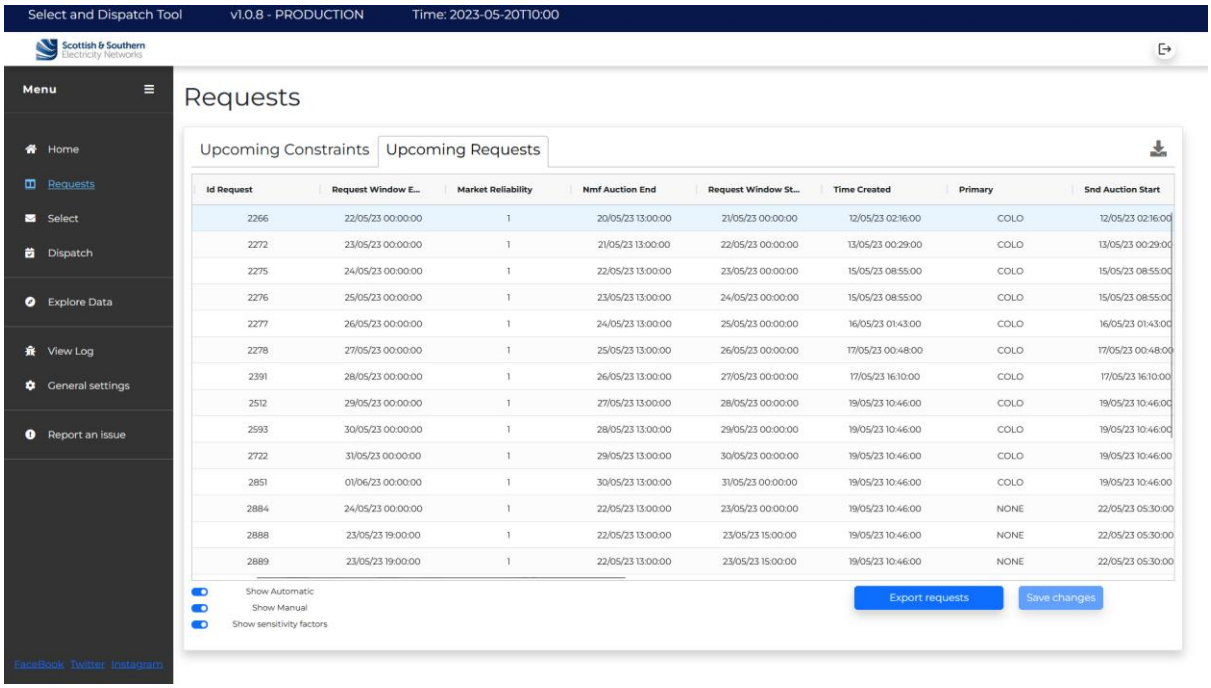

Figure 9: Requests window of tool

The "Select" page initially presents the user with all requests whose auction should now be active, with pending and accepted responses presented in further tabs. From here, the user can upload a .CSV[4] file of all responses which have been collected by the NMF following the previous stage. For the technical trials, this was a manual operation performed by the DSO swivel chair user. In a full BAU system, this process of passing data between systems should be automated to both minimise user error and administration time.

The pending responses tab describes all responses which have been uploaded, regardless of whether they have been accepted or rejected. From this window, the user may adjust these responses if necessary, and the tool would recalculate and adjust the set of selected responses. The "accepted responses" shows only the responses which have been accepted by the tool and will be promoted to contracts once the auction ends. The user can also export the decisions made for each response as a CSV so that the DSO swivel chair user can feedback to responders.

[4] The format of the CSV file was agreed and provided in the Low-Level Design (LLD) document of this tool. Further details on the LLD can be requested via FNP.PMO@sse.com.

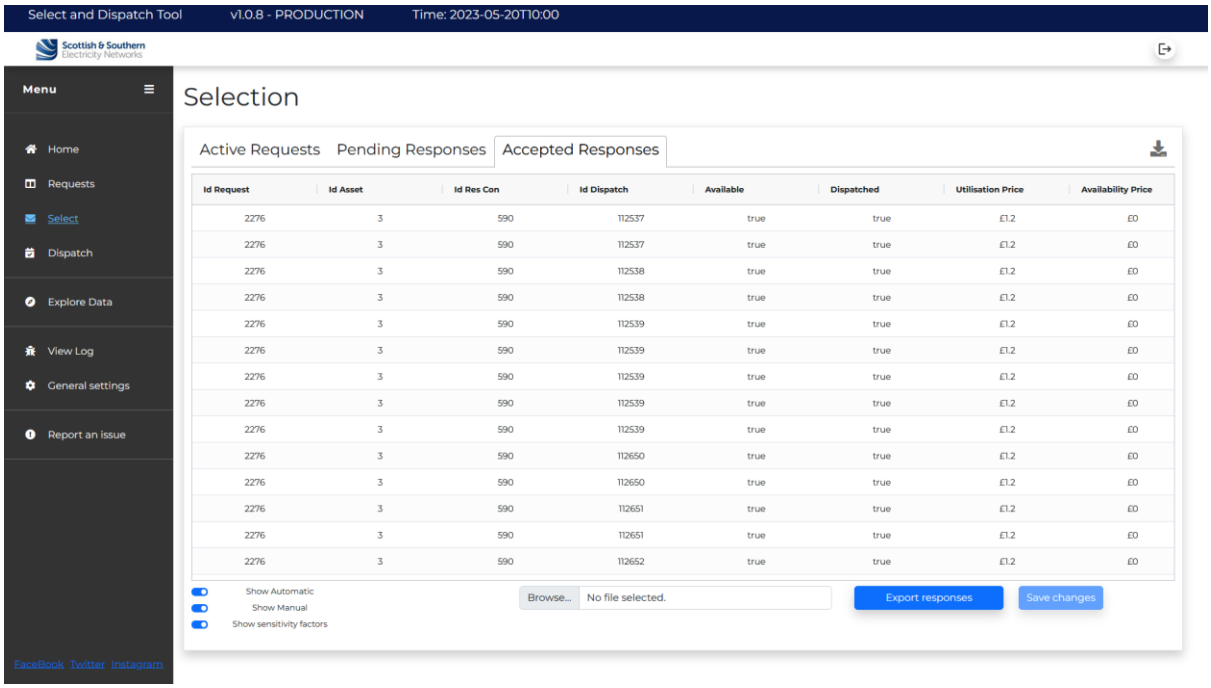

Figure 10: Select page of tool.

The "Dispatch" page shown in Figure 10 describes every contract which has been selected by the tool for dispatch, along with the specific amounts required by each responder at each time step to resolve the constraint. The "Export dispatches" button downloads a compiled and formatted CSV of all dispatches.

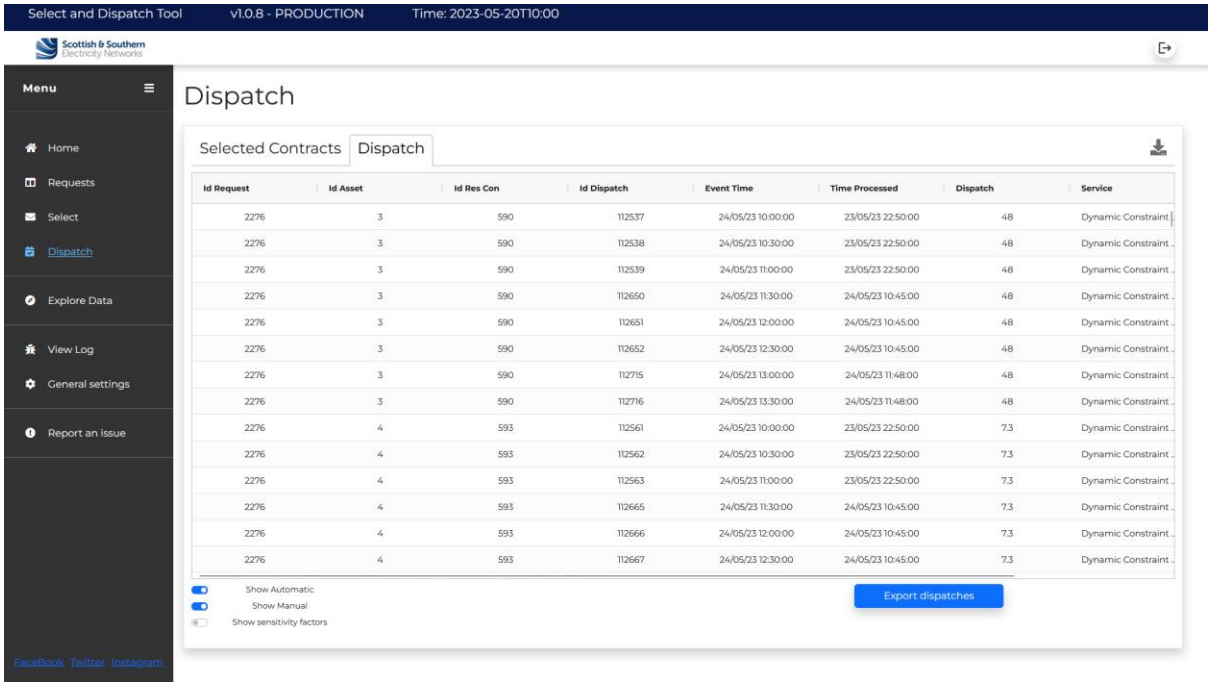

Figure 11: Dispatch page of tool.

Each table also includes a 'sensitivity factors' switch in the bottom left-hand corner. This adds the most recently calculated sensitivity factor (SF) to the data displayed on screen (which is relevant for the decision-making process. i.e., the SF which the response has against the constraint is offered flexibility to resolve or the SF which the contract had against the constraints its dispatching against). This feature was added to improve the transparency and aid the understanding of the tool's decisions.

The explore data page allows the user to navigate the full database of the tool using the same table interface as the other pages.

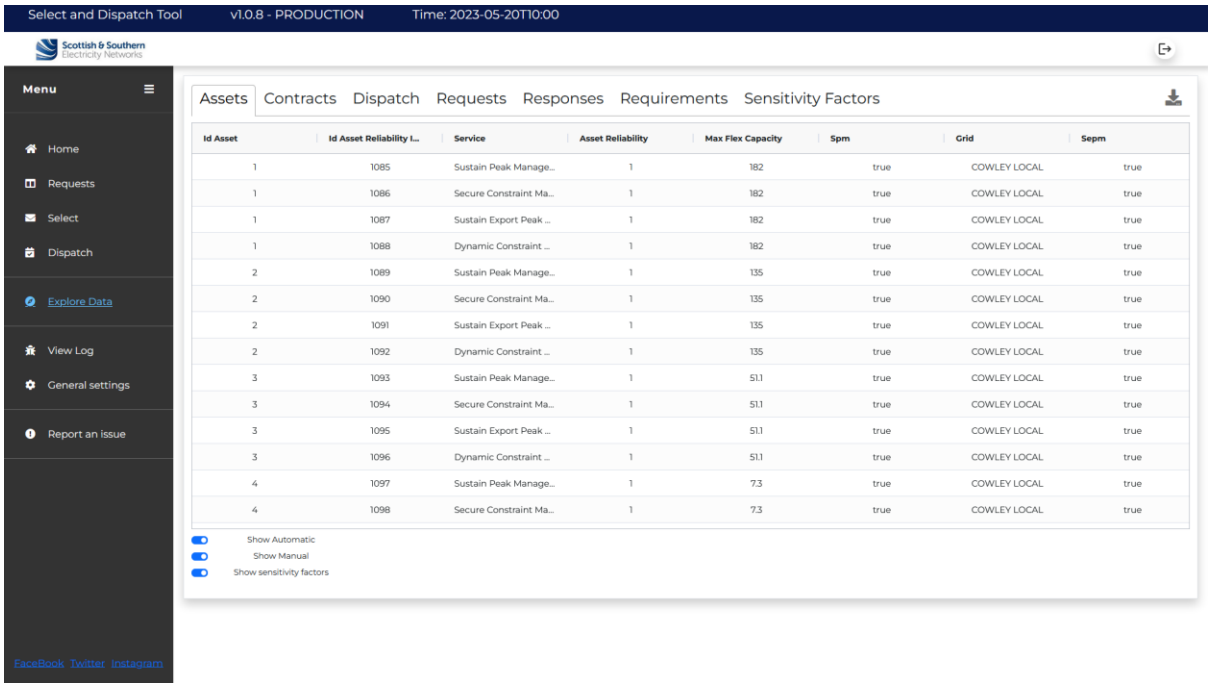

Figure 12: Explore data page

The “General Settings” window shown in Figure 13lets the user modify meta-information about the operation of the tool, e.g., market timings and total contract value (TCV).

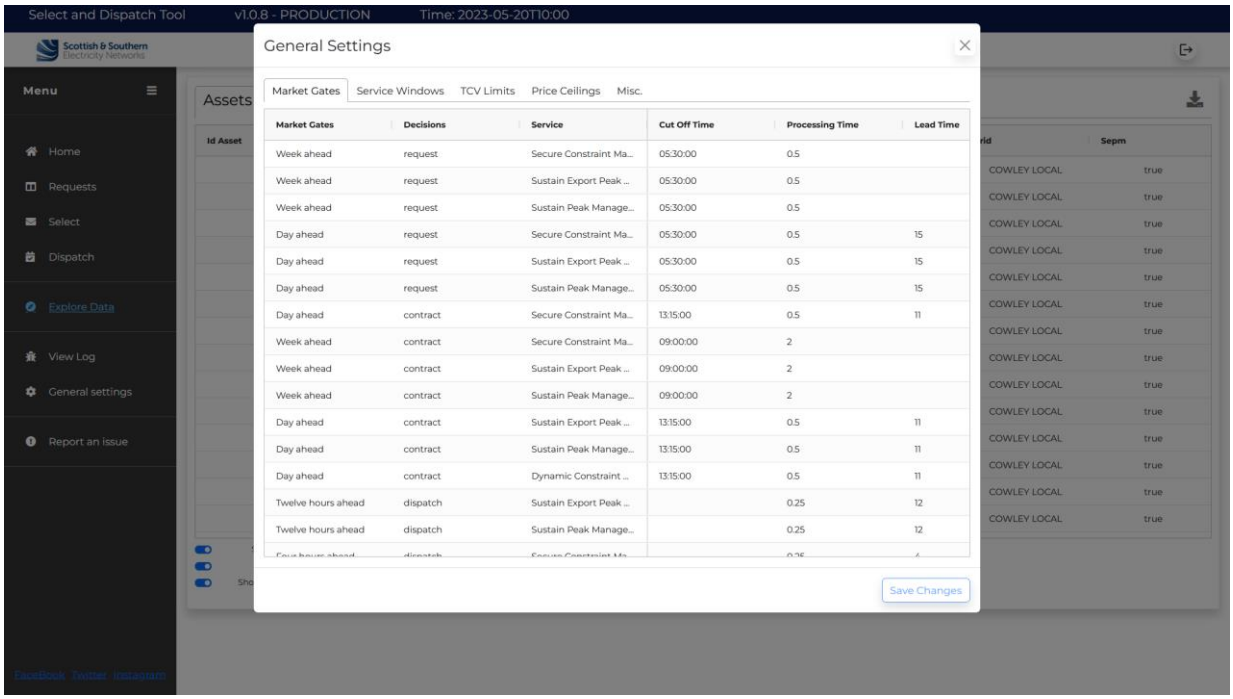

Figure 13: General settings window

Several additional features were also added to the tool to help improve the user experience. Two to note are the context view window, and the live log.

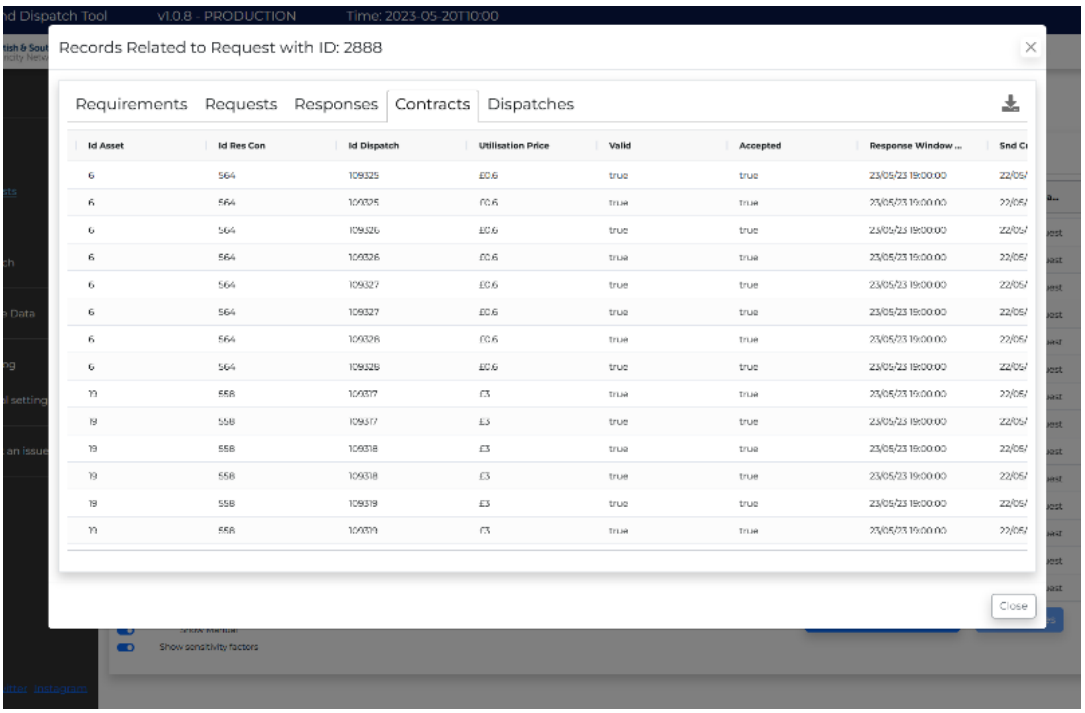

Figure 14: Context window view for all data related to request ID 2888

At any point, the user may open a sub-view window of all records related to single request, as shown in Figure 14. This feature was added during the trials, so the DSO swivel chair user can view all data for a full process, from creating requirements to dispatches, in a single window.

Finally, a live log is accessible from the tool's main menu. A user can access this at any time to see what the tool is currently processing, and to check if any errors have occurred due to any recent inputs or changes. This proved particularly useful for identifying issues quickly.

3.6 Security

During the trials, the server and all its data were hosted on, and only available through, an Azure virtual desktop environment which was accessed through a virtual private network (VPN). This environment included extra security levels, i.e., required 2 factor authentication, and only a small number of the team were granted access. For the S&D tool itself, a basic login system was implemented to manage user privileges.

These layers of security were considered sufficient to keep the trial secure. Therefore, it was deemed unnecessary to include further security considerations in the implementation of the tool.

A business as usual (BAU) solution would require some additional layers of security i.e., extra authentication layers, and database encryption (salting/hashing at the minimum), for all modes of accessing the tool, i.e., the user interface and all internal/external s. To an end user this would appear as a full login system and token management. Adding these in would provide additional security in the event an unauthorised person gains access to the secure environment.

4 Results of S&D Testing

This section outlines the different types of tests carried out on the S&D tool by the development team. These can be split into two types – automated and manual.

Automated testing of the tool's backend code was carried out in two key stages: (i) as part of the requirements testing, confirming that the tool meets the requirements outlined by SSEN, as well as (ii) functional testing, which helped the development team ensure that code was operating as expected during development. More detail on these tests is included below.

Additionally, manual testing of the frontend GUI was also performed. This did not follow a formal process due to the relatively simple nature of the GUI, and instead were designed as a "sense check" that visualisations, icons, etc looked and performed as intended.

4.1 Requirement Testing

These tests are designed to ensure that any tool or software developed meets the needs as outlined by SSEN. For each requirement, the development team created automated tests which performed the corresponding process and then checked that the tool's behaviour matched the required outcome of that process. These tests are entirely written in code and are standalone tests – meaning they must also simulate the expected behaviour of other integrated or interfaced processes, for example the interface between S&D and PSA. Additionally, these tests are also used to validate the tool's transferability to the Azure desktop environment by running them on both our local environment and azure environment.

Ensuring the tool passes each of these requirement tests confirms that it is meeting the minimum specified requirements. For most cases, a single, considered test is written per requirement or set of closely related requirements. These tests are performed exclusively for typical system behaviour. Therefore, they did not cover the edge cases which might come about during live trials and usage. Capturing these cases during development would have proved very difficult to anticipate, but it was expected that the more common of these cases would be captured during UAT.

As the project developed, the scope and priority of these requirements tests were adjusted to coincide with updates to the tool's specification and any changes in the tool's processes.

These tests were used throughout the development phase of the project and during maintenance to ensure any changes did not have a detrimental impact on the tool meeting its specification.

For final confirmation of the tool's ability to meet the requirements, the full requirements testing suite was applied to the final iteration of the tool's code. The S&D tool passed all tests, and confirmation of this is shown in the test report summaries included in Requirement testing results.

4.2 Functional testing

The development of the S&D tool was carried out in conjunction with unit tests, following the principles of test-driven development. Test-driven development involves writing tests for required functions before or concurrently with writing the corresponding code, thereby guiding the implementation process, and ensuring the code functions as intended.

The decision to implement these tests was based on the significance and sensitivity of the individual functions within the tool. To optimise efficiency and avoid redundant work, certain requirement tests were deemed sufficient to evaluate specific functional aspects of the code. For instance, the mathematical solver and its supporting code were rigorously tested by incorporating a diverse set of requirements and responses in these tests.

In cases where a sub-component of the tool held critical importance but was not adequately covered by requirement tests, additional basic tests were meticulously designed and implemented. These tests comprehensively covered each function, context, and scenario related to the PSA interfacing and communication through the API. The scope of these tests encompassed typical scenarios, edge cases, and error handling, thereby addressing potential challenges that might arise during real-world trials.

This testing approach complements requirement testing, which are specifically for demonstrating satisfaction of a predefined specification of requirements. Functional testing serves as a continuous feedback loop to support development by allowing for easier identification of deviations from the intended functionality.

4.3 Testing and development during UAT and trial blocks

As part of the efforts to adopt continuous integration processes, automated tests were performed whenever any changes were made the code base. These tests were conducted on the Azure DevOps service by the developers making the changes on their local machines.

During UAT and the live trials, several bugs and essential enhancements were uncovered. When these were identified, an issue was created on the Azure DevOps platform and a member of the TNEI team would begin working towards resolving that issue. Once completed, and after review, the solution would then be pushed to a staging version of the S&D tool for testing. These logs are compiled in Appendix A – S&D Change Logs S&D Change Logs. Change logs covering these updates and any potential impacts were then issued to the TRANSITION team.

A large proportion of these changes related to the GUI, but there were also some changes to the behaviour of the tool's main processes. Notably the automated creation and handling of dynamic requests, interfaces with PSA when points of ambiguity were identified, and improvements to the efficiency of the solver's iteration process[5].

[5] These changes are described in the amended LLD document, a copy of which is available upon request.

5 Learnings and Recommendations

The table overleaf reflects points for learning and opportunities to consider in future, our and their key insights and takeaways gained through each stage. We have grouped these into the following categories:

- **Design and development:** The modular approach taken to development was found to be very helpful, although there were some rare and unforeseen issues that arose due to interactions between different bits of the tool. However, it may have been beneficial to plan from the outset for using a more agile, iterative approach to design and development, embracing all the principles and practices of DevOps through every stage.
- **Testing:** Tests were developed in parallel to each module to ensure they behaved as expected. However, some time for testing was lost due to the design of the tool taking longer than anticipated to finalise. This also affected User Acceptance Testing, with some issues only discovered very late. Moreover, less refined requirements proved to be challenging to explicitly test due to the absence of some specific details around the desired criteria.
- **Integration:** The file system interface was critical to both S&D and the PSA system and was a focus during the design phase. However, there were still some slight differences in assumptions and approaches made by developers, which led to some issues during testing. The watchdog system operated well, however there was some duplication of effort due to S&D and PSA having separate watchdogs. It would have also been beneficial to spend more time evaluating different optimisation solver options (in terms of accuracy, reliability, and computation time).
- **User Experience:** A bespoke Graphical User Interface (GUI) was developed for users of the tool, however, after design, there was relatively limited time available for end-users to test this and provide feedback, and some very good suggestions could not be incorporated. Issues and bugs within the GUI proved to be difficult to identify as this was reliant on manual use. A thorough manual testing framework, or even an automated framework, could have been useful. A further significant challenge was that, while the underlying optimisation approach was completely transparent, there was sufficient complexity that it was viewed by users as a black-box, with insufficient outputs within the tool to explain every decision recommended by the tool.
- **Ways of Working:** Both SSEN and TNEI provided multi-disciplinary teams, and roles and responsibilities were, in general, clearly defined. Some of these roles were harder to separate in practice: for example, the TNEI team ended up requiring a detailed knowledge of how PSA worked, while at the outset of the project it was expected that all that mattered was the interface. An additional role with responsibility for the entire suite of integrated tools (including S&D, PSA, and the swivel-chair role) could have proved useful – considered as a Solution Architect. The overlapping interaction of the design and development phases required upkeep of a dynamic design document, which would be the primary responsibility of the Solution Architect. This role would involve maintaining a holistic overview of the changing interfaces, assumptions and functional processes involved across the entire system.
- **Select and dispatch for decision support:** Some of the more complex practical details about how flexibility services are procured and dispatched – such as maximum dispatch durations and the inability to partially accept responses - were not reflected in the S&D tool's design. This would need further development in the future but would require a more onerous type of optimisation. In addition, more thought is required about how to deal with cases where there is insufficient flexibility availability. For the purposes of the trials, TNEI and SSEN have used

the concept of “dummy flexibility”, but for business-as-usual implementation a more realistic alternative will be required.

Table 2: Learnings and future opportunities

Learnings and impacts	
Experience	**Opportunity**
Section 1: Design and Development This summarises learning about the design and development of the Select & Dispatch tool.	
Modular approach	
• The team have taken a modular approach to development which ensures all components of the S&D system are separated and their interactions are fully considered. This meant that the team could carry out isolated and concurrent development and testing of the various functional modules.	• The initial layout of this modular approach did not fully consider all requirements of the tool prior to starting development – although, this full consideration may have been impossible in practice. This resulted in some moving and manipulating code to integrate unforeseen features which could have been avoided. However, this initial layout made making these subsequent changes far easier. • Experience from this project highlights that concurrent realisation of user level requirements, development, testing and deployment, *without* finalising interfaces results in inefficient development of the tool due to the need to rewrite and re-implement code. • Furthermore, attempting to thoroughly test individual modules before this solidified specification was confirmed resulted in stagnated development which could have been avoided.
Limited feedback loop from a lack of an agile approach to design and development	
• The feedback within the project provided an opportunity to improve the team's understanding of user requirements. Through iterative development and close collaboration, the project team gained valuable insights into the specific needs	• Building on the improved understanding of requirements, future projects can leverage this knowledge to iteratively refine and clarify the user needs. This can involve continuous communication and collaboration with stakeholders to

Learnings and impacts	
Experience	**Opportunity**
and expectations of the users, resulting in a clearer understanding of the project requirements. • The sequential design and development process allowed the team to uncover and address system limitations and constraints. By encountering these limitations during development, the team gained a deeper understanding of the technical boundaries and challenges, enabling them to devise appropriate solutions.	ensure that the evolving requirements are effectively incorporated into the ongoing development process. • It is possible to enhance user satisfaction by incorporating a deeper understanding of user requirements and addressing any system limitations, leading to an improved user experience. • Aiming to deliver a minimal working system – even if it has incomplete features – at an earlier stage of the development phase would enable early feedback. Experience has shown that this is vital to effectively identify and resolve issues, thereby making any required changes easier to manage.
DevOps as a set of principles and practices	
• DevOps is a set of principles and practices aimed at fostering collaboration, communication, and alignment between development (Dev) and operations (Ops) teams to enable efficient software development, deployment, and operation. The team have been focussing on bringing in these practices into our development process. • The Azure DevOps environment provides a set of tools which are required to follow these practices. The teams previous experience in similar tools, e.g., GitHub, made it very easy to adapt to Azure. • Agile methodologies and DevOps practices enable organisations to adapt to evolving requirements and changes in the business environment. By embracing iterative development, continuous integration, and delivery practices, teams can respond quickly to feedback and market needs, ensuring that the software aligns closely with user expectations, even if that expectation changes.	• The adoption of agile/DevOps paradigms was limited to the development phase of the tool when it should have been adopted from the outset of the project as a part of projects planning and delivery timelines. Doing this could have better navigated the essential changes which were identified during UAT and live trials.

Learnings and impacts	
Experience	**Opportunity**
• By embracing the core elements of the DevOps paradigms, the team was able to work with a more controlled and predictable environment which allowed us to navigate and resolve issues much faster than if we had decided to work in a style which did not emphasise adaptability. • This most likely played a significant role in reducing disruptions and delays.	
Unexpected System interactions in Parallel Processing Environment	
• The multi-processing system allowed the tool to process requirements and responses simultaneously, ensuring there was no downtime for users. • The team also devised a solution for exception handling within the tool's backend code. This meant the error was reported, but then the tool 'rolls back' the database to the state before the error. This technique was used to reduce boilerplate code, improve readability, and allows users to amend any errors before changes become permanent.	• Whilst designing this approach the team did not consider the rare scenario where the database was being written to by both processes simultaneously. Due to the rolling back system, when a second process attempted to write to the database during this procedure a fatal exception occurred, and the process crashed. • The problem arose from interactions between systems which were thought to be independent from each other. This is a learning opportunity for the software team to deepen their understanding of the technology and take more precaution when considering interactions like these. • Fixing this problem during the trials would have required significant changes to the code, and due to the intensive resource to do this a pragmatic solution was adopted which where the issue was managed by resetting the program on the rare occasion the issue occurred.

Learnings and impacts	
Experience	**Opportunity**
Section 2: Testing This summarises learning about testing of the Select & Dispatch tool.	
Dedicated time for modular and systematic testing	
• The modular approach described before enabled specified and isolated testing of the system. Tests were developed alongside the implementation of each module, ensuring their behaviour was complete and accurate.	• Overlapping testing with concurrent development and user feedback increased the maintenance requirements for developers, necessitating higher support and debugging resource. This increases the risk of errors and impacts the wider timelines for deployment. Any change to a requirement would result in needing to change the test, nullifying the previous work. • Compressed development and testing timelines were observed from the extended design period however, the original plan for delivery of the project did not involve concrete timelines for dedicated testing of the system as a whole. We recognise that regular modular testing is optimal for ensuring a component of the wider tool can be integrated, however full integration tests (as found during UAT and live trials) is essential in identifying any shortcomings in the system or specification.
User acceptance testing - execution	
• User acceptance testing (UAT) is intended as a dedicated testing phase where users evaluate the software to determine it meets their requirements and expectations. This is designed as the final check to ensure the software works as intended. Despite compressed timelines for the project, time was made to ensure some form of user acceptance testing was carried out which helped identify and fix system issues before beginning live trials.	• The UAT phase posed specific challenges because of conflicting timelines and the parallel development of interfacing systems. Some issues were then discovered quite late in the project, approaching the end of dedicated development time, towards the beginning of UAT testing. This would be resolved by beginning UAT much earlier by adopting the methods discussed previously.

Learnings and impacts	
Experience	**Opportunity**
	• The overlap between UAT and the live trials resulted in a much greater complexity when integrating PSA and S&D systems, and any fixes to issues which arose were difficult to deploy as thorough testing is usually recommended prior to any update being integrated into a live tool. Both the overlap between UAT and the IT infrastructure added complexities into this process. Ideally, UAT should be given a much greater time frame, beginning before the complete development of the tool.
Requirement definition	
• An extended period of time was taken to define and clarify the requirements of the system, enabling a clearer view of how the system will operate and work. The team were able to implement full test suites which check the system's ability to implement each of the tool's defined requirements.	• From a software development perspective, the mapping of requirements to a definitive, functional, and unambiguous tests could have been executed more effectively. The defined requirements of the S&D tool contained non-functional elements which, while useful at a higher level, did not cover all of the nuance required for software implementation. This approach cannot capture or discuss the complexities which come with implementation. • Additionally, the purely sequential approach to design and implementation further exacerbated the issue, as several key questions were only uncovered during implementation phase which could have been uncovered earlier if design and development stages were agile and integrated. • Any organisation - including TNEI - developing bespoke applications within or without the power sector should engage in more 'software forward' high level design processes typically associated with agile development, i.e., defining user stories and clear specifications of tool behaviours and interfaces.

Learnings and impacts	
Experience	**Opportunity**
Section 3: Integration This summarises learning about the integration of the Select & Dispatch tool within SSEN's IT environment, and with other systems (particularly the PSA).	
Transparency of data exchanges	
• The file-system interface served as a dedicated directory for data exchanges between interfacing systems, PSA and S&D. This was envisaged as a means of ensuring the communications process was transparent for non-developers i.e., users and operators of S&D and PSA, as well as providing a record of all data and logs.	• As the project developed past the design phase, the underlying dependencies of the file system interface quickly became complex and more difficult for non-developers to review and digest. • Some essential specifications of the contents of these data files i.e., the uniqueness of IDs, was assumed rather than discussed. This discrepancy was revealed during end-to-end tests and required changes to both the interpretation and parsing of data. • The exact actions and methods of engaging with this data exchange system was not fully defined either, resulting in both systems interpreting and modifying the system differently. Which has the potential of causing confusion from an outside perspective. This is an aspect of such a system which needs to be discussed. • Developing a system which serves two conflicting purposes (efficient data transfer, and transparent depiction of system processing) proved challenging for developers and data-analysts.

Learnings and impacts	
Experience	**Opportunity**
Watchdog systems	
• The watchdog system, within S&D, was responsible for tracking, reading, and distributing files between interfacing systems i.e., with PSA. This acted as the channel for communication and as a trigger to begin processing requirements, responses, sensitivity factors etc. • Overall, the watchdog systems work reliably once adequate testing and validation was completed.	• Generally, the development and maintenance overhead of a file-based interface like the Watchdog system outweighed the benefit of transparency of data exchanges. • Differing approaches to system architecture resulted in the developers of PSA and S&D creating their own Watchdogs which accommodate their architecture. This could be viewed as duplicated effort.
Optimisation solvers	
• S&D was successfully implemented using widely available open-source optimisation solvers, meaning there was no cost associated with commercial solvers. These solvers were generally able to solve the underlying Selection and Dispatch problems very quickly.	• During the trials, some edges case where identified where specific solvers struggled to find solutions – for example, due to rounding tolerances on inputs which made solutions infeasible. If more time had been available, it would have been beneficial to comprehensively test and evaluate different solver options to identify a best option (in terms of accuracy, reliability, and computation time).
Section 4: User Experience This summarises learning about users experience with the Select & Dispatch tool.	
GUI	
• The GUI was developed in parallel with the backend and enabled users to view and interact with the S&D processes. The GUI exists primarily as a user-friendly view of the system's internal database, filtering and showing only essential information relevant to each page.	• Due to the compressed time scales, there was insufficient time for users to fully engage with the GUI prior to deployment, and early user engagement with the GUI had to be at a very high level. Early engagement with and use of any GUI is vital in order to align the design with both the tool use cases and the user

Learnings and impacts	
Experience	**Opportunity**
• Displaying some of the tool's complexities graphically, i.e., market gate timings, presented a challenge in visually communicating information in a way that transforms the developers low-level understanding of the tool to a user's high-level expectation of the process, although this was eventually achieved.	requirements, and make sure that there is a common understanding between developers and users. • This could have been mitigated by providing working versions earlier in the development process, and prior to the dedicated UAT phase. • A suggestion was made to convert these U.I. elements to a Gantt chart style. Through a more rigorous adoption of agile methodology with earlier chances for feedback, this request may have had the opportunity to be incorporated. • Issues and bugs in the GUI proved to be difficult to identify, and again due to the compressed timescales the issues were generally uncovered during live use of the tool during the trails, which could also have been mitigated through earlier engagement with tool users. A further opportunity to consider would be developing either a thorough manual testing framework, as well as closer inspection of code, or an automated testing framework but this would require significant additional time to develop.
Complex mathematical optimisation	
• The core of the tool utilises a complex mathematical optimisation process. This solver is fast, and the formulation is expandable and therefore can incorporate new constraints, objectives, and features readily. • SSEN and the TRANSITION team have endeavoured to communicate transparently with a wide range of stakeholders about the processes and tools used in the trials, and the complexity of the mathematical optimisation problem proved challenging to communicate to a non-specialist audience.	• The nature of optimisation models makes it extremely difficult to report the reasons for decisions in a non-mathematical form. This issue raised concerns during UAT and trials regarding the ability to explain every decision. In future projects like this one, developers should work towards better reporting of the tools decisions and reporting of an optimisers reasoning should be considered.

Learnings and impacts	
Experience	**Opportunity**
Section 5: Ways of working This section summarises learnings about the ways of working employed within the project.	
Interdisciplinary communication and terminology	
• The scope of the project required input from a range of experts, including software developers, power systems engineers, flexibility service managers, mathematics/optimisation experts, etc. Each discipline has their own preferred terminologies, interpretations, and ways of communication. This introduced a challenge in ensuring all parties sufficiently understood each component of the problem. • To navigate this issue, the team employed a mix of communication approaches, for example flow charts, examples spreadsheets, or mathematical notation. These techniques helped the team to articulate and then effectively debate the complex concepts and ideas involved in the project work. • This was supported by forming and adhering to a glossary of terms early in the project, which helped create a language which was mutually understandable. • It was also very useful to have team overlapping skills and disciplines between members of both the TNEI and SSEN teams. This aided in performing both external internal communication	• It was anticipated that the responsibilities of the PSA and S&D tools would be clearly separable. However, we found this separation was harder to achieve in practice and the understanding of how each tool operates was missing key components e.g., aggregating of sensitivity factors, the necessary uniqueness of the run IDs etc. This could have been navigated by communicating a more detailed understanding of how both tools operate, including what they expect as inputs and deliver as outputs. • A dedicated Solution Architect role would have helped to co-ordinate the development of all of the required tools, and how they operate as a whole, enabling developers to remain focussed on their own tool with co-ordination from the solution architect. This is discussed in more detail below.
Roles and responsibilities	
• Both TNEI and SSEN defined their team structure, including roles and responsibilities, and agreed ways of working for both technical topics and project management and reporting. This allowed technical discussions to	• This way of working was usually very effective; however, it was sometimes difficult to reconcile differences in understanding or expectations between the roles and teams due to the different areas of expertise. For example, if the TNEI

Learnings and impacts	
Experience	**Opportunity**
remained focused, with appropriate project management oversight and approval as required. • Within this structure, the relevant technical experts for different subject matter areas within both the TNEI and SSEN teams became clear. This developed somewhat organically and was especially useful when specific questions or issues arose, and those subject matter experts could discuss a point between the teams and report an outcome or decision back to the wider team.	technical team had different understanding to the project management team it took some time for us to effectively communicate the issue. Our usual approach was to consider how best to present the issue, communicate it in writing or via a short presentation, along with the options for resolving it. This was an effective approach, but sometimes meant it took more time to resolve than might have been anticipated, which was a challenge for the project overall due to the tight timescales. • The technical trials needed a combination of PSA, S&D, and the DSO swivel chair roles to cover all of the required capabilities. For future projects of similar complexity, it could be useful to include an additional role(s) working across all teams that would be responsible for implementing the entire suite of these tools/ roles. As discussed above, this role is often referred to as a System or Solution Architect and holds the responsibility for delivering the overall project requirements, whereas our adopted structure meant that most members of the team were more closely aligned with just one element of this suite.
Meetings and workshops	
• Regular technical and project management meetings were very useful in keeping discussion focussed, as was the separation between these. • Daily calls during UAT and live trials phases proved useful in keeping all participants engaged in the ongoings of the trials and PSA/SND status	• One or two full-day in-person sessions early during design could have been very useful in improving engagement and focus on the development of the tool.

Learnings and impacts	
Experience	**Opportunity**
Section 6: Select and dispatch for decision support This section summarises learnings about how the Select and Dispatch tool could function as a real decision-support tool.	
Formalising the select and dispatch processes	
• During the design phase, the team were able to formalise the commercial processes which underpinned both the selection of responses and dispatch of contracts, through mathematical formulations and flow charts. These provided a very useful reference for discussions and allowed us to make a link with existing approaches (e.g., OPF), where more reference literature is available. This resulted in a full mathematical specification of the problem, which was then easy to code and implement in the tool.	• The mathematical formulation was more complex than initially expected. The initial expectation was that a simple ranking process, using set criteria, would be sufficient. However, it was soon realised that it would only work for the simplest cases and was not a generalisable approach. This meant that, once tool was being used, not all colleagues / stakeholders could engage with it easily.
Alignment with procurement strategy and commercial terms	
• Both the TNEI and SSEN teams endeavoured to comprehensively define the procurement strategy, create quantitative examples, formalised in process diagrams or in a mathematical formulation. • Both teams were pragmatic about what was needed in the S&D tool and what could be managed by the DSO users (e.g., maximum dispatch durations, and partial acceptance of flexibility).	• Future iterations of the S&D tool would need to reflect all the commercial terms (such as customisable maximum dispatch times, no partial acceptance), rather than these being managed by members of the DSO team. • However, it is likely this would mean the optimisation engine would have to adopt mixed integer linear programming (MILP). MILPs are more complex and computationally expensive but would be required to solve for these additional constraints.
Insufficient flexibility available from the market	
• TNEI identified early that situations might arise where there is insufficient flexibility available from market participants to resolve a network issue.	• While the backend solution kept the tool running, it did not provide insight to the tool's users as dummy flexibility results were not recorded, reported in the

Learnings and impacts	
Experience	**Opportunity**
• The approach implemented for handling this was to use “dummy flexibility” within the underlying optimisation in the backend. If there was insufficient flexibility from the market, the solver was then able to fulfil the remaining need using these dummy assets. • This managed the issue for the S&D tool, meaning the network constraint could be mitigated and the tool did not crash, and for the technical trials it meant that flexibility service providers could continue to participate in the events.	UI, or accounted for in the PSA iteration loop exchange. To navigate this issue during the trials there could have been some system in place which keeps tracks of decisions like this that can be displayed back to the user. • Moreover, using dummy flexibility is not a credible BAU option. DSOs should consider what they will do if there is insufficient flexibility (e.g., load shedding? short-term overloading?) and reflect this in their decision support tools and constraint management processes.

5.1 Recommendations

Reflecting on these lessons learned, our key recommendations are:

- **Adopt a blended agile methodology across design and development.** Adopting agile methodologies across both design and development, like Scrum or Kanban, promotes collaboration and frequent feedback loops between users/solution architects and developers. An agile approach to development alone is entirely contingent on a completely well-defined set of requirements and scope of the problem and solution. Adopting iterative and incremental development enables continuous improvements and adaptability to changes. If this approach had been employed within the development of S&D, the HLD and LLD phases of the project would have been significantly compressed, if not omitted entirely. Instead, development of the tool would have commenced much sooner in a modular fashion, based on defined user stories. This might have involved developing the underlying optimisation solver first, then developing the backend around it and then the frontend.
  These early developments could then be used to gather feedback and iterate on the implementation. In practice, this iterative approach ended up happening anyway, as issues were identified during development or testing / trials that required design decisions to be revisited. By adopting an agile methodology from the start, this iteration could be managed deliberately and proactively.
- **Define a Minimum Viable Product (MVP) early on.** Adopting an MVP approach allows developers to focus on implementing the core functionality of the tool that addresses the most critical user needs. By doing so, agile development can quickly progress onto further feature development while gathering valuable feedback from users on the initial MVP, allowing developers to iterate and refine the tool based on wide and user input. Defining an MVP helps in several ways:
  - Faster time-to-production: Prioritising essential features and functionality can expedite the development process and deliver the tool more quickly.
  - User-centric approach: Obtaining user feedback on the MVP provides early validation that the tool aligns with their needs and expectations, as well as prioritising any changes.
  - Risk mitigation: Focusing on the core features in the initial development phase addresses technical risks and challenges early on, reducing the chances of encountering significant issues later in the process.
  - Adaptability and flexibility: Adopting an MVP approach creates room for incorporating new ideas, responding to user feedback, and adapting the product to changing market conditions. This flexibility means the product can evolve based on real-world insights. However, one likely challenge is that users may be reluctant to accept that any features of a tool are only “should” or “could” have features, rather than “must” haves.
- **Incorporate dedicated time for modular and systematic testing.** Modular testing should “bookend” agile sprints, whereby the module is first tested in isolation and ultimately as part of wider integration testing. Similarly, prior to deployment of the production tool, dedicated time must be apportioned to testing of the system, considering boundary testing, and taking a destructive approach. This will help to mitigate the challenges identified and ensure a more effective and efficient development process.

- **Express functional and non-functional requirements in terms of clear definitions and direct, comparable outputs.** These software-oriented requirements would be based on higher level requirements (like those in the RTC produced by SSEN for S&D) and developed in collaboration with all parties. These requirement definitions would include explicitly defined interaction points, as well as inputs and outputs. They could also incorporate, user stories to open discussion and encourage producing a user orientated tool.
- **Implement interfacing systems using programmatic methodologies, such as REST APIs, or provide the interfacing functionality as deployable, modular code.** Taking this approach will significantly improve both the efficiency and consistency of interfacing. File-based interfaces do provide transparency, and so these should be designed and developed as independent modules for tracing decisions and communications between systems, but they should become a child process of programmatic interfaces. Programmatic interfaces also allow for greater flexibility and customisation, avoiding miscommunication on sending/receiving ends which can take much more time to resolve. This might mean, for example, turning the underlying calculations for both PSA and S&D into separate Python packages, that a live tool(s) could then incorporate internally. This is similar to the approach taken for the TRANSITION baseline tool.
- **Complex optimisation processing requires automated solution interpretation.** Understanding the decision-making process of complex mathematical optimisation mechanisms in a BAU solution is crucial for a variety of reasons. Firstly, it promotes transparency, enabling stakeholders to comprehend how and why specific decisions were made, thereby fostering trust in the system. This aspect is particularly vital when the optimisation algorithm influences high-stakes sectors such as the energy industry. Secondly, it facilitates troubleshooting and improvement. By explaining each step in the decision process, potential weaknesses or errors within the model can be detected, allowing for necessary modifications and enhancements. Therefore, the implementation of a dedicated module for decoding the decision process of these mechanisms is highly recommended in future developments.

Appendix A – S&D Change Logs

The following is a record of the change logs issued alongside updates to the tool. These were distributed via email to the SSEN Technical Team.

Each change log is a compilation of the changes made to the tool, combining changes made in each previously non-reported revision which was not deployed. As the outcome of the trials is only relevant to the versions of the tool which are live, specific information about these intermediate non-deployed versions is not included in this report. That information may be obtained from the Azure DevOps platform upon which development was conducted.

It is important to note that each of these changes does not represent a 'release' of the tool as compressed timescales resulted in the need to deploy these intermediate versions.

Further information has been added to these change logs for the purpose of context and reasoning.

0.3.7b – 08/03/2023

- Backend API
  - Ensure asset level reliabilities can be changed and updates the request creation process.
  - Ensure market membership is considered whilst response filtering.
  - Increase timeout time for table GET requests to 4s.
    - Extra processes resulting in performance bottleneck require more time to process before the front end assumes an error has occurred.
- UI/Example data
  - display dynamic requests to UI.
  - Example data no longer assumes response ID
  - display BSP and Primary columns on requests (for commercial grouping).

0.3.9b – 13/03/2023

- Enhanced data handling on response upload
  - If all responses are invalid (i.e., by window/price ceiling/market membership, an error will be thrown to the user)
- Commit all irrelevant requirements.
  - Requirements that are ignored (for example a negative requirement outside of the 10-2 window) are committed and displayed in the tool.
- Display/export dynamic requests
  - Users can now export and respond to dynamic requests.
- Filter market gate relevant records (from now, until end of market gate)
  - The "upcoming" tabular data shows requirements/requests/responses/contracts/dispatches that are relevant to the current market gate.
  - Requests are now displayed until the cut off time for providing responses.
- Assume all DCM request branches to be Cowley Local primary.
  - Automatically generated DCM requests require some branch to be generated for as S&D has no knowledge of network topology this was hard-coded to the Cowley local primary
- Display AUT/MAN as a column across tables
  - The tool now reports any data as originating from a manually or automatically produced file.
- Content of PSA-NO-FLEX file is now irrelevant.
  - SND no longer reads of interprets this file and does not expect any kind of content.

- Generate example responses, created whenever the tool generates a request.
- Rename upcoming requirements page as upcoming constraints.
- Default DCM request of 20MW.
- Improve formatting of auction times in tables
- Update request timer
    - This timer now counts down to the time at which the next auction begins i.e., time left to publish request to NMF
- Update dispatch timer (countdown to end of notification for the next dispatch, action before timer ends)
    - This timer now counts down to the next dispatch notification deadline if one exists. For example, if a DA SCM dispatch is expected for tomorrow (Tue 14th March at 5pm), the timer will count down to 1pm (4 hours prior to dispatch).
- Manage PSA requirement ID as "run ID_req ID" (within S&D)
    - PSA requirement ID will be parsed and provided to PSA in selection/dispatch response exchange process.
- Updated market schedule
    - Week ahead requests made available by S&D from 6AM (instead of 8:30AM).
    - Week ahead responses must be provided by 9AM, contracts made available by 11AM.

1.0.0 – 21/03/2023

- PSA Interface
    - Various alignments with PSA file interface where differences where identified.
- Bug fixes
    - Added AUT/MAN filter on tables.
        - Allows user to filter the table for data related to either manual or automated flexibility requirements.
    - Added Sensitivity factor columns to tool for auditing
    - Acceptance of dynamic responses
        - All dynamic responses are now automatically accepted, a previously identified bug prevented this behaviour.
    - DA Auction start/end times
        - Tool now displays auction start/end times relevant to both S&D and NMF time scales to be given to swivel chair users and participants respectively.
    - Timers on home page show correct durations after feedback.
        - Request – time left to publish to NMF.
        - Select – time left to upload responses.
        - Dispatch – time left to notify IA of upcoming dispatch.
    - Tool now includes all upcoming dispatches in table.
        - Previously showed upcoming dispatches for current window only
- Outstanding issues to be resolved
    - Handling SEPM services
        - Considering the change in implementation based on last week's discussions, S&D's candidate check process is being refined to account for negative residual flex.
    - Implement ordering of table data columns
        - The current order is arbitrary and inconsistent.
    - Asset tab on explore data page incorrectly highlighted on explorer tab load.
        - There is a mismatch between the GUIs internal state and what is displayed, causing the tool to display the responses table, whilst claiming it's the asset table.

1.0.4a – 04/04/2023

- New Features:
  - Automated SQL database backups at each decision point.
    - Enable viewing and navigating between states of the tool before each meaningful interaction – uploading/modifying responses and receiving requirements.
  - Automated backups of any response uploads.
    - All uploaded responses are saved to the PSA2SND folder for backing up and providing context.
  - An improved log window which enables the user to view the processes being carried out by the tool.
  - Additions to the context menu view for easier navigation of results, right click a response, request, or dispatch to view all data directly linked to its request it satisfies.
    - Enables a user to view the entire process for one request in a single window.
  - Update handling of SEPM services as previously discussed.
  - Implemented algorithm which sorts all data into a consistent form – IDs, data, then time.
  - Fixed asset tab wrongfully highlighted in explore data window.
- Bug Fixes:
  - Fix to S&D's internal process which caused a crash when handling dispatchable requirements.

1.0.6 – 20/04/2023

- Process changes
  - If a constraint is in the past, S&D uses the original forecast (highlighted when reprocessing S&D analysis).
    - This was added to enable reprocessing of the trials using historical data.
  - Drop max residual flex exit condition.
    - This arbitrary exit condition was forcing early exit of the candidate check process, only in cases where the candidate check process was resolving the linearization of sensitivity factors for a set of responses while a large constraint resulted in maximum dispatch of relevant assets.
  - Enforce max iteration = 3 while PSA is not aggregating responses for individual assets.
    - This will be increased when PSA is deployed with the aggregation of responses/contracts at the SF calculation and candidate check phases.
- Bug fixes
  - Manage rounding/tolerance of the solver.
    - The outputs of the optimization solver included some floating-point rounding errors which resulted in the tool exhibiting unexplainable behavior in some cases. This issue was exacerbated by excessive volumes of dummy flexibility to resolve a constraint. Due to the large volume of expensive dummy flex, we observed a bias towards this. We have since adjusted the tolerance and method of the solver to prevent this.
  - Prior dispatches of SEPM services are considered in the correct direction (export).
  - Improve filtering of responses/contracts.
    - To avoid calculation of responses outside the constraint window for a given service.

- Dispatch table/export outputs have been corrected and distilled down to support the swivel chair user in the notification process.

1.0.7 – 02/05/2023

- Bug Fixes:
  - Prevent responses from providing a response window outside the request's date.
  - Fix duplicate dispatches in NMF export file caused by asset reliability data.
  - Remove expected columns when processing sensitivity factors which are no longer sent by PSA. (“offered_power_kw”, “secondary”)
    - The SND file system enforces a strict policy on received files to prevent erroneous data, this makes it inflexible against changing file content.
- Technical Changes:
  - Added functionality to update asset table by providing new data as a static csv
  - Updated asset data to v6.
  - Reject responses which extends beyond the date of the request window.
    - The tool would previously accept responses which offered flexibility over several calendar days, the processes of the tool do not expect this and results in unexpected behaviours.
- Performance enhancements:
  - Filter out sensitivity factors where no overlap exists between requirements and contracts during request creation.

Appendix B – Requirement testing results

The figures below show summary reports of the results of automated testing of the S&D Tool code.

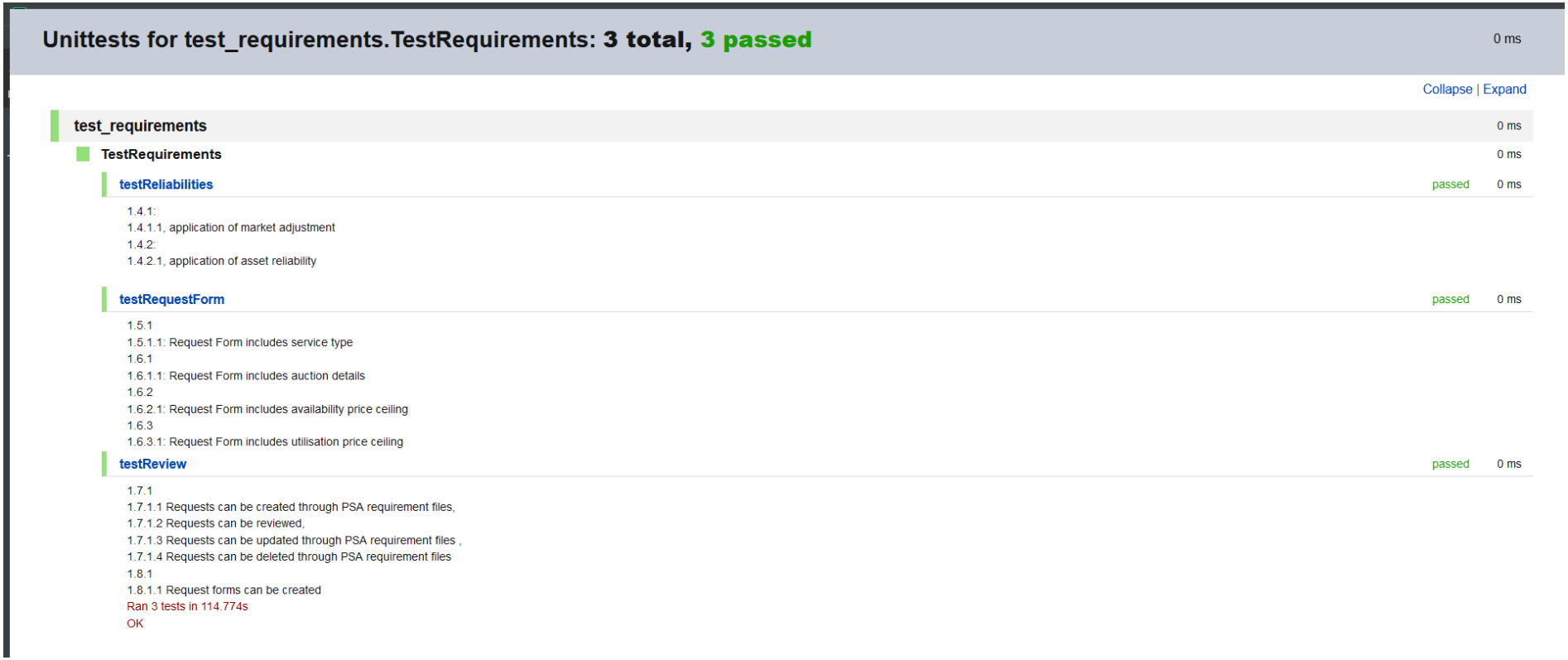

Figure 15: Requirement test results

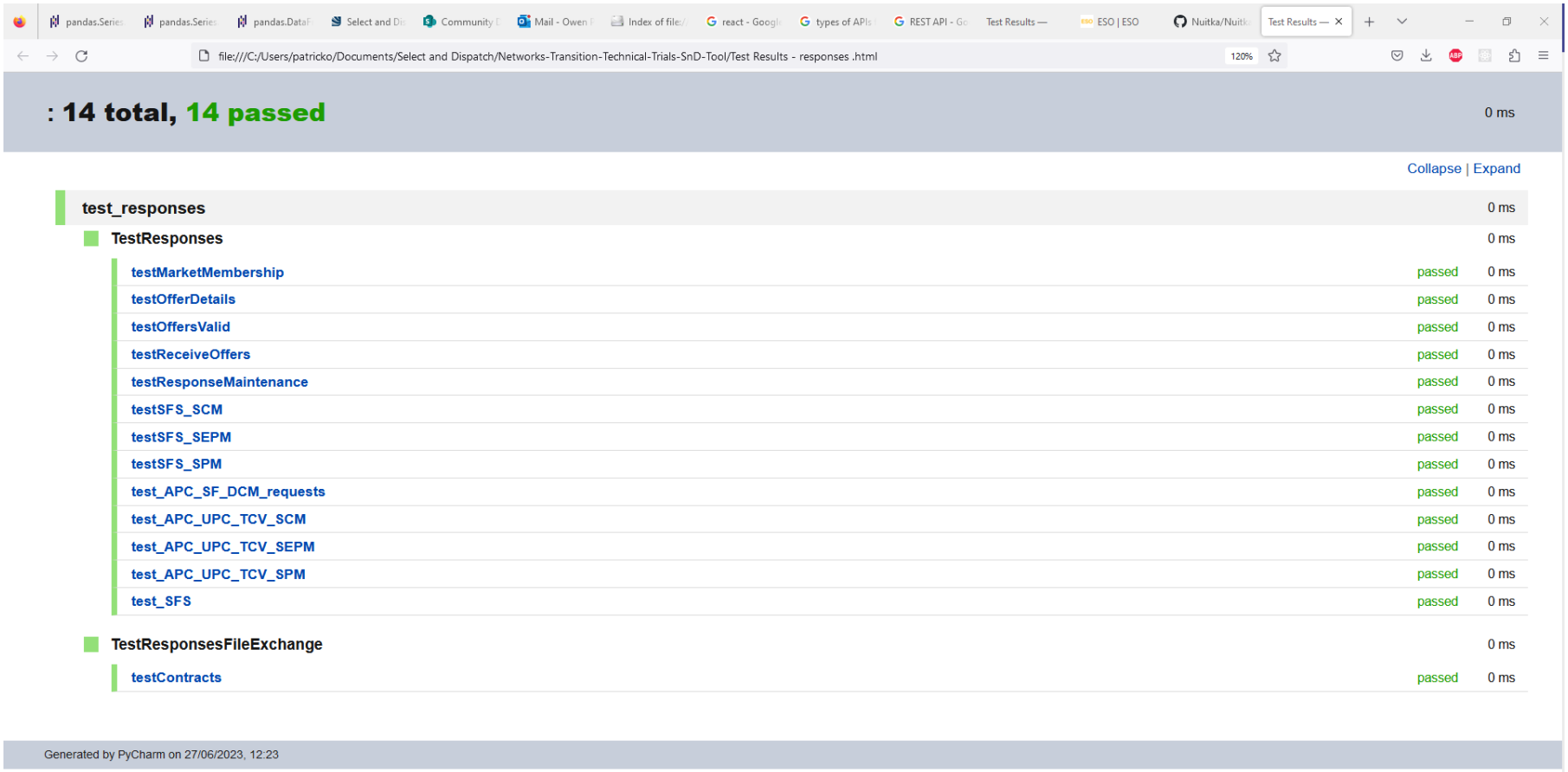

Figure 16: Response selection test results

Figure 17: Dispatch selection test results