Opturion – Mobile Workforce Optimiser
Data Formats and Web Service – Version 1

Opturion

November 26, 2026

Contents

0 Version History **1**

1 Overview **2**

2 Optimiser Data Formats **3**
2.1 Common Data Items . . . 3
2.1.1 Identifiers . . . 3
2.1.2 Times . . . 3
2.1.3 Durations . . . 3
2.1.4 Dates . . . 3
2.1.5 Date-Times . . . 3
2.1.6 Time Zones . . . 4
2.1.7 Date Window . . . 4
2.1.8 Date-Time Window . . . 4
2.1.9 Time Window . . . 4
2.1.10 Location Object . . . 4
2.1.11 Allowed Crew Object . . . 5
2.1.12 Response Status Codes . . . 5
2.1.13 Dispatch Response Status Codes . . . 5
2.2 Planning Optimiser Input . . . 6
2.2.1 Plan Request Object . . . 6
2.2.2 Plan Crew Object . . . 6
2.2.3 Plan Shift Object . . . 8
2.2.4 Visit Segment Penalties . . . 8
2.2.5 Plan Job Object . . . 8
2.2.6 Plan Request Assignment Object . . . 8
2.2.7 Plan Request Precedences . . . 9
2.3 Planning Optimiser Output . . . 10
2.3.1 Plan Submit Response Object . . . 10
2.3.2 Plan Query Response Object . . . 10
2.3.3 Plan Fetch Response Object . . . 10
2.3.4 Phase Codes . . . 11
2.3.5 Check Response Object . . . 11
2.3.6 Plan Response Assignment Object . . . 11
2.4 Dispatch Optimiser Input . . . 12
2.4.1 Dispatch Request Object . . . 12
2.4.2 Dispatch Crew Object . . . 12
2.4.3 Dispatch Job Object . . . 13
2.4.4 Job Type Code . . . 13
2.4.5 Job Status Code . . . 13
2.4.6 Assigned Job Object . . . 14
2.5 Dispatch Optimiser Output . . . 14
2.5.1 Dispatch Response Object . . . 14
2.5.2 Dispatch Recommendation Object . . . 14
2.5.3 Dispatch Stop Object . . . 15

3 Microsoft Excel Input **16**
3.1 Identifiers in Excel . . . 16

3.2 Times in Excel . . . 16
3.3 Dates . . . 17
3.4 Date-Times . . . 17
3.5 Sheet: General . . . 18
3.5.1 Column: Parameter . . . 18
3.5.2 Column: Value . . . 18
3.5.3 Parameter: Id . . . 18
3.5.4 Parameter: Name . . . 18
3.5.5 Parameter: Owner . . . 18
3.5.6 Parameter: Time Zone . . . 18
3.5.7 Parameter: Start Date . . . 18
3.5.8 Parameter: Finish Date . . . 19
3.5.9 Parameter: Next Day . . . 19
3.5.10 Parameter: Iterations . . . 19
3.5.11 Parameter: Segment Fixed Penalty . . . 19
3.5.12 Parameter: Segment Distance Penalty . . . 19
3.6 Sheet: Time Windows . . . 20
3.6.1 Column: Id . . . 20
3.6.2 Column: Start . . . 20
3.6.3 Column: End . . . 20
3.7 Sheet: Locations . . . 21
3.7.1 Column: Id . . . 21
3.7.2 Column: Latitude . . . 21
3.7.3 Column: Longitude . . . 21
3.8 Sheet: Crews . . . 22
3.8.1 Column: Id . . . 22
3.8.2 Column: Name . . . 22
3.8.3 Column: Start Time . . . 22
3.8.4 Column: Finish Time . . . 22
3.8.5 Column: Start Location . . . 22
3.8.6 Column: Finish Location . . . 22
3.8.7 Column: Cost per Use . . . 22
3.8.8 Column: Cost per Hour . . . 23
3.8.9 Column: Cost per Kilometre . . . 23
3.8.10 Column: Hidden Cost per Use . . . 23
3.8.11 Column: Hidden Cost per Hour . . . 23
3.8.12 Column: Hidden Cost per Kilometre . . . 23
3.8.13 Column: Visit Segment Distance Penalty . . . 23
3.8.14 Column: Visit Segment Fixed Penalty . . . 23
3.8.15 Column: Subcontractor . . . 24
3.8.16 Column: Overtime . . . 24
3.9 Sheet: Crew Locations . . . 25
3.9.1 Column: Crew . . . 25
3.9.2 Column: Date . . . 25
3.9.3 Column: Start Location . . . 25
3.9.4 Column: Finish Location . . . 25
3.10 Sheet: Roster . . . 26
3.10.1 Column: Id . . . 26
3.10.2 Column: All Other Columns . . . 26

3.11 Sheet: Jobs . . . 27
3.11.1 Column: Id . . . 27
3.11.2 Column: Name . . . 27
3.11.3 Column: Latitude . . . 27
3.11.4 Column: Longitude . . . 27
3.11.5 Column: Default Time Window . . . 27
3.11.6 Column: Default Duration . . . 27
3.11.7 Column: Default Cost . . . 27
3.11.8 Column: Required Equipment . . . 28
3.12 Sheet: Job Times . . . 29
3.12.1 Column: Id . . . 29
3.12.2 Column: All Other Columns . . . 29
3.13 Sheet: Job Durations . . . 30
3.13.1 Column: Id . . . 30
3.13.2 Column: All Other Columns . . . 30
3.14 Sheet: Job Costs . . . 31
3.14.1 Column: Id . . . 31
3.14.2 Column: All Other Columns . . . 31
3.15 Sheet: Job Crew Compatibility . . . 32
3.15.1 Column: Job . . . 32
3.15.2 Column: All Other Columns . . . 32
3.16 Sheet: Job Crew List . . . 33
3.16.1 Column: Job . . . 33
3.16.2 Column: Crew . . . 33
3.17 Sheet: Precedences . . . 34
3.17.1 Column: before . . . 34
3.17.2 Column: after . . . 34
3.17.3 Column: Minimum Gap . . . 34
3.17.4 Column: Maximum Gap . . . 34
3.17.5 Column: Same Crew . . . 34
3.18 Sheet: Assignments . . . 35
3.18.1 Column: Job . . . 35
3.18.2 Column: Crew . . . 35
3.18.3 Column: Start Time . . . 35
3.18.4 Column: Finish Time . . . 35
3.18.5 Column: Flex . . . 35

4 Microsoft Excel Output **36**
4.1 Sheet: Solution Summary . . . 36
4.1.1 Scenario Id . . . 36
4.1.2 Scenario Name . . . 36
4.1.3 Scenario Owner . . . 36
4.1.4 Total Cost . . . 36
4.1.5 Total Distance . . . 36
4.1.6 Assigned Crew Shifts . . . 36
4.1.7 Total Crew Shifts . . . 37
4.1.8 Assigned Jobs . . . 37
4.1.9 Total Jobs . . . 37
4.2 Sheet: Assignments . . . 37

4.2.1 Crew . . . 37
4.2.2 Job . . . 37
4.2.3 Column: Start Time . . . 37
4.2.4 Column: Finish Time . . . 37
4.3 Sheet: Unassigned . . . 37
4.3.1 Job . . . 37
4.3.2 Column: Latitude . . . 38
4.3.3 Column: Longitude . . . 38
4.3.4 Column: Required Equipment . . . 38

5 Planning Web Service **39**
5.1 Planning Task Identifiers . . . 39
5.2 Submit-Plan Endpoint . . . 39
5.2.1 Submit-Plan Requests . . . 39
5.2.2 Submit-Plan Request Example . . . 39
5.2.3 Submit-Plan Responses . . . 40
5.3 Query-Plan Endpoint . . . 41
5.3.1 Query-Plan Requests . . . 41
5.3.2 Query-Plan Request Parameters . . . 41
5.3.3 Query-Plan Request Example . . . 41
5.3.4 Query-Plan Responses . . . 41
5.4 Cancel-Plan Endpoint . . . 43
5.4.1 Cancel-Plan Requests . . . 43
5.4.2 Cancel-Plan Request Example . . . 43
5.4.3 Cancel-Plan Responses . . . 43
5.5 Fetch-Plan Endpoint . . . 44
5.5.1 Fetch-Plan Requests . . . 44
5.5.2 Fetch-Plan Request Example . . . 44
5.5.3 Fetch-Plan Responses . . . 44
5.6 Check-Plan Endpoint . . . 45
5.6.1 Check-Plan Requests . . . 45
5.6.2 Check-Plan Request Example . . . 45
5.6.3 Check-Plan Responses . . . 45

6 Dispatch Web Service **46**
6.1 Dispatch Endpoint . . . 46
6.1.1 dispatch Requests . . . 46
6.1.2 dispatch Request Parameters . . . 46
6.1.3 dispatch Request Example . . . 46
6.1.4 dispatch Responses . . . 46
6.2 Check-Dispatch Endpoint . . . 48
6.2.1 Check-Dispatch Requests . . . 48
6.2.2 Check-Dispatch Request Example . . . 48
6.2.3 Check-Dispatch Responses . . . 48

Appendices **49**

A Planning Start-Of-Month Input Example **49**

B Planning Nightly Reschedule Input Example **54**

C Dispatch Input Example **59**

D Unsupported Excel Functions **62**

0 Version History

Version	Date	Author(s)	Description
1	2021-03-24	jfischer	Initial version.

November 26, 2026

1 Overview

This document describes the web services of the Opturion mobile workforce optimiser (MWO) and the input and output data formats that it uses.

The MWO consists of a *planning* optimiser and a *dispatch* optimiser. The planning optimser is used to construct and maintain long term schedules that assign jobs to crews; the dispatch optimiser generates recommendations for the assignment of responsive (i.e. "on-the-day") jobs to crews.

2 Optimiser Data Formats

The input and output to both the planning and dispatch optimisers takes the form of UTF-8 encoded JSON documents.

Any fields are omitted in the input take default values. Unless noted otherwise, these default values are `0` for numeric fields, `false` for Boolean fields and `null` for other types of field (including strings).

2.1 Common Data Items

This section describes the representation of data that is common across the MWO input and output formats.

2.1.1 Identifiers

Entities, such as crews and jobs, must have a unique *identifier* (Id) that is used to refer to them.

Identifiers must contain at least one character and must not contain the semicolon (`;`) character. They are not case or "white space" sensitive.

For example, "`JOB2`", "`job2`" and "`Job 2 `" will all be treated as identifying the same entity by the optimiser because they only differ in case or the amount of white space they contain.

Even though they may be composed of digits, identifiers are not numeric. For example, the optimiser will treat "`1`" and "`001`" as distinct identifiers.

All references to an entity are made via its identifier.

In JSON, identifiers are representing using strings.

2.1.2 Times

Points in time during a day are given in *HH:mm* format using the 24 hour clock, for example `12:30` or `22:05`.

Times are represented using strings in JSON.

2.1.3 Durations

A duration of time is given in *HH:mm* format, for example `12:30`. Unlike times, these can extend over 24 hours such as `37:00`.

Durations are represented using strings in JSON.

2.1.4 Dates

Calendar dates are given in *YYYY-MM-DD* format, for example `2019-07-16` or `2020-01-02`.

Dates are represented using strings in JSON.

2.1.5 Date-Times

Combined dates and times are given in *YYYY-MM-DD***T***HH:mm* format, for example `2019-07-16T10:59` or `2019-12-24T13:01`.

Date-times are represented using strings in JSON.

2.1.6 Time Zones

Time zones have the form *Area/Location* and must match an entry in the IANA time zone database [1], for example "Australia/Melbourne" or "Asia/Kolkata".

Time zones are represented using strings in JSON.

2.1.7 Date Window

A *date window* is a contiguous span of dates. It is identified by, and inclusive of, the endpoints of the window. It is represented in JSON using a **date window** object, which has the following fields:

1. `start`: the date on which this date window begins.
2. `finish`: the date on which this date window ends.

2.1.8 Date-Time Window

A *date-time window* is a contiguous span of time where the endpoints include both a date and a time component. It is identified by, and inclusive of, the endpoints of the window. It is represented in JSON using a **date-time window** object, which has the following fields:

1. `start`: the date-time on which this date-time window begins.
2. `finish`: the date-time on which this date-time window ends.

2.1.9 Time Window

A *time window* is a contiguous span of time. It is identified by, and inclusive of, the end points of the window. It is represented in JSON using a **time window** object, which has the following fields:

1. `start`: the time on which this time window begins.
2. `finish`: the time on which this time window ends.

If the **finish** field occurs before the **start** field, then that means that the finish of the time window occurs on the following day.

2.1.10 Location Object

A location is given using its geographic coordinates. It is represented in JSON using a **location** object, which has the following fields:

1. `longitude`: a number giving the longitude in decimal degrees.
2. `latitude`: a number giving the latitude in decimal degrees.

[1] https://www.iana.org/time-zones

2.1.11 Allowed Crew Object

An **allowed crew** object describes *if* a crew can undertake a job; the amount of time they require complete the job; a measure of the desirability of assigning the job to the crew and the monetary cost of assigning the crew to the job.

1. `crew_id`: the identifier of the crew.
2. `duration`: the number (integer) of minutes the crew require to complete the job (once they are on-site).
3. `preference`: a number indicating the desirability (or not) of assigning the job to this crew. A value of `0` means that there is no preference.
4. `cost`: a number, giving the monetary cost in dollars for the crew to undertake the job.

2.1.12 Response Status Codes

A *response status code* is used by the optimiser to indicate if an operation succeeded or not. There are two possible values:

1. `OK`: indicates that the operation was successful.
2. `ERROR`: indicates that the operation failed and an error occurred.

2.1.13 Dispatch Response Status Codes

A *dispatch response status code* is used by the dispatch optimiser to indicate if an operation succeeded or not. There are four possible values:

1. `OK`: indicates that the operation was successful.
2. `NO_COMPATIBLE_CREWS`: indicates that the operation was successful, but no crews were compatible with the new job.
3. `MISSING_REQUIRED_EQUIPMENT`: indicates that the operation was successful, but no crews had the equipment required to undertake the new job.
4. `ERROR`: indicates that operation failed and an error occurred.

2.2 Planning Optimiser Input

This section describes the input to the planning optimiser. The planning optimiser has two modes of operation:

1. `Plan`: construct a new schedule of job-crew assignments of a given planning period (e.g. one week, three months etc.)
2. `Replan`: reschedule planned jobs within an existing schedule.

There is some overlap between these two modes of operation. Both modes of operation use the same input format, which takes the form of a plan request object.

2.2.1 Plan Request Object

The input to the planning optimiser is a **plan request** object. It has the following fields:

1. `id`: an **identifier** assigned by the client that identifies this invocation of the optimiser.
2. `name`: a string naming this instance; may be empty or `null`.
3. `time_zone`: a string giving the time zone.
4. `start_date`: the date of the first day of the planning period.
5. `finish_date`: the date of the last day he planning period.
6. `next_day`: a date identifying the next day from which rescheduling is to occur. This date must be inside the start and finish dates defined in the `start_date` field and `finish_date` field respectively. For the "plan" mode of operation this field must be `null`.
7. `crews`: a non-empty array of plan crew objects that describes crews and their availability.
8. `jobs`: a non-empty array of plan job objects that describes planned jobs.
9. `assignments`: an array of plan request assignment objects that describes any existing assignments of planned jobs to crews; may be `null` or empty if there are no existing assignments.
10. `precedences`: an array of plan request precedence objects that describes rules about the minimum and / or maximum time between jobs; may be `null` or empty if there are no existing precedences.

2.2.2 Plan Crew Object

A **plan crew** object describes a single crew. It has the following fields:

1. `id`: the unique identifier for the crew.
2. `name`: a string giving a descriptive label for the crew; may be `null`.
3. `roster`: an array of plan shift objects, giving the rostered shifts for the crew.
4. `start_location`: the default location where the crew begins their shift; must *not* be `null`.
5. `finish_location`: the default location at where the crew should end their shift; may be `null` if the crew is considered to finish their shift at the completion of their last job.
6. `is_subcontractor`: a Boolean, if `true` the crew are subcontractors.

7. `allowed_overtime`: a Boolean, if `true` the crew are allowed overtime.
8. `visit_segment_fixed_penalty`: a number giving the value of the visit segment fixed penalty for this crew.
9. `visit_segment_distance_penalty`: a number giving the value of the visit segment distance penalty for this crew.

2.2.3 Plan Shift Object

A **plan shift** object describes a shift in a crew's roster. It has the following fields:

1. `start_time`: the rostered shift start date-time of the crew; this must *not* be `null`.
2. `finish_time`: the rostered shift finish date-time of the crew; this must *not* be `null`.
3. `start_location`: the location where the crew begins their shift. If present, this overrides the default start location for this shift.
4. `finish_location`: the location at where the crew should end their shift. If present, this overrides the default finish location for this shift.

2.2.4 Visit Segment Penalties

Visit segment penalties are used to affect the "shape" of a crew's route. A *segment* is defined as the transit between two distinct locations at which jobs occur. Transit to and from the crew's start or finish location (if defined) is *not* considered a segment.

There are two types of visit segment penalty:

1. **visit segment fixed penalty**: applied for each segment in a crew's route. A positive value for this penalty will have the effect of minimising the number of locations that a crew visits.
2. **visit segment distance penalty**: a factor that is multiplied by the transit distance in each segment in crew's route. A positive value for this penalty will have the effect of minimising the distance travelled between jobs, resulting in the jobs being clustered within "close" proximity to each other.

2.2.5 Plan Job Object

An **plan job** object describes a single planned job. It has the following fields:

1. `id`: the unique identifier for the job.
2. `name`: a string giving a descriptive label for the job; may be `null`.
3. `location`: the location of the job; must *not* be `null`.
4. `allowed_crews`: an array of allowed crew objects listing the crews capable of undertaking the job. May be `null` or empty if no crews is capable of undertaking the job. Crews present in this array must be present in the plan crew object array.
5. `required_equipment`: an array of non-null and non-empty strings listing additional equipment that is required to complete the job; the array itself may be empty or `null` which indicates that no additional equipment is required.
6. `time_windows`: an array of date-time windows, specifying when the job can be undertaken; may be `null` or empty in which case where there are no restrictions on when the job may be done. Any part of a time window that falls outside the planning period will be discarded.

2.2.6 Plan Request Assignment Object

An **plan request assignment** object describes the allocation of an job to a crew. It has the following fields:

1. `crew_id`: the unique identifier of the crew.
2. `job_id`: the unique identifier of the job.
3. `start_time`: the date-time at which the job is scheduled to commence.
4. `finish_time`: the date-time at which the job is scheduled to complete.
5. `flex`: a number between `0` and `1` (inclusive), which is a measure of how much the optimiser may change the date of an assigned job. A value of `0` means that the job *must* be assigned on the date given by the **start_time** field; a value of `1` means the optimiser is completely free to reschedule the job; values between `0` and `1` control how close to the originally assigned date the job can be rescheduled.

2.2.7 Plan Request Precedences

An **plan request precedence** object describes a time constraint between two jobs. A precedence constrain can specify a *minimum* and *maximum* duration between jobs. These durations occur between the completion of the first job and the start of the second.

It has the following fields:

1. `before`: the unique identifier of the first job.
2. `after`: the unique identifier of the second job.
3. `minimum_gap`: the minimum duration between the `before` and `after` jobs.
4. `maximum_gap`: the maximum duration between the `before` and `after` jobs.
5. `same_crew`: a Boolean, if `true` then both jobs must be assigned to the same crew.

2.3 Planning Optimiser Output

This section describes the output from the planning optimiser.

2.3.1 Plan Submit Response Object

A **plan submit response** object is returned when a plan request is submitted via the SUBMIT-PLAN endpoint. It has the following fields:

1. `client_id`: a string giving the user assigned identifier of the optimisation job. (This is the same as the `id` field of the plan request object with which the optimiser was invoked.) This field may be `null` if the `status` field is `ERROR` (e.g. if the input was misformed and no usable `client_id` could be found).
2. `task_id`: a string giving the task identifier assigned to the submitted instance by the optimiser. This field will be `null` if the `status` field is `ERROR`.
3. `status`: a string giving one of the response status codes.
4. `errors`: if `status` is `ERROR` then this is an array of strings describing the errors encountered; it is `null` otherwise.
5. `warnings`: an array of strings describing potential problems.

2.3.2 Plan Query Response Object

A **plan query response** object is returned when querying the state of a planning optimisation job via the QUERY-PLAN endpoint. It has the following fields:

1. `client_id`: a string giving the user assigned identifier of the optimisation job.
2. `phase`: a string which is one of the phase codes.
3. `progress`: a number between `0` and `1` indicating optimiser progress.
4. `iteration`: an integer, giving the current number of iterations that the optimiser has run for. This will be `0` if the `phase` is one of `PENDING` or `COMPUTE_MATRIX`.
5. `of`: the total number (integer) of iterations that the optimiser will run for.
6. `objective`: the value of the optimiser's objective function for the best solution found so far. This is not meaningful if the `phase` is `PENDING` or `COMPUTE_MATRIX`. "Better" solutions are those whose `objective` value tends towards $-\infty$.

2.3.3 Plan Fetch Response Object

A **plan fetch response** object is returned when fetching the result of a planning optimisation job via the QUERY-PLAN endpoint. It has the following fields:

1. `client_id`: a string giving the user assigned identifier of the optimisation job.
2. `status`: a string giving one of the response status codes.
3. `assignments`: if `status` is `OK` then this is (possibly empty) array of plan response assignment objects giving the assignment of jobs to crews; it is `null` if `status` is `ERROR`.
4. `errors`: if `status` is `ERROR` then this is an array of strings describing the errors encountered; it is `null` otherwise.

5. `total_cost`: if `status` is `OK` then this a number giving the total monetary cost for the solution; it is `0` otherwise.
6. `total_distance`: if `status` is `OK` then this is a number giving the total distance (in km) for the solution; it is `0` otherwise.
7. `total_crew_shifts`: if `status` is `OK` then this is an integer giving the total available crew-shifts for the planning period. (A *crew-shift* denotes a crew for given day / shift.)
8. `assigned_jobs`: if `status` is `OK` then this is an integer giving the number of assigned jobs for the solution; it is `0` otherwise.
9. `total_jobs`: if `status` is `OK` then this is an integer giving the total number of jobs to be assigned for the planning period; it is `0` otherwise.
10. `objective`: if `status` is `OK` then this a number giving the optimiser's objective function value for the solution; it is `0` otherwise;
11. `build`: a string giving the version of the optimiser that created this response.

2.3.4 Phase Codes

A **phase code** describes the state of a planning optimisation job. There are five possible values:

1. `PENDING`: the job has been accepted by the optimiser but it has not yet started running.
2. `COMPUTE_MATRIX`: the optimiser is computing route information required for the job.
3. `OPTIMISE`: the optimiser is running the job.
4. `READY`: the optimiser has finished running the job and a result is available.
5. `FAILED`: an error occurred while the optimiser was running the job.

2.3.5 Check Response Object

A **check response** object is returned when a plan request or dispatch request is submitted via the CHECK-PLAN or CHECK-DISPATCH endpoint. It has the following fields:

1. `status`: a string giving one of the response status codes.
2. `errors`: if `status` is `ERROR` then this is an array of strings describing the errors encountered; it is `null` otherwise.

2.3.6 Plan Response Assignment Object

An **plan response assignment** object describes the allocation of an job to a crew. It has the following fields:

1. `crew_id`: the unique identifier of the crew.
2. `job_id`: the unique identifier of the job.
3. `start_time`: the date-time at which the job is scheduled to commence.
4. `finish_time`: the date-time at which the job is scheduled to be completed.

2.4 Dispatch Optimiser Input

This section describes the input to the dispatch optimiser. The input takes the form of a dispatch request object.

2.4.1 Dispatch Request Object

The input to the dispatch optimiser is a **dispatch request** object. It has the following fields:

1. `id`: a non-empty string that uniquely identifies each invocation of the optimiser.
2. `time_zone`: a string giving the time zone.
3. `current_time`: a date-time giving the current date and time.
4. `new_job_id`: the identifier of the new responsive job to generate dispatch recommendations for.
5. `crews`: an array of dispatch crew objects describing the crews that are, or will become, available for assignment of new responsive jobs. This array must not be `null` or empty.
6. `jobs`: an array of dispatch job objects, describing the responsive *and* planned jobs. The array should include assigned planned and responsive jobs, unassigned planned jobs that could possibly be assigned and the new responsive job identified by the **new_job_id** field. This array must not be `null` or empty.

2.4.2 Dispatch Crew Object

A **dispatch crew** object describes a single crew. It has the following fields:

1. `id`: the unique identifier for the crew.
2. `name`: a string giving a descriptive label for the crew; may be `null`.
3. `start_time`: the rostered shift start date-time of the crew.
4. `finish_time`: the rostered shift finish date-time of the crew.
5. `start_location`: the location at which the crew begins their shift; must *not* be `null`.
6. `finish_location`: the location at where the crew should end their shift; may be `null` if the crew is considered to finish their shift at the completion of their last job.
7. `current_location`: the current location of the crew.
8. `is_subcontractor`: a Boolean, if `true` the crew are subcontractors.
9. `allowed_overtime`: a Boolean, if `true` the crew are allowed overtime.
10. `equipment`: an array of strings listing any extra equipment *currently* in the crew's possession.
11. `current_activity`: an array of assigned job objects listing the currently scheduled activity of the crew. The array *must* be ordered by time. Can be `null` or empty if the crew has no currently scheduled activity.

2.4.3 Dispatch Job Object

A **dispatch job** object describes a responsive job *or* a planned job that is, or could be, undertaken on this day. It has the following fields:

1. `id`: the unique identifier of the job.
2. `name`: a string giving a descriptive label for the job; may be `null`.
3. `type`: a string, must be one of the job type codes.
4. `status`: a string, must be one of the job status codes.
5. `priority`: a number (integer), between `1` and `4`, giving the priority of a responsive job; this field is ignored for planned jobs.
6. `location`: the location of the job. Must *not* be `null`.
7. `allowed_crews`: an array of allowed crew objects listing the crews capable of undertaking the job.
8. `required_equipment`: an array of strings listing additional equipment required to complete the job.
9. `response_time_windows`: an array of date-time window objects giving time spans in which a responsive job must begin; this field must be `null` for planned jobs. Must not be `null` or empty for responsive jobs.
10. `resolution_time_windows`: an array of date-time window objects giving time spans in which a responsive job must be completed; this field must be `null` for planned jobs. Must not be `null` or empty for responsive jobs.
11. `previously_responded`: a Boolean, `true` means that this *case* has already been responded to in a previous job; `false` means that this *case* has *not* been responded to in a previous job.

2.4.4 Job Type Code

An **job type code** describes the type of a job. There are two possible values:

1. `PLANNED`: the job is a planned job.
2. `RESPONSIVE`: the job is a responsive job.

2.4.5 Job Status Code

An **job status code** describes the status of an job. Possible values are:

1. `NEW`: the job is new and has not yet been assigned to a crew.
2. `PENDING`: the job has been assigned to a crew, but that crew have not yet arrived on-site.
3. `STARTED`: the crew have arrived on-site and commenced the job; the job may still be pre-empted.
4. `COMMITTED`: the job has commenced and cannot be pre-empted.

2.4.6 Assigned Job Object

An **assigned job** object describes the position of an assigned job in the daily schedule for a crew. It has the following fields:

1. `job_id`: the identifier of the job.
2. `start_time`: actual or estimated start date-time.
3. `finish_time`: actual or estimated finish date-time.

2.5 Dispatch Optimiser Output

2.5.1 Dispatch Response Object

The output from the dispatch optimiser is a **dispatch response** object. It has the following fields:

1. `client_id`: a string giving the user assigned identifier of the dispatch optimisation job. (This is the same as the `id` field of the dispatch request object with which the optimiser was invoked.) This field may be `null` if the `status` field is `ERROR` (e.g. if the input was misformed and no usable `id` could be found).
2. `new_job_id`: The unique identifier of the new job; will be `null` if the `status` field is `ERROR`.
3. `recommendations`: an array of dispatch recommendation objects describing the recommendations for the new responsive job; will be `null` if the `status` field is `ERROR`, `NO_COMPATIBLE_CREWS`, or `MISSING_REQUIRED_EQUIPMENT`.
4. `status`: a string giving one of the dispatch response status codes.
5. `errors`: if `status` is `ERROR` then this is an array of strings describing the errors encountered; it is `null` otherwise.
6. `warnings`: an array of strings describing potential problems.

2.5.2 Dispatch Recommendation Object

A **dispatch recommendation** object describes a recommended route for a crew. It has the following fields:

1. `crew_id`: the unique identifier of the crew.
2. `route`: an array of dispatch stop objects describing the recommended route for the vehicle.
3. `on_time`: a Boolean, `true` means the new responsive job was responded to on time.
4. `late`: the number of minutes that the new responsive job is late; `0` if the job is not late.
5. `displaced_jobs`: an array of job identifiers, each of which corresponds to an job that is removed from the crew's route by this recommendation; `null` if no jobs are removed.
6. `cost`: a number, giving the monetary cost for assigning the job to this crew. (This is the same as `cost` field of the allowed crew object for this job and crew in the input.)
7. `preference`: a number, giving the preference for assigning the job to this crew. (This is the same as `preference` field of the allowed crew object for this job and crew in the input.)

2.5.3 Dispatch Stop Object

A **dispatch stop** object describes a recommended stop for a crew. It has the following fields:

1. `job_id`: the unique identifier of the job.
2. `arrival_time`: the date-time of the arrival at the stop.
3. `departure_time`: the date-time of the departure from the stop.

3 Microsoft Excel Input

This section describes the Microsoft Excel input format for the optimiser.

Excel input must be either an Office Open XML workbook (`.xlsx`) or macro-enabled workbook (`.xlsm`). Office Open XML binary workbooks (`.xlsb`) are *not* currently supported. Workbooks in the old Excel binary file format (`.xls`) are *not* supported.

The workbook should contain the sheets described in the following sections. Those sheets that are specified as "required" must be present in the workbook. The workbook may contain sheets other than those recognised by the DTO-Excel format. Such extra sheets are ignored by the optimiser.

Unless otherwise noted, sheets recognised by the optimiser may contain columns other than those required by the DTO-Excel format. Such extra columns are ignored by the optimiser.

Columns recognised by the optimiser may appear in any order.

All sheets must have a non-empty first row.

The optimiser will evaluate any formulas used in the workbook when it is loaded. Formulas must *not* refer to sheets in other workbooks.

Not all Excel functions are supported by the optimiser; appendix D lists the unsupported Excel functions.

3.1 Identifiers in Excel

Identifiers in MWO-Excel may be provided as either text or numeric data. Identifiers given in text cells are taken *as-is* subject only to the rules concerning case and "white space" insensitivity. Identifiers given in numeric cells are *only* accepted by the optimiser if they are integer (i.e. their fractional part is zero). Formulas that evaluate to either text or (integer) numeric data may also be used to specify identifiers in Excel.

3.2 Times in Excel

In Excel, times are *stored* as decimal numbers, where `0` is `00:00:00` and `1` is `24:00:00`.

The integer component of the decimal represents a 24 hour period of time and is used to represent times greater than `23:59:59`. When using this representation, it is useful to think of the integer component as the day on which the time occurs, with times less than `24:00:00` occurring on *day zero*. For example, `1.50` would be `36:00`, or `12:00` on the second day.

In Excel, cells containing times may be formatted in various ways, for example, `hh:mm`, `d hh:mm` or `h:mm AM/PM`. Times may be given to the optimiser in any of the Excel time formats.

	hh:mm	**[hh];mm**	**dd/mm/yyy hh:mm**	**d hh:mm**
0.5	12:00	12:00	00/01/1900 12:00	0 12:00
1.5	12:00	36:00	01/01/1900 12:00	1 12:00

Table 2: Excel time formatting examples.

Note here that if using the `dd/mm/yyy hh:mm` format, *day zero* occurs on `00/01/1900`, which is Excel's earliest recognised date.

3.3 Dates

In Excel, dates are *stored* as integers, where `1` represents `1/1/1900 12:00:00 a.m`.

In Excel, dates may be formatted in various ways, for example, *dd/mm/yyyy*. Dates may be given to the optimiser in any of the Excel date formats.

3.4 Date-Times

In Excel, dates (integers) and times (decimal fractions) can be combined to express a date-time. For example, `32331.06` represents the date-time `7/7/1988 1:26:24 a.m`.

In Excel, date-times may be formatted in various ways, for example, *ddd dd/mm/hh:mm*, *ddd mm/dd h:mm*, *ddd hh:mm* etc.[2].

[2]For further detail on formatting dates and times in Excel, as well as their combination, see https://support.office.com/en-us/article/format-numbers-as-dates-or-times-418bd3fe-0577-47c8-8caa-b4d30c528309

3.5 Sheet: General

This sheet is required.

3.5.1 Column: Parameter

Required: Yes.
Type: Text.
Description: Parameters that control the settings of scenario.

3.5.2 Column: Value

Required: Yes.
Type: Text or Number.
Description: Values of the parameters. Can be a number or text depending on the parameter.

3.5.3 Parameter: Id

Required: No.
Value Type: Text.
Default: No id.
Description: A name for the scenario.

3.5.4 Parameter: Name

Required: No.
Value Type: Text.
Default: No name.
Description: A name for the scenario.

3.5.5 Parameter: Owner

Required: No.
Value Type: Text.
Default: No owner.
Description: The client name for the scenario.

3.5.6 Parameter: Time Zone

Required: No.
Value Type: Text.
Default: Specific to the deployment.
Description: The time zone for the scenario.

3.5.7 Parameter: Start Date

Required: Yes.
Value Type: Excel date.
Description: The first day of the planning period.

3.5.8 Parameter: Finish Date

Required: Yes.
Value Type: Excel date.
Description: The last day of the planning period.

3.5.9 Parameter: Next Day

Required: No.
Value Type: Excel date.
Default: No next day.
Description: A date identifying the next day from which rescheduling is to occur. This date must be inside the start and finish dates defined by "Start Date" and "Finish Date" date respectively. For the "plan" mode of operation this field must not be present.

3.5.10 Parameter: Iterations

Required: No.
Type: Number.
Default: `3000`.
Description: The number of iterations that the optimiser should run for.

3.5.11 Parameter: Segment Fixed Penalty

Required: No.
Type: Number.
Default: `0.0`: No penalty.
Description: The default visit segment fixed penalty.
Note: This is overridden by the crew specific value if present.

3.5.12 Parameter: Segment Distance Penalty

Required: No.
Type: Number.
Default: `0.0`: No penalty.
Description: The default visit segment distance penalty.
Note: This is overridden by the crew specific value if present.

3.6 Sheet: Time Windows

The **Time Windows** sheet may be used to define groups of time windows that can be associated with crew rosters or job times.

This sheet can be omitted if no named groups of time windows are required.

3.6.1 Column: Id

Required: Yes.
Type: Text or (integer) number.
Description: An identifier giving the named group of time windows that this window is part of.

3.6.2 Column: Start

Required: Yes.
Type: Excel time.
Description: Starting time of the window. Must occur before the End time and be nonnegative.

3.6.3 Column: End

Required: Yes.
Type: Excel time.
Description: Finishing time of the window. Must occur after the Start time and be nonnegative.

3.7 Sheet: Locations

The **locations** sheet is used to define the locations for crews.

Each row represents an individual location.

This sheet is required if the crew locations sheet is present or the "Start Location" "Finish Location" columns are present in the Crews sheet.

3.7.1 Column: Id

Required: Yes.
Type: Text or (integer) number.
Description: An identifier for this crew.

3.7.2 Column: Latitude

Required: Yes.
Type: Number.
Description: The latitude of this location.

3.7.3 Column: Longitude

Required: Yes.
Type: Number.
Description: The longitude of this location.

3.8 Sheet: Crews

The **Crews** sheet is used to define the crews.

Each row represents an individual crew.

This sheet is required.

3.8.1 Column: Id

Required: Yes.
Type: Text or (integer) number.
Description: An identifier for this crew.

3.8.2 Column: Name

Required: Yes.
Type: Text.
Description: The name of this crew.

3.8.3 Column: Start Time

Required: Yes.
Type: Excel time.
Description: The default start time for this crew.

3.8.4 Column: Finish Time

Required: Yes.
Type: Excel time.
Description: The default finish time for this crew.

3.8.5 Column: Start Location

Required: Yes.
Type: Text or (integer) number.
Description: The identifier of the crew's start location as defined in the Locations sheet.

3.8.6 Column: Finish Location

Required: Yes.
Type: Text or (integer) number.
Description: The identifier of the crew's finish location as defined in the Locations sheet.

3.8.7 Column: Cost per Use

Required: No.
Type: Number.
Default: `0`.
Description: A fixed cost for using the crew.

3.8.8 Column: Cost per Hour

Required: No.
Type: Number.
Default: `0`.
Description: Cost incurred per hour of crew's work time.

3.8.9 Column: Cost per Kilometre

Required: No.
Type: Number.
Default: `0`.
Description: Cost incurred per kilometre of transit distance.

3.8.10 Column: Hidden Cost per Use

Required: No.
Type: Number.
Default: `0`.
Description: Hidden "cost" for using the crew. This is not included in the cost calculation for the solution and could be used to prioritise certain crews.

3.8.11 Column: Hidden Cost per Hour

Required: No.
Type: Number.
Default: `0`.
Description: Additional cost incurred per hour of crew use. This is not included in the cost calculation for the solution and could be used to prioritise crews.

3.8.12 Column: Hidden Cost per Kilometre

Required: No.
Type: Number.
Default: `0`.
Description: Additional cost incurred per km of crew use. This is not included in the cost calculation for the solution and could be used to prioritise certain crews.

3.8.13 Column: Visit Segment Distance Penalty

Required: No.
Type: Number.
Default: `0`.
Description: The per-km distance component of the visit segment penalty for this crew.

3.8.14 Column: Visit Segment Fixed Penalty

Required: No.
Type: Number.
Default: `0`.
Description: The fixed component of the visit segment penalty for this crew.

3.8.15 Column: Subcontractor

Required: No.
Value Type: Text (Boolean).
Default: `FALSE`.
Description: If `TRUE`, then this crew is a subcontractor.

3.8.16 Column: Overtime

Required: No.
Value Type: Text (Boolean).
Default: `FALSE`.
Description: If `TRUE`, then this crew can perform overtime.

3.9 Sheet: Crew Locations

The **Crew Locations** sheet is used to define the start and finish locations for a crew on a given date.

Each row represents an individual crew.

This sheet is only required if the crews do not always start and a finish at the locations defined on the Crews sheet.

3.9.1 Column: Crew

Required: Yes.
Type: Text or (integer) number.
Description: The identifier of a crew.

3.9.2 Column: Date

Required: Yes.
Type: Excel date.
Description: The date when this crew uses these locations.

3.9.3 Column: Start Location

Required: Yes.
Type: Text or (integer) number.
Description: The identifier of the crew's start location as defined in the Locations sheet.

3.9.4 Column: Finish Location

Required: Yes.
Type: Text or (integer) number.
Description: The identifier of the crew's finish location as defined in the Locations sheet.

3.10 Sheet: Roster

This sheet specifies the availability of the crews on a given date.

This sheet is not required and if it is not present, the crew will be available on each day of the scheduling period between its start and finish times.

3.10.1 Column: Id

Required: Yes.
Type: Text.
Description: The identifiers for the crews are listed in this column starting from row `2`.

3.10.2 Column: All Other Columns

The remaining columns are used to define work times for the crew on a given date. The dates are present in the top row, with the work time for a crew on this date are located in the relevant row and column.

These cells can have the following values:

- Empty: This crew cannot work on this day.
- `false`: This crew cannot work on this day.
- `true`: This crew can work using the default start and finish time window as per the crews sheet.
- `Time Window Id:` The cell contains the identifier that matches one of the codes defined in Section 3.6.

Note: An empty cell and a cell containing `false` both indicate the crew cannot work on that date.

An example of sheet can be seen in Table 3.

Id	2018/02/01	2018/02/02	2018/02/03	2018/02/04	2018/02/05
Crew 1	TRUE	AFTERNOON	FALSE	ALL	ALL
Crew 2	MORNING	TRUE	AFTERNOON	ALL	
Crew 3	AFTERNOON	AFTERNOON	ALL	ALL	ALL
Crew 4			FALSE	MORNING	MORNING

Table 3: Crew Roster.

3.11 Sheet: Jobs

The **Jobs** sheet is used to define the jobs.

Each row represents an individual job.

This sheet is required.

3.11.1 Column: Id

Required: Yes.
Type: Text or (integer) number.
Description: An identifier for this job.

3.11.2 Column: Name

Required: Yes.
Type: Text.
Description: The name of this job.

3.11.3 Column: Latitude

Required: Yes.
Type: Number.
Description: The latitude component of the location for this job.

3.11.4 Column: Longitude

Required: Yes.
Type: Number.
Description: The longitude component of the location for this job.

3.11.5 Column: Default Time Window

Required: Yes.
Type: Text.
Description: The identifier for the default time window for this job.

3.11.6 Column: Default Duration

Required: No, as long as Job Durations sheet is present and contains a valid entry for each compatible crew.
Type: Number.
Description: The default duration of this job on any crew. Overridden by the Job Durations sheet crew specific duration if present.

3.11.7 Column: Default Cost

Required: No, as long as Job Costs sheet is present and contains a valid entry for each compatible crew.
Type: Number.
Description: The default cost of this job on any crew. Overridden by the Job Costs sheet crew specific costs if present.

3.11.8 Column: Required Equipment

Required: No.
Type: Text.
Default: No equipment required.
Description: The equipment required to complete the job.
Note: Multiple equipment requirements must be separated by semicolons.

3.12 Sheet: Job Times

This sheet defines the time windows when a job can be serviced on each day of the roster.

This sheet is not required and if it is not present, the job can be serviced on each day of the scheduling period between during its default time window.

3.12.1 Column: Id

Required: Yes.
Type: Text.
Description: The identifiers for the jobs are listed in this column starting from row `2`.

3.12.2 Column: All Other Columns

The remaining columns are used to define the time window for the job on a given date. The dates are present in the top row, with the time window for a job on this date are located in the relevant row and column.

These cells can have the following values:

- Empty: This job cannot be serviced on this day.
- `false`: This job cannot be serviced on this day.
- `true`: This job can be serviced between the default start and finish time window as per the jobs sheet.
- `Time Window Id:` The cell contains the identifier that matches one of the codes defined in Section 3.6.

Note: An empty cell and a cell containing `false` both indicate the job cannot serviced on that date.

An example of sheet can be seen in Table 4.

Id	2018/02/01	2018/02/02	2018/02/03	2018/02/04	2018/02/05
Job 1	TRUE	AFTERNOON	FALSE	ALL	ALL
Job 2	MORNING	TRUE	AFTERNOON	ALL	
Job 3	AFTERNOON	AFTERNOON	ALL	ALL	ALL
Job 4			FALSE	MORNING	MORNING

Table 4: Crew Roster.

3.13 Sheet: Job Durations

This sheet defines the amount of time required to service a job for each crew.

This sheet is not required and if it is not present, the job requires the amount of time defined by default duration for each crew.

Note: If this sheet is not present, a default duration must be defined for each job.

3.13.1 Column: Id

Required: Yes.
Type: Text.
Description: The identifiers for the jobs are listed in this column starting from row 2.

3.13.2 Column: All Other Columns

The remaining columns are used to define the duration for the job if serviced by a given crew. The crew identifiers are present in the top row, with the duration for a job to be serviced by this crew are located in the relevant row and column.

These cells can have the following values:

- Empty: This job cannot be serviced by this crew.
- An excel time describing the service duration for this job by this crew.

An example of sheet can be seen in Table 5.

Id	Crew 1	Crew 2	Crew 3	Crew 4	Crew 5
Job 1	02:15	03:51			01:13
Job 2	01:43		03:58	01:59	
Job 3	03:30	03:42	02:56	02:15	03:16
Job 4	00:55	03:08		02:20	01:47

Table 5: Crew Roster.

3.14 Sheet: Job Costs

This sheet defines the cost to service a job for each crew.

This sheet is not required and if it is not present, the job costs the amount defined by default cost for each crew.

Note: If this sheet is not present, a default cost must be defined for each job.

3.14.1 Column: Id

Required: Yes.
Type: Text.
Description: The identifiers for the jobs are listed in this column starting from row 2.

3.14.2 Column: All Other Columns

The remaining columns are used to define the cost incurred if the job is serviced by a given crew. The crew identifiers are present in the top row, with the costs are located in the relevant row and column.

These cells can have the following values:

- Empty: This job cannot be serviced by this crew.
- An cost incurred if this crew services this job.

An example of sheet can be seen in Table 6.

Id	Crew 1	Crew 2	Crew 3	Crew 4	Crew 5
Job 1	176.00	113.00			135.00
Job 2	119.00		190.00	150.00	
Job 3	156.00	234.00	125.00	220.00	202.00
Job 4	150.00	37.00		170.00	173.00

Table 6: Crew Roster.

3.15 Sheet: Job Crew Compatibility

This sheet defines the which crews can service a job. It can be used in partnership with the Job Crew List sheet.

This sheet is not required and if both it and the Job Crew List sheet are not present then it is assumed the job is compatibly with each crew.

3.15.1 Column: Job

Required: Yes.
Type: Text.
Description: The identifiers for the jobs are listed in this column starting from row 2.

3.15.2 Column: All Other Columns

The remaining columns are used to define the compatibility relationship between the job and a given crew. The crew identifiers are present in the top row, with the compatibility boolean values are located in the relevant row and column.

These cells can have the following values:

- Empty: This job cannot be serviced by this crew.
- `false`: This job cannot be serviced by this crew.
- `true`: This job can be serviced by this crew.

An example of sheet can be seen in Table 7.

Job	Crew 1	Crew 2	Crew 3	Crew 4	Crew 5
Job_1	TRUE	TRUE			TRUE
Job_2		FALSE	TRUE	TRUE	TRUE
Job_3		TRUE	TRUE	FALSE	TRUE
Job_4		TRUE		TRUE	

Table 7: Job Crew Compatibility.

3.16 Sheet: Job Crew List

This sheet defines the which crews can service a job. It can be used in partnership with the Job Crew Compatibility sheet.

This sheet is not required and if both it and the Job Crew Compatibility sheet are not present then it is assumed the job is compatibly with each crew.

3.16.1 Column: Job

Required: Yes.
Type: Text.
Description: The identifiers for the jobs are listed in this column starting from row 2.

3.16.2 Column: Crew

Required: Yes.
Type: Text.
Description: The identifiers for the crew are listed in this column starting from row 2.

An example of sheet can be seen in Table 8.

Job	Crew
Job_1	Crew 1
Job_2	Crew 1
Job_3	Crew 1
Job_4	Crew 1

Table 8: Job Crew List.

3.17 Sheet: Precedences

The **Precedences** sheet is used to define the precedences between jobs.

Each row represents an precedence rule between two jobs.

This sheet is not required if there are no precedences.

3.17.1 Column: before

Required: Yes.
Type: Text or (integer) number.
Description: The identifier for the before job.

3.17.2 Column: after

Required: Yes.
Type: Text or (integer) number.
Description: The identifier for the after job.

3.17.3 Column: Minimum Gap

Required: No.
Type: Excel time.
Default: No minimum gap.
Description: The minimum gap between these jobs.

3.17.4 Column: Maximum Gap

Required: No.
Type: Excel time.
Default: No maximum gap.
Description: The maximum gap between these jobs.

3.17.5 Column: Same Crew

Required: No.
Value Type: Text (Boolean).
Default: False, there is no requirement that both jobs must be serviced by the same crew.
Description: These jobs must be serviced by the same crew.

3.18 Sheet: Assignments

The **Assignment** sheet is used to define any existing assignment of jobs to crews.

Each row represents an assignment of a job to a crew.

This sheet is not required if there are no assignments.

3.18.1 Column: Job

Required: Yes.
Type: Text or (integer) number.
Description: The identifier for the job.

3.18.2 Column: Crew

Required: Yes.
Type: Text or (integer) number.
Description: The identifier for the crew.

3.18.3 Column: Start Time

Required: Yes.
Type: Excel date time.
Description: The time the job is planned to start.

3.18.4 Column: Finish Time

Required: Yes.
Type: Excel date time.
Description: The time the job is planned to finish.

3.18.5 Column: Flex

Required: Yes.
Type: Number.
Description: A number between `0` and `1` (inclusive), which is a measure of how much the optimiser may change the date of an assigned job. A value of `0` means that the job *must* be assigned on the date given by the **start_time** field; a value of `1` means the optimiser is completely free to reschedule the job; values between `0` and `1` control how close to the originally assigned date the job can be rescheduled.

4 Microsoft Excel Output

This description describes the Microsoft Excel output format for the optimiser.

The Excel output will contain references (e.g. crew identifiers, job identifiers) to the request from which it was generated.

The optimiser generates output workbooks in Office OpenXML format (`.xlsx`).

4.1 Sheet: Solution Summary

This sheet contains the various KPIs for the solution.

Each row contains a separate KPI.

4.1.1 Scenario Id

Type: Text.
Description: The identifier of the scenario from which the solution was generated. Matches the `name` field of the General sheet.
Default: If the relevant input was not defined then this will display `<not set>`.

4.1.2 Scenario Name

Type: Text.
Description: The name of the scenario from which the solution was generated. Matches the `name` field of the General sheet.
Default: If the relevant input was not defined then this will display `<not set>`.

4.1.3 Scenario Owner

Type: Text.
Description: The owner of the scenario from which the solution was generated. Matches the `name` field of the General sheet.
Default: If the relevant input was not defined then this will display `<not set>`.

4.1.4 Total Cost

Type: Number.
Description: The total cost for all crews in the solution.

4.1.5 Total Distance

Type: Number.
Description: The total transit distance of all crews in the solution.

4.1.6 Assigned Crew Shifts

Type: Number.
Description: The total number of shifts worked by the crews in the solution.

4.1.7 Total Crew Shifts

Type: Number.
Description: The total number of available shifts for the crews in the scenario.

4.1.8 Assigned Jobs

Type: Number.
Description: The total number of jobs assigned in the solution.

4.1.9 Total Jobs

Type: Number.
Description: The total number of jobs in the scenario.

4.2 Sheet: Assignments

This sheet contains the list of job assignments to the crews.

Each row in the sheet represents a job assignment to a crew.

4.2.1 Crew

Type: Text.
Description: The identifier of the crew.

4.2.2 Job

Type: Text.
Description: The identifier of the job.

4.2.3 Column: Start Time

Type: Excel date time.
Description: The time the job is scheduled to start.

4.2.4 Column: Finish Time

Type: Excel date time.
Description: The time the job is scheduled to finish.

4.3 Sheet: Unassigned

This sheet contains the list of jobs which could not be assigned during optimisation.

Each row in the sheet represents an unassigned job.

4.3.1 Job

Type: Text.
Description: The identifier of the job.

4.3.2 Column: Latitude

Type: Number.
Description: The latitude component of the location for this job.

4.3.3 Column: Longitude

Type: Number.
Description: The longitude component of the location for this job.

4.3.4 Column: Required Equipment

Type: String.
Description: The list of equipment required for this job. Each item of equipment appears on a separate line.

5 Planning Web Service

This section describes the planning optimiser web service. Using the web service, clients *submit* an instance (in the form of a plan request object) to the optimiser. The instance is optimised in the background and clients can *query* its state as it runs. When optimisation of the scenario is complete clients can *fetch* the results.

The web service runs over HTTP/1.1 and has the following endpoints:

1. SUBMIT-PLAN – used to submit a new planning optimisation task.
2. QUERY-PLAN – used to query the state of a previously submitted planning optimisation task.
3. CANCEL-PLAN – used to cancel a previously submitted planning optimisation task.
4. FETCH-PLAN – used to retrieve the results of a completed planning optimisation task.
5. CHECK-PLAN – used to validate the input to the planning optimiser.

Opturion will provide authentication credentials for use with the web service if required.

5.1 Planning Task Identifiers

When a planning instance is submitted to the planning optimiser (see Section 5.2) it is assigned a unique **task identifier** that is used to identify it over the lifetime of the optimisation. Task identifiers take the form of a universally unique identifier (UUID) in canonical form [3].

5.2 Submit-Plan Endpoint

The SUBMIT-PLAN endpoint is used to submit planning optimisation tasks to the optimiser.

5.2.1 Submit-Plan Requests

A SUBMIT-PLAN request is an HTTP POST request to a URL of the following form:

`https://`*server-name*`/mwo/v2/submit-plan`

where *server-name* will be provided by Opturion.

The body of the POST request is a JSON document containing a plan request object.

5.2.2 Submit-Plan Request Example

Here is an example URL for a SUBMIT-PLAN request:

```
https://demo.opturion.com/mwo/v2/submit-plan
```

On success (see below), a SUBMIT-PLAN request will return with HTTP status code 202 (ACCEPTED) and the response body will contain a JSON encoded plan submit response object. This object will contain the newly assigned task identifier. The task identifier can be used with the QUERY-PLAN endpoint to examine the current state of the job.

Here is an example of a plan submit response object:

[3] https://www.ietf.org/rfc/rfc4122.txt

```
{
  "status": "OK",
  "task_id": "40a56579-bc85-4f26-aa37-d83332815b2d"
}
```

5.2.3 Submit-Plan Responses

The following table lists of all of the HTTP status codes that SUBMIT-PLAN specifically returns and their meanings. Any other HTTP status code returned by SUBMIT-PLAN that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
202 (ACCEPTED)	Task has been accepted and scheduled to run.	A JSON plan submit response object giving the task identifier for the newly accepted job.
400 (BAD REQUEST)	Errors encountered in input data.	A JSON plan submit response object describing the errors.
401 (UNAUTHORIZED)	Authentication (if enabled) failed; the user is not authorized to access the optimiser.	Empty.
429 (TOO MANY REQUESTS)	Rate limiting has been applied by the server.	Empty.
500 (INTERNAL SERVER ERROR)	An error occurred on the server.	The body *may* contain a JSON object further describing the problem.

Note that in the case where rate limiting has been applied, clients should wait and then re-submit the optimisation task at a later time.

5.3 Query-Plan Endpoint

The QUERY-PLAN endpoint is used to check the state of a previously submitted planning optimisation task. It returns a short description of the task's current progress or the best solution found so far.

5.3.1 Query-Plan Requests

A QUERY-PLAN request takes the form of an HTTP GET request to a URL of the following form:

`https://`*server-name*`/mwo/v2/query-plan/`*task-identifier*`?`*parameters*

where *server-name* will be provided by Opturion, *task-identifier* is the task identifier returned by a successful call to the SUBMIT-PLAN service and *parameters* is described below.

5.3.2 Query-Plan Request Parameters

The QUERY-PLAN endpoint accepts one optional parameter. It is:

1. **what** – this should be one of "`progress`" or "`solution`". If this parameter is not present then it defaults to "`progress`".

5.3.3 Query-Plan Request Example

Here is an example QUERY-PLAN request URL:

```
https://demo.opturion.com/mwo/v2/query-plan/9046b6e9-7d7d-4453-a935-d331edb268bb
```

5.3.4 Query-Plan Responses

The following table lists of all of the HTTP status codes that QUERY-PLAN specifically returns and their meaning. Any other HTTP status code returned by QUERY-PLAN that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
200 (OK)	Task status information has been returned.	For "`progress`" queries a JSON plan query response object describing the current progress. For "`solution`" queries a plan fetch response object if a solution has been found or empty otherwise.
401 (UNAUTHORIZED)	Authentication (if enabled) failed; the user is not authorized to access the optimiser.	Empty.
404 (NOT FOUND)	A task matching the given task identifier *and* owned by the authenticated user could not be found.	Empty.
429 (TOO MANY REQUESTS)	May be returned if rate limiting has been applied by the server.	Empty.
500 (INTERNAL SERVER ERROR)	Indicates that an error occurred on the server.	The body *may* contain a JSON object further describing the error.

Note that in the case where rate limiting has been applied, clients should wait and then query the task status again at a later time.

In order to avoid rate limiting, it is recommended that clients leave at least `500` milliseconds or more between consecutive QUERY-PLAN requests.

5.4 Cancel-Plan Endpoint

The CANCEL-PLAN endpoint is used to cancel previously submitted planning tasks.

5.4.1 Cancel-Plan Requests

A CANCEL-PLAN request takes the form of an HTTP DELETE request to a URL of the following form:

`https://`*server-name*`/mwo/v2/cancel-plan/`*task-identifier*

where *server-name* will be provided by Opturion and *task-identifier* is the task identifier returned by a successful call to the SUBMIT-PLAN service.

5.4.2 Cancel-Plan Request Example

Here is an example CANCEL-PLAN request URL:

```
https://demo.opturion.com/mwo/v2/cancel-plan/9046b6e9-7d7d-4453-a935-d331edb268bb
```

5.4.3 Cancel-Plan Responses

The following table lists of all of the HTTP status codes that CANCEL-PLAN specifically returns and their meaning. Any other HTTP status code returned by CANCEL-PLAN that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
200 (OK)	The task has been cancelled.	Empty.
401 (UNAUTHORIZED)	Authentication (if enabled) failed; the user is not authorized to access the optimiser.	Empty.
404 (NOT FOUND)	A task matching the given task identifier and owned by the authenticated user could not be found.	Empty.
429 (TOO MANY REQUESTS)	May be returned if rate limiting has been applied by the server.	Empty.
500 (INTERNAL SERVER ERROR)	Indicates that an error occurred on the server.	The body *may* contain a JSON object further describing the error.

5.5 Fetch-Plan Endpoint

The FETCH-PLAN endpoint is used to retrieve the result of a completed planning optimisation task.

5.5.1 Fetch-Plan Requests

A FETCH-PLAN request takes the form of an HTTP GET request to a URL of the following form:

`https://`*server-name*`/mwo/v2/fetch-plan/`*task-identifier*

where *server-name* will be provided by Opturion and *task-identifier* is the task identifier returned by SUBMIT-PLAN.

5.5.2 Fetch-Plan Request Example

Here is an example FETCH-PLAN request:

```
https://demo.opturion.com/mwo/v2/fetch-plan/9046b6e9-7d7d-4453-a935-d331edb268bb
```

5.5.3 Fetch-Plan Responses

The following table lists of all of the HTTP status codes that FETCH-PLAN specifically returns and their meaning. Any other HTTP status code returned by FETCH-PLAN that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
200 (OK)	Optimisation result has been returned.	A JSON plan fetch response object.
401 (UNAUTHORIZED)	Authentication (if enabled) failed; the user is not authorized to access the optimiser.	Empty.
404 (NOT FOUND)	A task matching the given task identifier and owned by the authenticated user could not be found.	Empty.
429 (TOO MANY REQUESTS)	May be returned if rate limiting has been applied by the server.	Empty.
500 (INTERNAL SERVER ERROR)	Indicates that an error occurred on the server.	The body *may* contain a JSON object further describing the error.

After the result has been retrieved using FETCH-PLAN, it will be deleted from the server and subsequent FETCH-PLAN requests for the same task identifier will return HTTP status `404`.

Depending on the service configuration, optimisation results that have not been retrieved will be deleted after a given time elapses. (Opturion will provide details of this configuration.)

5.6 Check-Plan Endpoint

The CHECK-PLAN endpoint is used to validate the inputs given to the optimiser for planning optimisation tasks.

5.6.1 Check-Plan Requests

A CHECK-PLAN request is an HTTP POST to a URL of the following form:

`https://`*server-name*`/mwo/v2/check-plan`

where *server-name* will be provided by Opturion.

The body of the POST request is a JSON document containing a plan request object.

5.6.2 Check-Plan Request Example

Here is an example URL for a CHECK-PLAN request:

```
https://demo.opturion.com/mwo/v2/check-plan
```

On success (see below) a CHECK-PLAN request will return with HTTP status code `200` (OK) and the response body will contain a JSON encoded check response object describing any errors in the input request.

Here is an example of a check response object:

```
{
   "status": "OK"
}
```

5.6.3 Check-Plan Responses

The following table lists of all of the HTTP status codes that CHECK-PLAN specifically returns and their meanings. Any other HTTP status code returned by CHECK-PLAN that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
200 (OK)	The input request has been checked for errors.	A JSON check response object describing if the input was valid or not.
401 (UNAUTHORIZED)	Authentication (if enabled) failed; the user is not authorized to access the optimiser.	Empty.
429 (TOO MANY REQUESTS)	Rate limiting has been applied by the server.	Empty.
500 (INTERNAL SERVER ERROR)	An error occurred on the server.	The body *may* contain a JSON object further describing the problem.

Note that in the case where rate limiting has been applied, clients should wait and then re-submit at a later time.

6 Dispatch Web Service

This section describes the dispatch optimiser web service.

This web service runs over HTTP/1.1 and has two endpoints. These are:

1. DISPATCH – used to send jobs to the dispatch optimiser; responses are returned over the *same* connection.
2. CHECK-DISPATCH – used to validate the dispatch input to the optimiser.

Opturion will provide authentication credentials for use with the web service if required.

6.1 Dispatch Endpoint

The DISPATCH endpoint is used to submit dispatch optimisation jobs to the optimiser.

6.1.1 dispatch Requests

A DISPATCH request takes the from of an HTTP POST request to a URL for the following form:

`https://`*server-name*`/mwo/v2/dispatch/`

where *server-name* will be provided by Opturion.

The body of the POST request is a JSON document containing a dispatch request object.

6.1.2 dispatch Request Parameters

The DISPATCH endpoint accepts no parameters.

6.1.3 dispatch Request Example

Here is an example DISPATCH request URL:

```
https://demo.opturion.com/mwo/v2/dispatch
```

On success (see below) a DISPATCH request will return with HTTP status code `200` (OK) and the response body will contain a JSON encoded dispatch response object.

6.1.4 dispatch Responses

The following table lists of all of the HTTP status codes that DISPATCH specifically returns and their meaning. Any other HTTP status code returned by DISPATCH that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
200 (OK)	The job was accepted and successfully run.	A JSON dispatch response containing a set of assignment recommendations.
401 (Unauthorized)	Indicates that authentication (if enabled) failed and the user is not authorized to access the MWO.	Empty.
429 (Too Many Requests)	May be returned if rate limiting has been applied by the server.	Empty.
500 (Internal Server Error)	Indicates that an error occurred on the server.	The body *may* contain a JSON object further describing the error.

6.2 Check-Dispatch Endpoint

The CHECK-DISPATCH endpoint is used to validate the inputs given to the optimiser for dispatch optimisation jobs.

6.2.1 Check-Dispatch Requests

A CHECK-DISPATCH request is an HTTP POST to a URL of the following form:

`https://`*server-name*`/mwo/v2/check-dispatch`

where *server-name* will be provided by Opturion.

The body of the POST request is a JSON document containing a dispatch request object.

6.2.2 Check-Dispatch Request Example

Here is an example URL for a CHECK-DISPATCH request:

```
https://demo.opturion.com/mwo/v2/check-dispatch
```

On success (see below) a CHECK-PLAN request will return with HTTP status code `200` (OK) and the response body will contain a JSON encoded check response object describing any errors in the input request.

Here is an example of a check response object:

```
{
  "status": "OK"
}
```

6.2.3 Check-Dispatch Responses

The following table lists of all of the HTTP status codes that CHECK-DISPATCH specifically returns and their meanings. Any other HTTP status code returned by CHECK-DISPATCH that is not covered by this table has its usual meaning.

Status Code	Description	Response Body
200 (OK)	The input request has been checked for errors.	A JSON check response object describing if the input was valid or not.
401 (UNAUTHORIZED)	Authentication (if enabled) failed; the user is not authorized to access the optimiser.	Empty.
429 (TOO MANY REQUESTS)	Rate limiting has been applied by the server.	Empty.
500 (INTERNAL SERVER ERROR)	An error occurred on the server.	The body *may* contain a JSON object further describing the problem.

Note that in the case where rate limiting has been applied, clients should wait and then re-submit the job at a later time.

Appendices

A Planning Start-Of-Month Input Example


```
{
  "id": "78f91273-32a4-4914-91eb-9e6e752c2184",
  "start_date": "2021-05-01",
  "finish_date": "2021-05-31",
  "next_day": null,
  "crews": [
    {
      "id": "Crew1",
      "start_time": "7:00",
      "finish_time": "18:00",
      "roster": [
        {
          "start_time": "2021-05-01T07:00",
          "finish_time": "2021-05-01T18:00"
        },
        {
          "start_time": "2021-05-02T07:00",
          "finish_time": "2021-05-02T18:00"
        },
        {
          "start_time": "2021-05-03T07:00",
          "finish_time": "2021-05-03T18:00"
        },
        {
          "start_time": "2021-05-04T07:00",
          "finish_time": "2021-05-04T18:00"
        },
        {
          "start_time": "2021-05-05T07:00",
          "finish_time": "2021-05-05T18:00"
        },
        {
          "start_time": "2021-05-06T07:00",
          "finish_time": "2021-05-06T18:00"
        },
        {
          "start_time": "2021-05-07T07:00",
          "finish_time": "2021-05-07T18:00"
        },
        {
          "start_time": "2021-05-08T07:00",
          "finish_time": "2021-05-08T18:00"
        },
        {
```

```
        "start_time": "2021-05-09T07:00",
        "finish_time": "2021-05-09T18:00"
      },
      {
        "start_time": "2021-05-10T07:00",
        "finish_time": "2021-05-10T18:00"
      },
      {
        "start_time": "2021-05-15T07:00",
        "finish_time": "2021-05-15T18:00"
      },
      {
        "start_time": "2021-05-16T07:00",
        "finish_time": "2021-05-16T18:00"
      },
      {
        "start_time": "2021-05-17T07:00",
        "finish_time": "2021-05-17T18:00"
      },
      {
        "start_time": "2021-05-18T07:00",
        "finish_time": "2021-05-18T18:00"
      },
      {
        "start_time": "2021-05-19T07:00",
        "finish_time": "2021-05-19T18:00"
      },
      {
        "start_time": "2021-05-20T07:00",
        "finish_time": "2021-05-20T18:00"
      }
    ],
    "start_location": {
      "longitude": 144.966167,
      "latitude": -37.82261
    },
    "finish_location": {
      "longitude": 144.966167,
      "latitude": -37.82261
    },
    "is_subcontractor": false,
    "allowed_overtime": true
  },
  {
    "id": "Crew2",
    "start_time": "7:00",
    "finish_time": "18:00",
    "roster": [
      {
```

```
  "start_time": "2021-05-01T07:00",
  "finish_time": "2021-05-01T18:00"
},
{
  "start_time": "2021-05-02T07:00",
  "finish_time": "2021-05-02T18:00"
},
{
  "start_time": "2021-05-03T07:00",
  "finish_time": "2021-05-03T18:00"
},
{
  "start_time": "2021-05-04T07:00",
  "finish_time": "2021-05-04T18:00"
},
{
  "start_time": "2021-05-05T07:00",
  "finish_time": "2021-05-05T18:00"
},
{
  "start_time": "2021-05-06T07:00",
  "finish_time": "2021-05-06T18:00"
},
{
  "start_time": "2021-05-07T07:00",
  "finish_time": "2021-05-07T18:00"
},
{
  "start_time": "2021-05-08T07:00",
  "finish_time": "2021-05-08T18:00"
},
{
  "start_time": "2021-05-09T07:00",
  "finish_time": "2021-05-09T18:00"
},
{
  "start_time": "2021-05-10T07:00",
  "finish_time": "2021-05-10T18:00"
},
{
  "start_time": "2021-05-15T07:00",
  "finish_time": "2021-05-15T18:00"
},
{
  "start_time": "2021-05-16T07:00",
  "finish_time": "2021-05-16T18:00"
},
{
  "start_time": "2021-05-17T07:00",
```

```
        "finish_time": "2021-05-17T18:00"
      },
      {
        "start_time": "2021-05-18T07:00",
        "finish_time": "2021-05-18T18:00"
      },
      {
        "start_time": "2021-05-19T07:00",
        "finish_time": "2021-05-19T18:00"
      },
      {
        "start_time": "2021-05-20T07:00",
        "finish_time": "2021-05-20T18:00"
      }
    ],
    "start_location": {
      "longitude": 144.966167,
      "latitude": -37.82261
    },
    "finish_location": null,
    "is_subcontractor": true,
    "allowed_overtime": false
  }
],
"jobs": [
  {
    "id": "Job1",
    "location": {
      "longitude": 144.966167,
      "latitude": -37.82261
    },
    "allowed_crews": [
      {
        "crew_id": "Crew2",
        "duration": 200,
        "preference": 100,
        "cost": 80
      },
      {
        "crew_id": "Crew1",
        "duration": 180,
        "preference": 0,
        "cost": 40
      }
    ],
    "required_equipment": [
      "EDUCTOR",
      "CRANE"
    ],
```

```
      "time_windows": [
        {
          "start": "2021-05-01T10:00",
          "finish": "2021-05-01T12:00"
        },
        {
          "start": "2021-05-03T11:00",
          "finish": "2021-05-03T13:00"
        }
      ]
    },
    {
      "id": "Job2",
      "location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "allowed_crews": [
        {
          "crew_id": "Crew2",
          "duration": "255",
          "preference": 0,
          "cost": 110
        }
      ],
      "required_equipment": []
    }
  ],
  "assignments": [
    {
      "crew_id": "Crew2",
      "job_id": "Job2",
      "start_time": "2021-05-01T09:00",
      "finish_time": "2021-05-01T12:00"
    }
  ]
}
```

B Planning Nightly Reschedule Input Example


```
{
  "id": "78f91273-32a4-4914-91eb-9e6e752c2184",
  "start_date": "2021-05-01",
  "finish_date": "2021-05-31",
  "next_day": "2021-05-02",
  "crews": [
    {
      "id": "Crew1",
      "start_time": "7:00",
      "finish_time": "18:00",
      "roster": [
        {
          "start_time": "2021-05-01T07:00",
          "finish_time": "2021-05-01T18:00"
        },
        {
          "start_time": "2021-05-02T07:00",
          "finish_time": "2021-05-02T18:00"
        },
        {
          "start_time": "2021-05-03T07:00",
          "finish_time": "2021-05-03T18:00"
        },
        {
          "start_time": "2021-05-04T07:00",
          "finish_time": "2021-05-04T18:00"
        },
        {
          "start_time": "2021-05-05T07:00",
          "finish_time": "2021-05-05T18:00"
        },
        {
          "start_time": "2021-05-06T07:00",
          "finish_time": "2021-05-06T18:00"
        },
        {
          "start_time": "2021-05-07T07:00",
          "finish_time": "2021-05-07T18:00"
        },
        {
          "start_time": "2021-05-08T07:00",
          "finish_time": "2021-05-08T18:00"
        },
        {
          "start_time": "2021-05-09T07:00",
          "finish_time": "2021-05-09T18:00"
        },
```

```
      {
        "start_time": "2021-05-10T07:00",
        "finish_time": "2021-05-10T18:00"
      },
      {
        "start_time": "2021-05-15T07:00",
        "finish_time": "2021-05-15T18:00"
      },
      {
        "start_time": "2021-05-16T07:00",
        "finish_time": "2021-05-16T18:00"
      },
      {
        "start_time": "2021-05-17T07:00",
        "finish_time": "2021-05-17T18:00"
      },
      {
        "start_time": "2021-05-18T07:00",
        "finish_time": "2021-05-18T18:00"
      },
      {
        "start_time": "2021-05-19T07:00",
        "finish_time": "2021-05-19T18:00"
      },
      {
        "start_time": "2021-05-20T07:00",
        "finish_time": "2021-05-20T18:00"
      }
    ],
    "start_location": {
      "longitude": 144.966167,
      "latitude": -37.82261
    },
    "finish_location": {
      "longitude": 144.966167,
      "latitude": -37.82261
    },
    "is_subcontractor": false,
    "allowed_overtime": true
  },
  {
    "id": "Crew2",
    "start_time": "7:00",
    "finish_time": "18:00",
    "roster": [
      {
        "start_time": "2021-05-01T07:00",
        "finish_time": "2021-05-01T18:00"
      },
```

```
{
  "start_time": "2021-05-02T07:00",
  "finish_time": "2021-05-02T18:00"
},
{
  "start_time": "2021-05-03T07:00",
  "finish_time": "2021-05-03T18:00"
},
{
  "start_time": "2021-05-04T07:00",
  "finish_time": "2021-05-04T18:00"
},
{
  "start_time": "2021-05-05T07:00",
  "finish_time": "2021-05-05T18:00"
},
{
  "start_time": "2021-05-06T07:00",
  "finish_time": "2021-05-06T18:00"
},
{
  "start_time": "2021-05-07T07:00",
  "finish_time": "2021-05-07T18:00"
},
{
  "start_time": "2021-05-08T07:00",
  "finish_time": "2021-05-08T18:00"
},
{
  "start_time": "2021-05-09T07:00",
  "finish_time": "2021-05-09T18:00"
},
{
  "start_time": "2021-05-10T07:00",
  "finish_time": "2021-05-10T18:00"
},
{
  "start_time": "2021-05-15T07:00",
  "finish_time": "2021-05-15T18:00"
},
{
  "start_time": "2021-05-16T07:00",
  "finish_time": "2021-05-16T18:00"
},
{
  "start_time": "2021-05-17T07:00",
  "finish_time": "2021-05-17T18:00"
},
{
```

```
          "start_time": "2021-05-18T07:00",
          "finish_time": "2021-05-18T18:00"
        },
        {
          "start_time": "2021-05-19T07:00",
          "finish_time": "2021-05-19T18:00"
        },
        {
          "start_time": "2021-05-20T07:00",
          "finish_time": "2021-05-20T18:00"
        }
      ],
      "start_location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "finish_location": null,
      "is_subcontractor": true,
      "allowed_overtime": false
    }
  ],
  "jobs": [
    {
      "id": "Job1",
      "location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "allowed_crews": [
        {
          "crew_id": "Crew2",
          "duration": 200,
          "preference": 100,
          "cost": 80
        },
        {
          "crew_id": "Crew1",
          "duration": 180,
          "preference": 0,
          "cost": 40
        }
      ],
      "required_equipment": [
        "EDUCTOR",
        "CRANE"
      ],
      "time_windows": [
        {
          "start": "2021-05-01T10:00",
```

```
          "finish": "2021-05-01T12:00"
        },
        {
          "start": "2021-05-03T11:00",
          "finish": "2021-05-03T13:00"
        }
      ]
    },
    {
      "id": "Job2",
      "location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "allowed_crews": [
        {
          "crew_id": "Crew2",
          "duration": "255",
          "preference": 0,
          "cost": 110
        }
      ],
      "required_equipment": []
    }
  ],
  "assignments": [
    {
      "crew_id": "Crew2",
      "job_id": "Job2",
      "start_time": "2021-05-01T09:00",
      "finish_time": "2021-05-01T12:00"
    }
  ]
}
```

C Dispatch Input Example


```
{
  "id": "78f91273-32a4-4914-91eb-9e6e752c2184",
  "current_time": "2019-04-01T13:30",
  "new_job_id": "Job3",
  "crews": [
    {
      "id": "Crew1",
      "start_time": "2019-04-01T07:00",
      "finish_time": "2019-04-01T18:00",
      "start_location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "finish_location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "current_location": {
        "longitude": 144.966577,
        "latitude": -37.82461
      },
      "is_subcontractor": false,
      "allowed_overtime": true,
      "equipment": [
        "EDUCTOR",
        "CRANE"
      ],
      "current_jobs": [
        {
          "job_id": "Job1",
          "start_time": "2019-04-01T14:00",
          "finish_time": "2019-04-01T17:00"
        }
      ]
    },
    {
      "id": "Crew2",
      "start_time": "2019-04-01T07:00",
      "finish_time": "2019-04-01T18:00",
      "start_location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "finish_location": null,
      "current_location": {
        "longitude": 144.926117,
        "latitude": -37.82351
```

```
      },
      "is_subcontractor": true,
      "allowed_overtime": false,
      "equipment": []
    }
  ],
  "jobs": [
    {
      "id": "Job1",
      "type": "PLANNED",
      "status": "PENDING",
      "priority": 0,
      "location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "allowed_crews": [
        {
          "crew_id": "Crew1",
          "duration": 200,
          "preference": 100,
          "cost": 110
        },
        {
          "crew_id": "Crew2",
          "duration": 180,
          "preference": 0,
          "cost": 30
        }
      ],
      "required_equipment": [
        "EDUCTOR",
        "CRANE"
      ],
      "response_time_windows": null,
      "resolution_time_windows": null
    },
    {
      "id": "Job2",
      "type": "PLANNED",
      "status": "COMMITTED",
      "priority": 0,
      "location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "allowed_crews": [
        {
          "crew_id": "Crew2",
```

```
          "duration": "255",
          "preference": 0,
          "cost": 80
        }
      ],
      "required_equipment": [],
      "response_time_windows": null,
      "resolution_time_windows": null,
      "previously_responded": false
    },
    {
      "id": "Job3",
      "type": "RESPONSIVE",
      "status": "NEW",
      "priority": 1,
      "location": {
        "longitude": 144.966167,
        "latitude": -37.82261
      },
      "allowed_crews": [
        {
          "crew_id": "Crew1",
          "duration": "13",
          "preference": 0
        }
      ],
      "required_equipment": [],
      "response_time_windows": [
        {
          "start": "2019-04-01T00:00",
          "finish": "2019-04-01T15:30"
        }
      ],
      "resolution_time_windows": [
        {
          "start": "2019-04-01T14:00",
          "finish": "2019-04-02T03:00"
        }
      ],
      "previously_responded": false
    }
  ]
}
```

D Unsupported Excel Functions

The following table lists all of the Excel functions that are *not* supported by the optimiser.

ABSREF	DBCS	GET.CELL	LOGEST	RSQ
APP.TITLE	DDB	GET.DOCUMENT	LOGINV	SAVE.TOOLBAR
ARGUMENT	ENABLE.TOOL	GET.WINDOW	LOGNORMDIST	SEARCHB
ASC	END.IF	GET.WORKBOOK	MIDB	SKEW
BETADIST	ERROR	GET.WORKSPACE	N	SLN
BETAINV	EVALUATE	GETPIVOTDATA	NEGBINOMDIST	STEP
BINOMDIST	EXEC	GOTO	NUMBERSTRING	STEYX
CALL	EXPONDIST	GROWTH	PERMUT	SYD
CELL	FDIST	HARMEAN	PHONETIC	TINV
CHIDIST	FINDB	HYPGEOMDIST	PRESS.TOOL	TRIMMEAN
CHIINV	FINV	INFO	PROB	TTEST
CHITEST	FISHER	ISPMT	QUARTILE	TYPE
CONFIDENCE	FISHERINV	KURT	REGISTER.ID	USDOLLAR
CRITBINOM	FTEST	LAST.ERROR	RELREF	VDB
DATEDIF	GAMMADIST	LEFTB	REPLACEB	WEIBULL
DATESTRING	GAMMAINV	LENB	RETURN	WINDOW.TITLE
DB	GAMMALN	LINEST	RIGHTB	ZTEST