|  |  |
| --- | --- |
| Sign in / sign out system  Technical Documentation | Abstract  Technical documentation for the Sign in Sign out project at Gaelectric offices in Belfast.  Lucas McGonagle |

Contents

[System description 4](#_Toc491348434)

[Purpose of the new system 4](#_Toc491348435)

[Inputs, Outputs, Processes 4](#_Toc491348436)

[Inputs 4](#_Toc491348437)

[Outputs 4](#_Toc491348438)

[Processes 4](#_Toc491348439)

[Flow chart 6](#_Toc491348440)

[System Specification 7](#_Toc491348441)

[Downloads 7](#_Toc491348442)

[Applications 7](#_Toc491348443)

[Java libraries 7](#_Toc491348444)

[Services 7](#_Toc491348445)

[Account details 8](#_Toc491348446)

[Clickatell 9](#_Toc491348447)

[Log in details 9](#_Toc491348448)

[Accessing the Tomcat web app manager 9](#_Toc491348449)

[Screen designs 10](#_Toc491348450)

[Data Dictionary 11](#_Toc491348451)

[Database standards 11](#_Toc491348452)

[Entity Diagrams 12](#_Toc491348453)

[Tables 14](#_Toc491348454)

[10MinAverages table 14](#_Toc491348455)

[BudgetMarketPriceTypes table 16](#_Toc491348456)

[BudgetMarketPrices table 17](#_Toc491348457)

[Currencies Table 18](#_Toc491348458)

[CurtailmentProfile table 19](#_Toc491348459)

[Devices table 20](#_Toc491348460)

[DeviceTypes table 21](#_Toc491348461)

[ErrorMessages table 22](#_Toc491348462)

[ExchangeRates table 23](#_Toc491348463)

[Jurisdictions Table 24](#_Toc491348464)

[LUItemTypes table 25](#_Toc491348465)

[LUNodePurposes table 26](#_Toc491348466)

[LUProbabilityFactors table 27](#_Toc491348467)

[LUProbabilityTypes table 28](#_Toc491348468)

[MessageDistribution table 29](#_Toc491348469)

[Nodes table 30](#_Toc491348470)

[NotifiableEvents table 32](#_Toc491348471)

[PowerCurves table 33](#_Toc491348472)

[PriceData table 34](#_Toc491348473)

[PriceTypes table 35](#_Toc491348474)

[ProbabilityValues table 36](#_Toc491348475)

[SignIns table 37](#_Toc491348476)

[SignInDetails 38](#_Toc491348477)

[Sites table 39](#_Toc491348478)

[StatusCodes table 40](#_Toc491348479)

[StatusComments table 41](#_Toc491348480)

[StatusData table 42](#_Toc491348481)

[TemplateNodes table 44](#_Toc491348482)

[Turbines table 46](#_Toc491348483)

[TurbineTypes table 47](#_Toc491348484)

[Users table 48](#_Toc491348485)

[WindSites table 49](#_Toc491348486)

[Database connection details 51](#_Toc491348487)

[Code listings 52](#_Toc491348488)

[Message class 52](#_Toc491348489)

[MessageDao class 57](#_Toc491348490)

[MessageResource class 58](#_Toc491348491)

[MessagesResource class 63](#_Toc491348492)

[PostDataDumperFilter class 67](#_Toc491348493)

[Web.xml 74](#_Toc491348494)

[MessageProcessing class 75](#_Toc491348495)

[Problems encountered so far 82](#_Toc491348496)

[Test plan 83](#_Toc491348497)

# System description

## Purpose of the new system

The purpose of the prosed sign in / sign out system is to aid in health and safety standards when visitors are present on any wind site currently managed by Gaelectric where internet connection is limited. This system should allow Gaelectric to be aware of all personnel currently on each of their windfarms who has signed in via an SMS message. The system should alert a Gaelectric representative if a visitor has not signed out of a wind site before their estimated departure time. The Gaelectric representative should then be able to contact the visitor to ensure that they have not been involved in an incident and extend their departure time if it is required.

## Inputs, Outputs, Processes

### Inputs

**Sign in SMS –** This input is sent to the program via an SMS message. In this SMS message the visitor should include the name of the wind site, their company name, each team members name, what aspect of the wind site they are working on and finally the time they estimate they will be finished (this should be in minutes). Each of these should be separated by a full stop. Each team member name should be separated by a & symbol. This is because the SMS gateway we are using send the text message as one long string but we want it to be separated to allow us to save each sign in to a database.

**Sign out SMS –** This input is sent to the program vis an SMS message. This sign out message is a simple four-digit code which will correspond with the sign in that they have created earlier. The team should ensure that this code is sent back before their estimated time has ran out.

### Outputs

**Ticket No. –** This ticket no is the automatically assigned number given to each new sign in. This is the four-digit code which the site visitor must then use to sign out when they are leaving the site.

**Alert –** This is a message sent out to a selected Gaelectric staff member or group of staff members alerting them that there is a visitor on site who has failed to sign out. This would most likely be an email as the alert should include all the details which the staff member will require to contact the team that is currently on site

**Error SMS –** This is an SMS message that is sent back to the visitor in the event that the SMS message they have sent does not pass the validation checks. This message will change depending on the error that is found.

### Processes

**Add new ticket -** This process involves taking the sign in SMS message, splitting it into each part required for the database table and assigning a new ticket number to the sign in. if it is the first ticket being made then the ticket number will be 1000. If there are previous tickets made, then a new ticket number will be generated by obtaining the latest ticket number and incrementing onto it. The ticket number, wind-site name, company name, aspect they are working on and estimated departure time. Each team member for that ticket will be added to another table along with the ticket number.

**Check tickets –** This process involves the system checking which tickets have not been resolved before their estimated departure time. If a ticket has not been resolved before the estimated departure time, then the alert is sent out to the Gaelectric representatives.

**Sign out –** This process involves resolving an open ticket. When the visitor sends an SMS with the four-digit code to their ticket a Boolean option will be changed to false. That way the check tickets process will see that the visitor has signed out so when their departure time comes up there will be no need to send any alerts.

**Manual sign out –** This process is when after a Gaelectric staff member gets in contact with a visitor who hasn’t signed out before their departure time. The Gaelectric staff member should be able to resolve the ticket in the event that the visitor has simply forgotten to sign out.

**Departure time extension -** This process is when after a Gaelectric staff member gets in contact with a visitor who hasn’t signed out before their departure time. The Gaelectric staff member should be able to add time onto the estimated departure time in the event that the visitors work is taking longer than expected or has encountered delays.

## Flow chart

Team arrives on site

Team will receive a four digit code as a reply

Has the visit exceeded expected departure time?

N

Y

Has the team signed out?

N

Y

Gaelectric representative will attempt to contact the team via the number the sign in text was sent by

Ticket will be closed

This text will contain the name of the windfarm, company name, members of the team, aspect they are working on and the time they estimate they will be finished.

The team signs out by sending the four digit code, they received previously, to the Gaelectric number.

Team member sends text message to Gaelectric to sign in

# System Specification

This following section will detail the software that was used in the development of this system. It will also provide links to their download pages so that development can be done on any pc if the desktop pc which development was started on is not available. It will also detail the locations of important files relevant to the project on the original pc the project was started on.

## Downloads

The REST API can be downloaded at the following GitHub library: https://github.com/GaelectricSoftware/SISO

### Applications

* **Eclipse java neon** (older version of eclipse however newer editions should still work. Libraries may need to be updated, however)
  + https://www.eclipse.org/neon/
* **DBeaver**
  + <https://dbeaver.jkiss.org/download/>
* **PuTTY** (Used to interact with the amazon web service instance of Linux which is where the tomcat server was hosted)
  + <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
* **Restlet client** (Google chrome add-on used to test the Rest API)
  + <https://chrome.google.com/webstore/detail/restlet-client-rest-api-t/aejoelaoggembcahagimdiliamlcdmfm>

### Java libraries

* **Javax**
  + http://www.java2s.com/Code/Jar/j/Downloadjavaxjar.htm
* **Jersey**
  + https://jersey.github.io/download.html
* **Tomcat 8.0** 
  + Installed onto AWS Linux instance

### Services

* Clickatell (log into communicator / central)
  + <https://central.clickatell.com/index>
* Amazon web service EC2 (Used to remotely host the Tomcat 8 Web app server)
  + <https://aws.amazon.com/>

## Account details

* **Gmail** 
  + Email address: [gaelectric.software@gmail.com](mailto:gaelectric.software@gmail.com)
  + Password: GaelectricTechnical
* **GitHub**
  + Username: GaelectricSoftware
  + Password: GaelectricTechnical
* **Amazon web service**
  + Email: [gaelectric.software@gmail.com](mailto:gaelectric.software@gmail.com)
  + Password: GaelectricTechnical

## Clickatell

Clickatell is the service which the system uses as an SMS gateway. Essentially a monthly fee is paid to Clickatell in order to supply us and maintain a two-way number for us. This two-way number allows you to both send and receive SMS messages. This is a vital part of the system so in the section I will go over how to use the Clickatell central and how to access logs of all attempted posts to our API.

### Log in details

|  |  |
| --- | --- |
| **Username:** | GaelectricTechnical |
| **Password:** | dfKSLEQgQAdZKV |

**Remember to frequently ensure that the monthly bill has been paid for Clickatell otherwise the two-way number will be deactivated until the bill is paid. There will be a delay between payment and activation which could lead to delays in development.**

## Accessing the Tomcat web app manager

* On any browser type into the address bar: <http://54.246.224.9:8080/manager/html>
* On opening this webpage you will be prompted to enter login details:

**Username:** admin

**Password:** gaelectric

## Screen designs

3.

4.

5.

2.

1.

**Windfarm:**

**Company:**

**Team members:**

3.

Type: Button

Text: “Sign Out”

Size: 0.95 cm x 2.54 cm

3.

Type: Button

Text: “Search”

Size: 0.95 cm x 2.54 cm

2.

Type: Textbox

Size: 0.95 cm x 5.4 cm

1.

Image: logo.png

Size: 2.86 cm x 7.62 cm

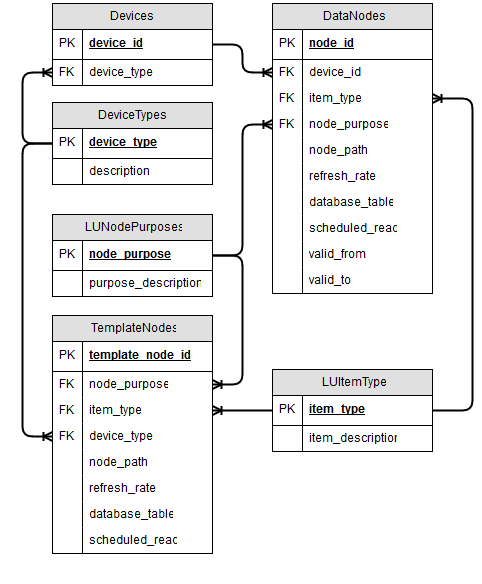
# Data Dictionary

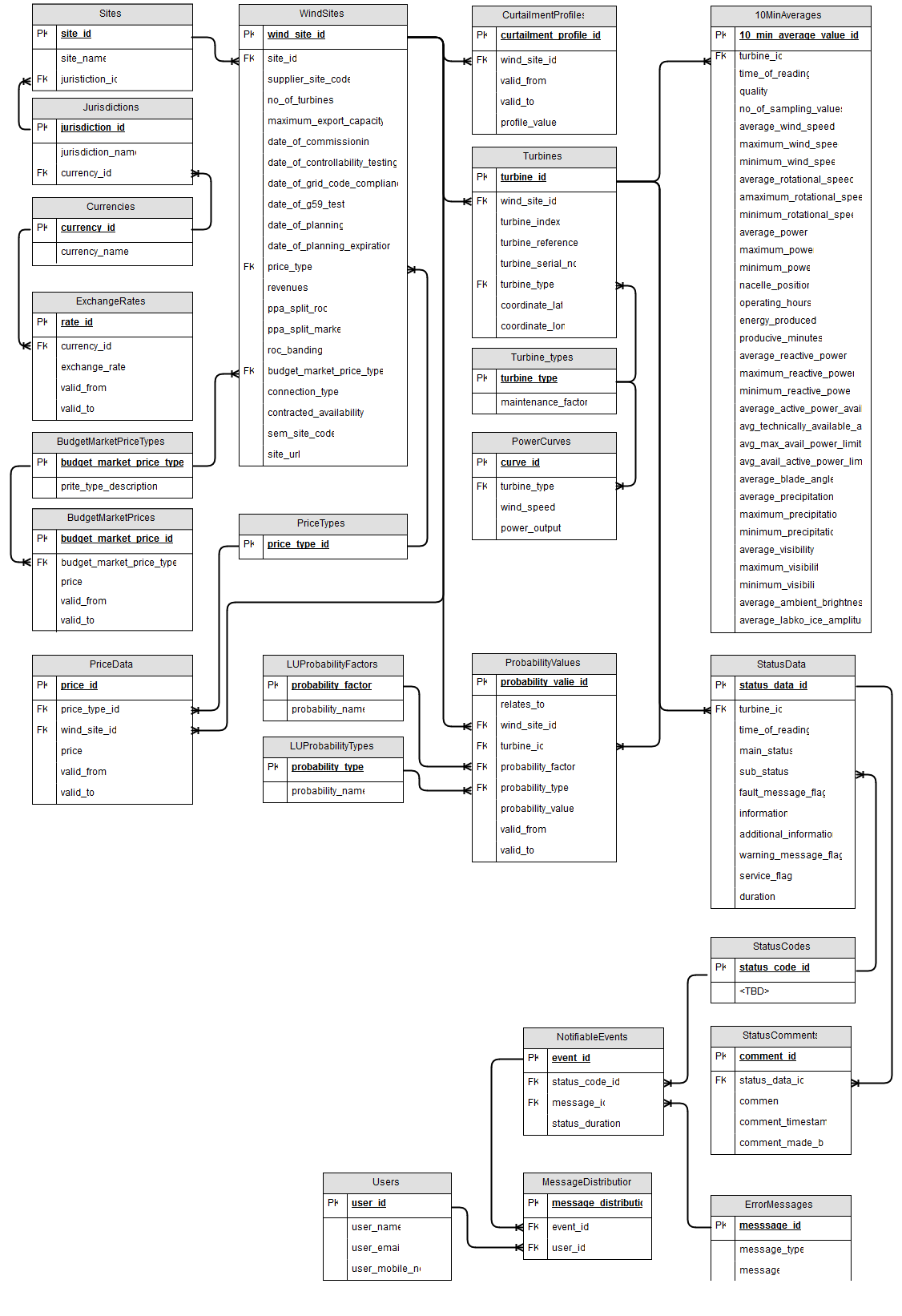
## Database standards

* Database table names to be Pascal Case (start with a capital letter, and a capital letter for each word thereafter and no spaces between words)
* Database table names to be the plural of the type of value they hold e.g. a table for holding the ten minute average data retrieved from the sites will be called “10MinAverages”
* All database field names to be snake\_case (all lowercase, with underscores separating words)
* All database field names to be meaningfully descriptive of their contents
* All database field names to be singular version of the value held

·

## Entity Diagrams





Team members

Ticket No

SignIns

PK

Area

Company

Site

Ticket No

SignIns

PK

## Tables

### 10MinAverages table

This table is populated with values returned from the SCADA server and as such its structure is dictated by the SCADA server’s data. Only the ten\_min\_average\_value\_id (which is an autonumber) and the turbine\_id are not supplied by the SCADA server

#### Fields

| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| --- | --- | --- | --- |
| ten\_min\_average\_value\_id | BIGSERIAL |  | Unique reference for the 10 Min average data. Not user defined |
| turbine\_id | varchar(10) |  | Foreign key – references the Turbines table (turbine\_id field) |
| time\_of\_reading | timestamptz |  | Contains the time of reading (full date/time with timezone) |
| quality | varchar(30) |  | Indication of the quality of the data returned by the SCADA server |
| no\_of\_sampling\_values | integer |  |  |
| average\_wind\_speed | decimal(3, 1) |  |  |
| maximum\_wind\_speed | decimal(3, 1) |  |  |
| minimum\_wind\_speed | decimal(3, 1) |  |  |
| average\_rotational\_speed | decimal(5, 2) |  |  |
| maximum\_rotational\_speed | decimal(5, 2) |  |  |
| minimum\_rotational\_speed | decimal(5, 2) |  |  |
| average\_power | integer |  |  |
| maximum\_power | integer |  |  |
| minimum\_power | integer |  |  |
| nacelle\_position | integer |  |  |
| operating\_hours | integer |  |  |
| energy\_produced | bigint |  |  |
| productive\_minutes | integer |  |  |
| average\_reactive\_power | integer |  |  |
| maximum\_reactive\_power | integer |  |  |
| minimum\_reactive\_power | integer |  |  |
| average\_active\_power\_available\_in\_wind | integer |  |  |
| avg\_technically\_available\_active\_power | integer |  |  |
| avg\_max\_avail\_power\_limited\_by\_force\_majeuere | integer |  |  |
| avg\_avail\_active\_power\_limited\_by\_ext\_set\_point | integer |  |  |
| average\_blade\_angle | decimal(4, 1) |  |  |
| average\_precipitation | decimal(6, 3) |  |  |
| maximum\_precipitition | decimal(6, 3) |  |  |
| minimum\_precipitition | decimal(6, 3) |  |  |
| average\_visibility | decimal(6, 1) |  |  |
| maximum\_visibility | decimal(6, 1) |  |  |
| minimum\_visibility | decimal(6, 1) |  |  |
| average\_ambient\_brightness | integer |  |  |
| average\_labko\_ice\_amplitude | integer |  |  |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_10\_min\_average\_value\_id | ten\_min\_average\_value\_id | Table primary key constraint |
| fk\_10\_min\_average\_turbine\_id | turbine\_id | Foreign key constraint ensuring integrity with Turbines table |
| unique\_10\_min\_average | turbine\_id, time\_of\_reading | Prevents recording of duplicate information for a specific turbine. |

#### Indices

|  |  |
| --- | --- |
| **Index name** | **Applies to fields** |
| 10MinAverageIndex | turbine\_id, time\_of\_reading |

### BudgetMarketPriceTypes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| budget\_market\_price\_type | varchar(5) |  | Code for price type e.g. POWYL, POWYM |
| description | varchar(25) |  | description of the budget market price type |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_budget\_market\_price\_types | budget\_market\_price\_type | Table primary key constraint |

### BudgetMarketPrices table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| budget\_market\_price\_id | serial |  | Unique reference for the budget market price. Not user defined |
| budget\_market\_price\_type | Varchar(5) |  | Foreign key - The budget\_market\_price\_type this entry relates to – see BudgetMarketPriceTypes table |
| price | decimal(6, 2) |  | The price applicable for this Price type |
| valid\_from | Date |  | The date from which this price is valid for this price type |
| valid\_to | Date |  | The date this price is valid to for this price type |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_budget\_market\_price | budget\_market\_price | Table primary key constraint |
| fk\_budget\_market\_price\_type | budget\_market\_price\_type | Foreign key constraint ensuring integrity with BudgetMarketPriceTypes table |
| chk\_budget\_market\_price\_valid\_to | valid\_to | ensures that the valid\_to value is after valid\_from |
| exclude\_overlapping\_budget\_market\_prices | budget\_market\_price\_type, valid\_from, valid\_to | Ensures that it is not possible to have two budget market prices for the same price type that have overlapping date ranges.  **NB** this is PostgreSQL specific functionality and will not migrate to other DBMS platforms |

### Currencies Table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| currency\_id | Varchar(3) |  | ISO 4217 currency code e.g. GPB, EUR, USD |
| currency\_name | Varchar(20) |  | Full name of the currency |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_currency\_id | currency\_id | Table primary key constraint |

### CurtailmentProfile table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| curtailment\_profile\_id | serial |  | Unique reference for the curtailment profile. Not user defined |
| wind\_site\_id | Varchar(5) |  | Foreign key - The site id this profile relates to – from Site table |
| valid\_from | Date |  | The date this curtailment profile is valid from |
| valid\_to | Date |  | The date this curtailment profile is valid to |
| profile\_value | Numeric() |  | The value of the curtailment profile |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_curtailment\_profile\_id | curtailment\_profile\_id | Table primary key constraint |
| fk\_curtailment\_profile\_wind\_site\_id | wind\_site\_id | Foreign key constraint ensuring integrity with Currencies table |
| chk\_curtailment\_valid\_to | valid\_to | ensures that the valid\_to value is after valid\_from |
| exclude\_overlapping\_curtailment\_profile | wind\_site\_id, valid\_from, valid\_to | Ensures that it is not possible to have two curtailment profiles for the same wind site that have overlapping date ranges.  **NB** this is PostgreSQL specific functionality and will not migrate to other DBMS platforms |

### Devices table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| device\_id | varchar(30) |  | this will be the same as the underlying turbine/wind site id to ensure returned data will meet the foreign key constraints for the tables in which it is recorded |
| device\_type | varchar(30) |  | Foreign Key – references the DeviceTypes table (device\_type column) |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_device\_id | device\_id | Table primary key constraint |
| fk\_device\_type | device\_type | Foreign key constraint ensuring integrity with DeviceTypes table |

### DeviceTypes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| device\_type | varchar(30) |  | Device type – see notes below |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_device\_type | device\_type | Table primary key constraint |

#### Notes

The DeviceTypes table is ordinarily populated by database triggers by insertions into appropriate table e.g. the TurbineTypes table. The device\_type will generally be the turbine type – however in the case of a Site an entry must be created for a site type e.g. “xSite” where x is the site use (“WindSite”, “SolarSite” etc.)

### ErrorMessages table

Final structure to be confirmed – table will hold “templated” text for error messages to be sent in response to certain status events

### ExchangeRates table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| rate\_id | serial |  | Auto-incrementing number to act as unique row identifier |
| currency\_id | Varchar(3) |  | Code identifying the currency e.g. GBP, EUR |
| exchange\_rate | numeric(10, 9) |  | Exchange rate from this currency to “base” currency for reporting. The base currency will exist in this table and will always have an exchange rate of 1 |
| valid\_from | Date |  | Date the exchange rate is valid from |
| valid\_to | Date |  | Date the exchange rate is valid to |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_rate\_id | rate\_id | Table primary key constraint |
| fk\_exchange\_rate\_currencies | currency\_id | Foreign key constraint ensuring integrity with Currencies table |
| chk\_exchange\_rate\_valid\_to | valid\_to | ensures that the valid\_to value is after valid\_from |
| exclude\_overlapping\_date\_ranges | currency\_id, valid\_from, valid\_to | Ensures that it is not possible to have two exchange rates for the same currency that have overlapping date ranges.  **NB** this is PostgreSQL specific functionality and will not migrate to other DBMS platforms |

### Jurisdictions Table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| jurisdiction\_id | Varchar(10) |  | Short reference e.g. UK, ROI |
| jurisdiction\_name | Varchar(40) |  | Full description e.g. United Kingdom |
| currency\_id | Varchar(3) |  | Code that references the currency used in this jurisdiction – see currency table. |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_jurisdiction\_id | jurisdiction\_id | Table primary key constraint |
| fk\_currency\_id | currency\_id | Foreign key constraint ensuring integrity with Currencies table |

### LUItemTypes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| item\_type | varchar(20) |  | The class name for the class that must be created in the logger application to handle the requested data being returned by the node |
| item\_description | varchar(20) |  | Expanded name for the item type – will be the displayed values in web interface form fields |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_item\_type | item\_type | Table primary key constraint |

### LUNodePurposes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| node\_purpose\_id | varchar(5) |  | Abbreviation representing the data type to be collected e.g. “TMA”, “SD” |
| node\_purpose | varchar(35) |  | Full name of the node purpose e.g. “Ten Minute Averages”, “Status Data”. This will be the displayed value in the web interface forms |
| purpose\_description | varchar(200) |  | Expanded additional description of the data being collected – can be used to provide tooltip for value selection in web interface |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_node\_purpose\_id | node\_purpose\_id | Table primary key constraint |

### LUProbabilityFactors table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| probability\_factor | varchar(3) |  | Code representing the probability factor e.g. “P50”, “P75” etc. |
| probability\_name | varchar(20) |  | Name for probability factor (will be the display name for selections in web portal) |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_probability\_factor | probability\_factor | Table primary key constraint |

### LUProbabilityTypes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| probability\_type | varchar(3) |  | Code representing the probability name |
| probability\_name | varchar(20) |  | Name for probability type e.g. “Production”, “Wind speed” |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_probability\_type | probability\_type | Table primary key constraint |

### MessageDistribution table

Final structure to be confirmed – this table will identify which NotifiableEvents need to be sent to which staff

### Nodes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| node\_id | UUID |  | Unique reference for this node (generated by front end systems) |
| device\_id | varchar(10) |  | Foreign key – references the Devices table |
| node\_purpose\_id | varchar(5) |  | Foreign key – references the LUNodePurposes table. Identifies what the node collects e.g. 10 minute averages, Status data etc. |
| item\_type | varchar(20) |  | Foreign key – references the LUItemTypes table. Identifies the class of object the logger application should create to manage data retrieval for this node. |
| node\_path | varchar(100) |  | The server path (from the SCADA server) from where the values can be retrieved |
| refresh\_rate | integer |  | Time in milliseconds between reads of the node from the SCADA server |
| database\_table | varchar(50) |  | The database table that the returned values should be stored to |
| scheduled\_read | Boolean |  | Indicates whether this node should be configured by the logger service to read the values at a specified interval (scheduled\_read = true) or to “subscribe” to the node on the SCADA server and “listen” for changes (scheduled\_read = false) |
| recovery\_index | Integer |  | Represents the first number that is appended to the end of the node\_path to recover historic information |
| valid\_from | Date |  | The date from which data should be collected for this node |
| valid\_to | Date |  | The data from which data should cease to be collected for this node. Allows short-term ‘set-and-forget’ additional data collection. |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_node\_id | node\_id | Table primary key constraint |
| fk\_node\_device\_id | device\_id | Foreign key constraint ensuring integrity with Devices table |
| fk\_node\_node\_purpose | node\_purpose\_id | Foreign key constraint ensuring integrity with LUNodePurposes table |
| fk\_node\_item\_type | item\_type | Foreign key constraint ensuring integrity with LUItemTypes table |
| chk\_nodes\_valid\_to | valid\_to | Checks that valid\_to date is after the valid\_from date |
| exclude\_overlapping\_node | device\_id, node\_purpose\_id, valid\_from, valid\_to | Ensures that it is not possible to have two probability values for the same device/node purpose/valid from/valid to that have overlapping date ranges.  **NB** this is PostgreSQL specific functionality and will not migrate to other DBMS platforms |

### NotifiableEvents table

Final structure to be confirmed – this table will contain data that identifies what status codes trigger the sending of messages (and for what duration these statuses need to be active before a message is sent)

### PowerCurves table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| power\_curve\_id | serial |  | Unique reference for the power curve data. Not user defined |
| turbine\_type | varchar(10) |  | Foreign key – references the TurbineTypes table |
| wind\_speed | integer |  |  |
| power\_output | integer |  | the power output (in kW) the turbine type is rated for at the specified wind speed |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_power\_curve\_id | power\_curve\_id | Table primary key constraint |
| fk\_power\_curve\_turbine\_type | turbine\_type | Foreign key constraint ensuring integrity with TurbineTypes table |
| unique\_power\_curve | turbine\_type, wind\_speed | Ensures it is not possible to have duplicate/contradictory values for a power curve |

### PriceData table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| price\_id | serial |  | Unique reference for the annual price data. Not user defined |
| price\_type\_id | varchar(5) |  | Foreign key – references the PriceTypes table |
| wind\_site\_id | varchar(5) |  | Foreign key – references the WindSites table. Some prices are site specific while some apply to the entire jurisdiction. This field being nullable allows for the recording of a site if the price is site-specific |
| price | Numeric(8, 4) |  | the price payable |
| valid\_from | Date |  | The date from which this price is valid |
| valid\_to | Date |  | The date this price ceases to be valid |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_price\_id | price\_id | Table primary key constraint |
| fk\_wind\_site\_id | wind\_site\_id | Foreign key constraint ensuring integrity with WindSites table |
| fk\_price\_data | price\_type\_id | Foreign key constraint ensuring integrity with PriceData table |
| chk\_price\_valid\_to | valid\_to | ensures that the valid\_to value is after valid\_from |
| unique\_pricing | price\_type\_id, wind\_site\_id, valid\_from, valid\_to | prevents creation of any pricing data that duplicates existing records |
| exclude\_overlapping\_price\_data | price\_type\_id, wind\_site\_id, valid\_from, valid\_to | Ensures that it is not possible to have two annual prices for the same price type and site that have overlapping date ranges.  **NB** this is PostgreSQL specific functionality and will not migrate to other DBMS platforms |

### PriceTypes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| price\_type\_id | varchar(5) |  | Code to identify the price type e.g. ROC, Refit |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_price\_type\_id | price\_type | Table primary key constraint |

### ProbabilityValues table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| probability\_id | serial |  | Unique reference for the probability value data. Not user defined |
| relates\_to | varchar(10) |  | Valid values – “Site”, “Turbine” |
| wind\_site\_id | varchar(5) |  | Foreign key – references the WindSites table |
| turbine\_id | varchar(10) |  | Foreign key – references the Turbines table |
| probability\_factor | varchar(3) |  | Foreign key – references the LUProbabilityFactors table |
| probability\_type | varchar(20) |  | Foreign key – references the LUProbabilityTypes table |
| valid\_from | date |  | Date from which this probability value is valid |
| valid\_to | date |  | Date at which this probability value is no longer valid |
| probability\_value | decimal(5,4) |  | The probability value for this entry – the wind speed/power |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_probability\_id | probability\_id | Table primary key constraint |
| fk\_probability\_value\_turbine\_id | turbine\_id | Foreign key constraint ensuring integrity with Turbines table |
| fk\_probability\_value\_wind\_site\_id | wind\_site\_id | Foreign key constraint ensuring integrity with WindSites table |
| chk\_relates\_to | relates\_to | Checks the value is in the list [‘Site’, ‘Turbine’] |
| fk\_probability\_value\_probability\_factor | probability\_factor | Foreign key constraint ensuring integrity with LUProbabilityFactors table |
| fk\_probability\_value\_probability\_type | probability\_type | Foreign key constraint ensuring integrity with LUProbabilityTypes table |
| chk\_probability\_values\_valid\_to | valid\_to | Checks that valid\_to date is after the valid\_from date |
| exclude\_overlapping\_probability\_values | wind\_site\_id, turbine\_id, valid\_from, valid\_to | Ensures that it is not possible to have two probability values for the same site/turbine/type/factor that have overlapping date ranges.  **NB** this is PostgreSQL specific functionality and will not migrate to other DBMS platforms |

### SignIns table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| ticket\_id | INT |  | Unique identifier for each sign in |
| wind\_site\_id | VARCHAR(5) |  | Foreign key - Unique 3 letter code for each Gaelectric wind-site E.g CHW. Reference wind-site table |
| area\_of\_work | VARCHAR(5) |  | Two letter code for an area of the wind-site E.g T1 for turbine 1 |
| contact\_no | VARCHAR(11) |  | A mobile number which the sign in was sent from. Used to contact the visitors |
| arrival\_time | TIMESTAMP |  | The time which the SMS message was sent. Used as the time which the visitor arrived on site. |
| departure\_time | TIMESTAMP |  | The time which the visitors expect to leave the site at. |
| resolved | BOOLEAN |  | Denotes whether or not the visitor has been signed out. |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pkTicket\_id | ticket\_id | Table primary key constraint |
| fkWind\_site\_id | wind\_site\_id | Foreign key constraint ensuring integrity with WindSites table |

### SignInDetails

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| ticket\_id | INT |  | Foreign key – Unique identifier for each sign in |
| member\_name | VARCHAR |  | The names of each member present for each sign in |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| fkTicket\_id | ticket\_id | Foreign key constraint ensuring integrity with WindSites table |
| pkSignInDetails | ticket\_id member\_name | Table primary key constraint. Composite Key |

### Sites table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| site\_id | Varchar(5) |  | Gaelectric 2 letter reference e.g CH, SK etc |
| site\_name | Varchar(20) |  | “Friendly” name for the site e.g. Carn Hill, Skrine |
| jurisdiction\_id | Varchar(10) |  | Foreign key - The jurisdiction this site is in. See Jurisdiction table |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_site\_id | site\_id | Table primary key constraint |
| fk\_jurisdiction\_id | jurisdiction\_id | Foreign key constraint ensuring integrity with Jurisdiction table |

### StatusCodes table

Final structure to be confirmed – this may derive into multiple tables dependant on the final normalisation of the data. Will hold status codes and their corresponding descriptions

### StatusComments table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| status\_comment\_id | serial |  |  |
| status\_id | integer |  | Foreign key – references StatusData table (status\_id field) |
| comment\_text | text |  | Unlimited free text |
| comment\_timestamp | timestamptz |  | full timestamp recording when comment was recorded (full date/time with timezone) |
| comment\_made\_by | varchar(50) |  | Name of user recording comment – see notes |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_status\_comment\_id | status\_comment\_id | Table primary key constraint |
| fk\_status\_data\_comments\_status\_id | status\_id | Foreign key constraint ensuring integrity with StatusData table |

#### Notes

The data recorded in ‘comment\_made\_by’ will likely be derived from a user table. As such the data type of this field is subject to change until final user data table is confirmed.

### StatusData table

This table is populated with values returned from the SCADA server and as such its structure is dictated by the SCADA server’s data. Only the status\_id (which is an autonumber), turbine\_id, and duration fields are not supplied by the SCADA server

#### Fields

| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| --- | --- | --- | --- |
| status\_id | BIGSERIAL |  | Unique reference for the Status data. Not user defined |
| turbine\_id | varchar(10) |  | Foreign key – references the Turbines table (turbine\_id field) |
| time\_of\_reading | timestamptz |  | Contains the time of reading (full date/time with timezone) |
| quality | varchar(30) |  | Indication of the quality of the data returned by the SCADA server |
| main\_status | integer |  |  |
| sub\_status | integer |  |  |
| fault\_message\_flag | Boolean |  |  |
| information | integer |  |  |
| additional\_information | integer |  |  |
| warning\_message\_flag | Boolean |  |  |
| service\_flag | Boolean |  |  |
| duration | interval |  | System calculated time duration between status becoming “live” and new status being received |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_status\_id | status\_id | Table primary key constraint |
| fk\_status\_data\_turbine\_id | turbine\_id | Foreign key constraint ensuring integrity with Turbines table |
| unique\_status\_data | turbine\_id, time\_of\_reading, main\_status, sub\_status, information, additional\_information | Prevents recording of duplicate information for a specific turbine. |

#### Indices

|  |  |
| --- | --- |
| **Index name** | **Applies to fields** |
| StatusDataIndex | turbine\_id, time\_of\_reading |

#### Notes

The main\_status, sub\_status, information and additional\_information fields represent codes that need to be mapped against their corresponding values. As such, once the Status table structure is finalised, these may become foreign key values.

### TemplateNodes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| template\_node\_id | serial |  | Unique reference for the template node data. Not user defined |
| device\_type | varchar(30) |  | Foreign Key – references the DeviceTypes table (device\_type column) |
| node\_purpose\_id | varchar(5) |  | Foreign Key – references the LUNodePurposes table (node\_purpose\_id column) |
| item\_type | varchar(20) |  | Foreign Key – references the LUItemTypes table (item\_type column) |
| node\_path | varchar(100) |  | Holds the node path to the data item – see notes below |
| refresh\_rate | integer |  | The rate in milliseconds that the SCADA server is polled |
| database\_table | varchar(50) |  | The database table where the values returned by this node are ordinarily stored |
| scheduled\_read | Boolean |  | Indicates whether this node should be configured by the logger service to read the values at a specified interval (scheduled\_read = true) or to “subscribe” to the node on the SCADA server and “listen” for changes (scheduled\_read = false) |
| recovery\_index | integer |  | Represents the first number that is appended to the end of the node\_path to recover historic information |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_template\_node | standard\_node | Table primary key constraint |
| fk\_template\_nodes\_node\_purpose | node\_purpose | Foreign key constraint ensuring integrity with LUNodePurposes table |
| fk\_template\_node\_device\_type | device\_type | Foreign key constraint ensuring integrity with DeviceTypes table |
| fk\_template\_nodes\_item\_type | item\_type | Foreign key constraint ensuring integrity with LUItemTypes table |
| unique\_template\_node | device\_type, node\_purpose\_id | Ensures it is not possible to create more than one template node of a particular purpose for a particular device. |

#### Notes

Template nodes are a means of templating the date nodes that will be collected for a particular device type. As such, they are normally only populated with information that can be applied generically. The exception is the node\_path, that will contain specific information relating to where the information is stored on the server. For this reason, node\_path should be populated with the node\_path suitably masked so that the standard String.format procedure can insert the values into the appropriate locations when building an actual node from the template

### Turbines table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| turbine\_id | varchar(10) |  | Turbine reference combining the wind\_site\_id and the turbine\_reference e.g. CLWT1 |
| wind\_site\_id | varchar(5) |  | Foreign key – references the WindSites table |
| turbine\_index | integer |  | numerical reference for the turbine – corresponds to the “PlantNo” on the SCADA server. May differ numerically from the number in the turbine\_reference |
| turbine\_reference | varchar(3) |  | Name reference for the turbine – normally Tx (where x is the turbine number) |
| turbine\_serial\_no | varchar(20) |  | Manufacturer’s turbine serial number for the turbine |
| turbine\_type | varchar(10) |  | Foreign key – references the TurbineTypes table |
| coordinate\_lat | decimal |  | latitude coordinate for the turbine |
| coordinate\_lon | decimal |  | longitude coordinate for the turbine |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_turbine\_id | turbine\_id | Table primary key constraint |
| fk\_turbine\_type | turbine\_type | Foreign key constraint ensuring integrity with TurbineTypes table |
| fk\_turbine\_wind\_site\_id | wind\_site\_id | Foreign key constraint ensuring integrity with WindSites table |

#### Triggers

|  |  |  |  |
| --- | --- | --- | --- |
| **Trigger name** | **Calls function** | **Acts on** | **Purpose** |
| create\_turbine\_device | turbine\_create\_device() | INSERT, UPDATE, DELETE | Creates a corresponding entry in the Devices table when a Turbine is created/amended/deleted |

### TurbineTypes table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| turbine\_type | varchar(10) |  | Turbine type e.g. “E70 2.0” |
| maintenance\_factor | decimal(5, 4) |  | % value of time required for maintenance |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_turbine\_type | turbine\_type | Table primary key constraint |

#### Triggers

|  |  |  |  |
| --- | --- | --- | --- |
| **Trigger name** | **Calls function** | **Acts on** | **Purpose** |
| turbine\_type\_devices | turbine\_types\_create\_device() | INSERT, UPDATE, DELETE | Creates a corresponding entry in the DeviceTypes table when a TurbineType is created/amended/deleted |

### Users table

Final structure to be confirmed – this table may be defined in the default tables created by Django for the web interface.

### WindSites table

#### Fields

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data Type** | **Nullable** | **Details/range of possible values** |
| wind\_site\_id | Varchar(5) |  | Gaelectric 3 letter reference e.g CHW, SKW etc |
| site\_id | Varchar(5) |  | Foreign key – the site this wind\_site is on. Reference site table |
| supplier\_site\_code | Varchar(20) |  | Code/reference used by supplying company to identify site |
| no\_of\_turbines | Integer |  | Total number of turbines on site |
| maximum\_export\_capacity | Numeric() |  | Total amount of power the site may export to the grid |
| date\_of\_commissioning | Date |  |  |
| date\_of\_controllability\_testing | Date |  |  |
| date\_of\_grid\_code\_compliance | Date |  |  |
| date\_of\_g59\_test | Date |  |  |
| date\_of\_planning | Date |  |  |
| date\_of\_planning\_expiration | Date |  | Duration of planning is needed for reporting – this supports calculation |
| price\_type | varchar(5) |  | Foreign key – references Annual Price Type table |
| revenues | varchar(10) |  | Valid values – “Fixed”, “Split” |
| ppa\_split\_roc | numeric(5, 4) |  | Optional value (representing percentage) for the split of ROC payments between Gaelectric/Funders |
| ppa\_split\_market | numeric(5, 4) |  | Optional value (representing percentage) for the split of market revenue payments between Gaelectric/Funders |
| budget\_market\_price\_type | varchar(5) |  | Foreign key – references BudgetMarketPriceType table |
| connection\_type |  |  | Valid values – “Firm”, “Not firm” |
| contracted\_availability | numeric(5, 4) |  | percentage representation of contracted availability |
| sem\_site\_code | varchar(20) |  | optional value to allow recording of appropriate SEM site code to assist data integration |
| site\_url | varchar(100) |  | URL of site’s SCADA server for data connection |

#### Constraints

|  |  |  |
| --- | --- | --- |
| **Constraint name** | **Applies to field** | **Description** |
| pk\_wind\_site\_id | wind\_site\_id | Table primary key constraint |
| fk\_site\_id | site\_id | Foreign key constraint ensuring integrity with Sites table |
| fk\_budget\_market\_price\_type | budget\_market\_price\_type | Foreign key constraint ensuring integrity with BudgetMarketPriceTypes table |
| fk\_site\_price\_type | price\_types | Foreign key constraint ensuring integrity with PriceTypes table |

#### Triggers

|  |  |  |  |
| --- | --- | --- | --- |
| **Trigger name** | **Calls function** | **Acts on** | **Purpose** |
| wind\_site\_devices | wind\_site\_create\_device() | INSERT, UPDATE, DELETE | Creates a corresponding entry in the Devices table when a WindSite is created/amended/deleted |

## Database connection details

The database language used to create this data base was PostgreSQL and the application which was used to create and manage this database was DBeaver.

Below is the connection details required to connect to the database sever which holds the Gaelectric database

Host: [dbtestinstance.cclcp0zv6ibp.eu-west-1.rds.amazonaws.com](http://dbtestinstance.cclcp0zv6ibp.eu-west-1.rds.amazonaws.com/)

Port: 5432

Username: Stephen

Password: superuser

(**Subject to change!)**

# Code listings

## Message class

**This class creates the l object which will hold message details. What is sent by Clickatell to our system will become an instance of this object.**

package com.vogella.jersey.todo.model;

import javax.xml.bind.annotation.XmlElement;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement (name = "clickmo")

public class Message

{

private String api\_id;

private String moMsgId;

private String from;

private String to;

private String timestamp;

private String text;

private String charset;

private String udh;

public Message()

{

this.api\_id = " ";

this.moMsgId = " ";

this.from = " ";

this.to = " ";

this.timestamp = " ";

this.text = " ";

this.charset = " ";

this.udh = " ";

}

@XmlElement (name = "api\_id")

public String getApiId()

{

return api\_id;

}

public void setApiId(String api\_id)

{

this.api\_id = api\_id;

}

@XmlElement (name = "moMsgId")

public String getMoMessageid()

{

return moMsgId;

}

public void setMoMessageid(String moMessageid)

{

this.moMsgId = moMessageid;

}

@XmlElement (name = "from")

public String getFrom()

{

return from;

}

public void setFrom(String from)

{

this.from = from;

}

@XmlElement (name = "to")

public String getTo()

{

return to;

}

public void setTo(String to)

{

this.to = to;

}

@XmlElement (name = "timestamp")

public String getTimestamp()

{

return timestamp;

}

public void setTimestamp(String timestamp)

{

this.timestamp = timestamp;

}

@XmlElement (name = "text")

public String getText()

{

return text;

}

public void setText(String text)

{

this.text = text;

}

@XmlElement (name = "charset")

public String getCharset()

{

return charset;

}

public void setCharset(String charset)

{

this.charset = charset;

}

@XmlElement (name = "udh")

public String getUdh()

{

return udh;

}

public void setUdh(String udh)

{

this.udh = udh;

}

public void toPrint()

{

System.out.print(this.api\_id + " " + this.moMsgId + " " + this.to + " " + this.from + " " + this.timestamp + " " + this.charset + " " + this.udh + " " + this.text);

}

}

## MessageDao class

package com.vogella.jersey.todo.dao;

import java.util.HashMap;

import java.util.Map;

import com.vogella.jersey.todo.model.Message;

public enum MessageDao

{

instance;

private Map<String, Message> contentProvider =new HashMap<>();

private MessageDao()

{

Message Message = new Message();

Message.setApiId("12345");

Message.setMoMessageid("1");

contentProvider.put("1", Message);

Message Message2 = new Message();

Message2.setApiId("12345");

Message2.setMoMessageid("2");

contentProvider.put("2", Message2);

}

public Map<String, Message> getModel()

{

return contentProvider;

}

}

## MessageResource class

package com.vogella.jersey.todo.resources;

import javax.ws.rs.Consumes;

import javax.ws.rs.DELETE;

import javax.ws.rs.GET;

import javax.ws.rs.PUT;

import javax.ws.rs.Produces;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Request;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.UriInfo;

import javax.xml.bind.JAXBElement;

import com.vogella.jersey.todo.JSON.\*;

import com.vogella.jersey.todo.model.Message;

import com.vogella.jersey.todo.dao.MessageDao;

public class MessageResource

{

@Context

UriInfo uriInfo;

@Context

Request request;

String id;

public MessageResource(UriInfo uriInfo, Request request, String id)

{

this.uriInfo = uriInfo;

this.request = request;

this.id = id;

}

**/\*\***

**\* Used to get a single Message as Application XML**

**\* Searches the messages (MessageDao) Hashmap to obtain the message with the requested id.**

**\* if it cannot find the hashmap member then it throws a Runtime Exception and a message saying that the requested message does not exist.**

**\***

**\***

**\* @return message (Message)**

**\*/**

@GET

@Produces(MediaType.APPLICATION\_XML)

public Message getMessage()

{

Message Message = MessageDao.instance.getModel().get(id);

if(Message==null)

throw new RuntimeException("Get: Message with " + id + " not found");

return Message;

}

**/\*\***

**\* Used to get a single Message as Application JSON**

**\* Searches the messages (MessageDao) Hashmap to obtain the message with the requested id.**

**\* if it cannot find the message then it throws a Runtime Exception and a message saying that the requested message does not exist.**

**\***

**\***

**\* @return message (Message)**

**\*/**

@GET

@Produces(MediaType.APPLICATION\_JSON)

public Message getMessageo()

{

Message Message = MessageDao.instance.getModel().get(id);

if(Message==null)

throw new RuntimeException("Get: Message with " + id + " not found");

return Message;

}

**/\*\***

**\* Used to get a single Message as Text XML**

**\* Searches the messages (MessageDao) Hashmap to obtain the message with the requested id and display it in a format that is readable by the user**

**\* if it cannot find the message then it throws a Runtime Exception and a message saying that the requested message does not exist.**

**\***

**\***

**\* @return message (Message)**

**\*/**

@GET

@Produces(MediaType.TEXT\_XML)

public Message getMessageHTML()

{

Message Message = MessageDao.instance.getModel().get(id);

if(Message==null)

throw new RuntimeException("Get: Message with " + id + " not found");

return Message;

}

**/\*\***

**\* Used to update an already existing message.**

**\* Consumes an XML object that replaces an existing message**

**\* Uses the putAndGetResponse method to check if the resource has been updated or not.**

**\* @param Message**

**\* @return Response**

**\*/**

@PUT

@Consumes(MediaType.APPLICATION\_XML)

public Response putMessage(JAXBElement<Message> Message)

{

Message c = Message.getValue();

return putAndGetResponse(c);

}

**/\*\***

**\* Deletes the message with the stated id**

**\* if it does not exist, it throws a runTime exception.**

**\*/**

@DELETE

public void deleteMessage()

{

Message c = MessageDao.instance.getModel().remove(id);

if(c==null)

throw new RuntimeException("Delete: Message with " + id + " not found");

}

**/\*\***

**\* This method is used to check that a resource has been created and then returns a response.**

**\* Used to send a response when the put method is used**

**\* @param Message (Message)**

**\* @return Response**

**\*/**

private Response putAndGetResponse(Message Message)

{

Response res;

if(MessageDao.instance.getModel().containsKey(Message.getMoMessageid()))

{

res = Response.noContent().build();

}

else

{

res = Response.created(uriInfo.getAbsolutePath()).build();

}

MessageDao.instance.getModel().put(Message.getMoMessageid(), Message);

return res;

}

}

## MessagesResource class

**This class contains the methods for getting and posting new messages. There are get methods for both XML and JSON but the post method is for XML. It is in this class that I believe the issues with the system are coming from. I believe that the post method that is within this class has not been set to receive XML in the same format as what Clickatell is posting to the server.**

package com.vogella.jersey.todo.resources;

import java.io.BufferedWriter;

import java.io.File;

import java.io.FileOutputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

import java.io.Writer;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

import javax.servlet.http.HttpServletResponse;

import javax.ws.rs.Consumes;

import javax.ws.rs.FormParam;

import javax.ws.rs.GET;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.PathParam;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.Context;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Request;

import javax.ws.rs.core.Response;

import javax.ws.rs.core.Response.Status;

import javax.ws.rs.core.UriInfo;

import javax.xml.crypto.dsig.XMLObject;

import com.vogella.jersey.todo.dao.MessageDao;

import com.vogella.jersey.todo.JSON.XML;

import com.vogella.jersey.todo.model.Message;

// Will map the resource to the URL Messages

@Path("/Messages")

public class MessagesResource {

@Context

UriInfo uriInfo;

@Context

Request request;

**/\*\***

**\* Creates a list of all Messages (Messages) as Text XML that can be displayed in a browser**

**\***

**\* @return Messages (messages)**

**\*/**

**@GET**

@Produces(MediaType.TEXT\_XML)

public List<Message> getMessagesBrowser() {

List<Message> Messages = new ArrayList<Message>();

Messages.addAll(MessageDao.instance.getModel().values());

return Messages;

}

**/\*\***

**\* Creates a list of all Messages (Messages) as JSON**

**\***

**\* @return**

**\*/**

@GET

@Produces(MediaType.APPLICATION\_JSON)

public List<Message> getMessagess() {

List<Message> Messages = new ArrayList<Message>();

Messages.addAll(MessageDao.instance.getModel().values());

return Messages;

}

**/\*\***

**\* Creates a list of all Messages (Messages) as Application XML**

**\***

**\* @return**

**\*/**

@GET

@Produces(MediaType.APPLICATION\_XML)

public List<Message> getMessages() {

List<Message> Messages = new ArrayList<Message>();

Messages.addAll(MessageDao.instance.getModel().values());

return Messages;

}

**/\*\***

**\* Returns a count of all Messages (Messages) that are in the Hashmap**

**\***

**\* @return**

**\*/**

@GET

@Path("count")

@Produces(MediaType.TEXT\_PLAIN)

public String getCount() {

int count = MessageDao.instance.getModel().size();

return String.valueOf(count);

}

**/\*\***

**\* Used to create a new instance of a Message (message) using the XML posted to the serverlet.**

**\* Creates a new Message (message) object using XML.**

**\* returns text message. 201 on successful creation**

**\***

**\***

**\* @throws IOException**

**\* @throws ClassNotFoundException**

**\* @throws SQLException**

**\*/**

@POST

@Consumes(MediaType.TEXT\_XML + ";charset=ISO-8859-1")

@Produces(MediaType.TEXT\_XML + ";charset=ISO-8859-1")

public Response newMessage(Message Message) throws IOException, ClassNotFoundException, SQLException

{

MessageDao.instance.getModel().put("3", Message);

System.out.println(Message.getText());

String result = "Message saved : " + Message;

return Response.status(201).entity(Message).build();

}

/\*\*

\* Will display only the Message (message) with the set message id

\* @param moMessageid

\* @return

\*/

@Path("{Message}")

public MessageResource getMessage(@PathParam("Message") String moMessageid) {

return new MessageResource(uriInfo, request, moMessageid);

}

}

## PostDataDumperFilter class

**This class was created to be used in conjunction with the servlet used by the API to get and post requests. This class would be called in the WEB.xml. This filter would have obtained all data that was posted to the tomcat server**

package com.vogella.jersey.todo.filters;

import java.io.BufferedReader;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.CharArrayWriter;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.ReadListener;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.ServletInputStream;

import javax.servlet.ServletOutputStream;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.WriteListener;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletRequestWrapper;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpServletResponseWrapper;

public class PostDataDumperFilter implements Filter {

private static class ByteArrayServletStream extends ServletOutputStream {

ByteArrayOutputStream baos;

ByteArrayServletStream(ByteArrayOutputStream baos) {

this.baos = baos;

}

public void write(int param) throws IOException {

baos.write(param);

}

@Override

public boolean isReady() {

// TODO Auto-generated method stub

return false;

}

@Override

public void setWriteListener(WriteListener arg0) {

// TODO Auto-generated method stub

}

}

private static class ByteArrayPrintWriter {

private ByteArrayOutputStream baos = new ByteArrayOutputStream();

private PrintWriter pw = new PrintWriter(baos);

private ServletOutputStream sos = new ByteArrayServletStream(baos);

public PrintWriter getWriter() {

return pw;

}

public ServletOutputStream getStream() {

return sos;

}

byte[] toByteArray() {

return baos.toByteArray();

}

}

private class BufferedServletInputStream extends ServletInputStream {

ByteArrayInputStream bais;

public BufferedServletInputStream(ByteArrayInputStream bais) {

this.bais = bais;

}

public int available() {

return bais.available();

}

public int read() {

return bais.read();

}

public int read(byte[] buf, int off, int len) {

return bais.read(buf, off, len);

}

@Override

public boolean isFinished() {

// TODO Auto-generated method stub

return false;

}

@Override

public boolean isReady() {

// TODO Auto-generated method stub

return false;

}

@Override

public void setReadListener(ReadListener arg0) {

// TODO Auto-generated method stub

}

}

private class BufferedRequestWrapper extends HttpServletRequestWrapper {

ByteArrayInputStream bais;

ByteArrayOutputStream baos;

BufferedServletInputStream bsis;

byte[] buffer;

public BufferedRequestWrapper(HttpServletRequest req) throws IOException {

super(req);

InputStream is = req.getInputStream();

baos = new ByteArrayOutputStream();

byte buf[] = new byte[1024];

int letti;

while ((letti = is.read(buf)) > 0) {

baos.write(buf, 0, letti);

}

buffer = baos.toByteArray();

}

public ServletInputStream getInputStream() {

try {

bais = new ByteArrayInputStream(buffer);

bsis = new BufferedServletInputStream(bais);

} catch (Exception ex) {

ex.printStackTrace();

}

return bsis;

}

public byte[] getBuffer() {

return buffer;

}

}

private boolean dumpRequest;

private boolean dumpResponse;

public void init(FilterConfig filterConfig) throws ServletException {

dumpRequest = Boolean.valueOf(filterConfig.getInitParameter("dumpRequest"));

dumpResponse = Boolean.valueOf(filterConfig.getInitParameter("dumpResponse"));

}

public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse,

FilterChain filterChain) throws IOException, ServletException {

final HttpServletRequest httpRequest = (HttpServletRequest)servletRequest;

BufferedRequestWrapper bufferedRequest= new BufferedRequestWrapper(httpRequest);

if (dumpRequest) {

System.out.println("REQUEST -> " + new String(bufferedRequest.getBuffer()));

}

final HttpServletResponse response = (HttpServletResponse) servletResponse;

final ByteArrayPrintWriter pw = new ByteArrayPrintWriter();

HttpServletResponse wrappedResp = new HttpServletResponseWrapper(response) {

public PrintWriter getWriter() {

return pw.getWriter();

}

public ServletOutputStream getOutputStream() {

return pw.getStream();

}

};

filterChain.doFilter(bufferedRequest, wrappedResp);

byte[] bytes = pw.toByteArray();

response.getOutputStream().write(bytes);

if (dumpResponse) System.out.println("RESPONSE -> " + new String(bytes));

}

public void destroy() {}

}

## Web.xml

**This is the serverlet of the API**

<?xml version=*"1.0"* encoding=*"ISO-8859-1"*?>

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://xmlns.jcp.org/xml/ns/javaee"* xsi:schemaLocation=*"http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"* id=*"WebApp\_ID"* version=*"3.1"*>

<display-name>com.vogella.jersey.message</display-name>

<servlet>

<servlet-name>Jersey REST Service</servlet-name>

<servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>

<!-- Register resources and providers under com.vogella.jersey.todos package. -->

<init-param>

<param-name>jersey.config.server.provider.packages</param-name>

<param-value>com.vogella.jersey.todo.resources</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>Jersey REST Service</servlet-name>

<url-pattern>/rest/\*</url-pattern>

</servlet-mapping>

</web-app>

## MessageProcessing class

**This class is used to split the text message which comes into the system as one long string into separate strings for each piece of data required for the system**

package com.vogella.jersey.todo.processing;

import java.sql.Connection;

import java.sql.Date;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.Time;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

public class MessagesProcessing

{

public static String Process(String text) throws ParseException, ClassNotFoundException, SQLException

{

Boolean multiple = false;

String teamName = "";

String[] teamNames = null;

String fullTeam = "";

String texts = text;

String[] signIn = texts.split(",");

String windSite = signIn[0].trim();

String company = signIn[1].trim();

int departureTime = 0;

//If the text message has more has more than one team member then it will be split each time an & symbol appears,

//if not then the team name is taken as one whole string

if(signIn[2].contains("&") == true)

{

teamNames = signIn[2].trim().split("&");

multiple = true;

}

else

{

teamName = signIn[2].trim();

}

String turbine = signIn[3].trim();

int departure = Integer.parseInt(signIn[4].trim());

Class.forName("postgresql-42.1.3");

Connection m\_Connection = DriverManager.getConnection(

"dbtestinstance.cclcp0zv6ibp.eu-west.1.rds.amazonaws.com:5432;DatabaseName=OPCTrial2", "stephen", "superuser");

Statement m\_Statement = m\_Connection.createStatement();

String query = "SELECT ";

ResultSet m\_ResultSet = m\_Statement.executeQuery(query);

if(windSite.replace(" ", "").compareToIgnoreCase("CarnHill") == 1)

{

windSite = "CHW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Inishative"))

{

windSite = "INW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Ballagh"))

{

windSite = "BAW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Rockmarshall"))

{

windSite = "RMW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Cloonty"))

{

windSite = "CLW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("CorbyKnowe"))

{

windSite = "CKW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("CregganconroeCrockbrack") || windSite.equalsIgnoreCase("Cregganconroe") || windSite.equalsIgnoreCase("Crockbrack"))

{

windSite = "CRW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Dunbeg"))

{

windSite = "DUW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Faughary"))

{

windSite = "FAW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Monnaboy"))

{

windSite = "MBW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Smulgedon"))

{

windSite = "SMW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Tullyneill"))

{

windSite = "TNW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Skrine"))

{

windSite = "SKW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Ballybay"))

{

windSite = "BBWF";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Roosky"))

{

windSite = "GACL";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Cnoc"))

{

windSite = "GNOC";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Leabeg"))

{

windSite = "FCML";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Ballyhanedin"))

{

windSite = "BHW";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("CAES"))

{

windSite = "CAES";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Cloghboola"))

{

windSite = "CLOGHBOOLA";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Foyle"))

{

windSite = "FOYLE";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Cashla"))

{

windSite = "CASHLA";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Portlaoise"))

{

windSite = "PORT";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Corracon"))

{

windSite = "CORR";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("DunbegExt"))

{

windSite = "DUWE";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Ballywater"))

{

windSite = "BWWL";

}

else if(windSite.replace(" ", "").equalsIgnoreCase("Castledockrell"))

{

windSite = "CDWG";

}

String value = (windSite + " " + company + " " + fullTeam + turbine + " " + departure);

return value;

}

}

# Problems encountered so far

So far, the major issue that we have encountered has been to do with getting messages posted to our API. The client add-on, which has been used to test if the api is capable of receiving xml posts returns a 201, which means the object was successfully created on the web app. However, when sending messages via Clickatell, they report a 415-error meaning that whatever they are posting does not match the format of what it is that we are looking for. The first attempted solution to this was changing the content type which the post method was looking for. This did not solve the issue. The next solution was to create a filter which would report all data posted to the web app. This would allow us to see the format which Clickatell was posting in. This was the furthest we have gotten so far as each attempted at making a log of all incoming traffic has only told us that someone has made a post request of the tomcat server, it has not revealed what exactly has been posted.

So far connections to the database have been successful. However, the application may not connect to the database on a computer that does not have the PostgreSQL drivers installed which will allow the local application to read and write data from the database. This is only an issue when testing the application local instead of via the tomcat server.

## What is still left to do

* Receive text messages
* System to check for overdue sign outs
* System to alert Gaelectric staff of overdue sign outs
* Program to allow remote sign out from Gaelectric’s end

# Test plan

## New Ticket

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 1 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 1 | Windfarm Name | - | Carnhil | The system should recognise that the intended name is CarnHill and the ticket will save the shortcode for Carnhill as the name of the windfarm | - | - |
| 2 | Windfarm Name | - | Carnhill | The ticket will save the shortcode for Carnhill as the name of the windfarm | - | - |
| 3 | Windfarm Name | - | Carn hill | The system should recognise that the intended name is CarnHill and the ticket will save the shortcode for Carnhill as the name of the windfarm | - | - |
| 4 | Windfarm Name | - | Faugrey | The system should recognise that the intended name is Faughary and the ticket will save the shortcode for Faughary as the name of the windfarm | - | - |
| 5 | Windfarm Name | - | Faughary | The ticket will save the shortcode for Faughary as the name of the windfarm | - | - |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 2 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 6 | Windfarm Name | - | Faughrey | The system should recognise that the intended name is Faughary and the ticket will save the shortcode for Faughary as the name of the windfarm | - | - |
| 7 | Windfarm Name | - | Shrine | The system should recognise that the intended name is Skrine and the ticket will save the shortcode for Skrine as the name of the windfarm | - | - |
| 8 | Windfarm Name | - | Skrine | The ticket will save the shortcode for Skrine as the name of the windfarm | - | - |
| 9 | Windfarm Name | - | Scrine | The system should recognise that the intended name is Skrine and the ticket will save the shortcode for Skrine as the name of the windfarm | - | - |
| 10 | Windfarm Name | - | Cloghbola | The system should recognise that the intended name is Cloghboola and the ticket will save the shortcode for Cloghboola as the name of the windfarm | - | - |
| 11 | Windfarm Name | - | Cloghboola | The ticket will save the shortcode for Cloghboola as the name of the windfarm | - | - |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 3 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 12 | Windfarm Name | - | Knocknagashiel | The system should recognise that the intended name is Cloghboola and the ticket will save the shortcode for Cloghboola as the name of the windfarm | - | - |
| 13 | Windfarm Name | - | Corby Know | The system should recognise that the intended name is CorbyKnowe and the ticket will save the shortcode for CorbyKnowe as the name of the windfarm | - | - |
| 14 | Windfarm Name | - | CorbyKnow | The system should recognise that the intended name is CorbyKnowe and the ticket will save the shortcode for CorbyKnowe as the name of the windfarm | - | - |
| 15 | Windfarm Name | - | CorbyKnowe | The ticket will save the shortcode for CorbyKnowe as the name of the windfarm | - | - |
| 16 | Windfarm Name | - | Monaboy | The system should recognise that the intended name is Monnaboy and the ticket will save the shortcode for Monnaboy as the name of the windfarm | - | - |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 4 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 17 | Windfarm Name | - | Inishative | The ticket will save the shortcode for Inishative as the name of the windfarm | - | - |
| 18 | Windfarm Name | - | Inishativ | The system should recognise that the intended name is Inishative and the ticket will save the shortcode for Inishative as the name of the windfarm | - | - |
| 19 | Windfarm Name | - | Initiative | The system should recognise that the intended name is Inishative and the ticket will save the shortcode for Inishative as the name of the windfarm | - | - |
| 20 | Windfarm Name | - | Leebeg | The system should recognise that the intended name is Leabeg and the ticket will save the shortcode for Leabeg as the name of the windfarm | - | - |
| 21 | Windfarm Name | - | Roskey | The system should recognise that the intended name is Rooskey and the ticket will save the shortcode for Rooskey as the name of the windfarm | - | - |
| 22 | Windfarm Name | - | Rooskey | The ticket will save the shortcode for Rooskey as the name of the windfarm | - | - |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 5 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 23 | Windfarm Name | - | Castledockrel | The system should recognise that the intended name is Castledockrell and the ticket will save the shortcode for Castledockrell as the name of the windfarm | - | - |
| 24 | Windfarm Name | - | Ballywater | the ticket will save the shortcode for Ballywater as the name of the windfarm | - | - |
| 25 | Windfarm Name | - | Bally Water | The system should recognise that the intended name is Ballywater and the ticket will save the shortcode for Ballywater as the name of the windfarm | - | - |
| 26 | Area of work | Boundary test | t | Sender should receive an error message stating that the area of work is not valid | - | - |
| 27 | Area of work | Boundary test | T1 | New ticket should be saved with T1 as the area of work | - | - |
| 28 | Area of work | Boundary test | Sub | New ticket should be saved with sub as the area of work | - | - |
| 29 | Area of work | Boundary test | Sub, T8, T9, T10, T11 | New ticket should be saved with Sub, T8, T9, T10, T11 as the area of work | - | - |
| 30 | Area of work | Boundary test | Sub, T9, T10, T11, T12 | New ticket should be saved with Sub, T9, T10, T11, T12 as the area of work | - | - |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 6 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 31 | Area of work | Boundary test | Sub, T9, T10, T11, T123 | Sender should receive an error message stating that the area of work is not valid | - | - |
| 32 | Estimated departure time | Boundary test | 9 | Sender should receive an error message stating that the departure time is not valid | - | - |
| 33 | Estimated departure time | Boundary test | 10 | The departure time should be added onto the time which the message is received in order to obtain the estimated departure time | - | - |
| 34 | Estimated departure time | Boundary test | 11 | The departure time should be added onto the time which the message is received in order to obtain the estimated departure time | - | - |
| 35 | Estimated departure time | Boundary test | 479 | The departure time should be added onto the time which the message is received in order to obtain the estimated departure time | - | - |
| 36 | Estimated departure time | Boundary test | 480 | The departure time should be added onto the time which the message is received in order to obtain the estimated departure time | - | - |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** New ticket | | |  | **Page:** 7 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 37 | Estimated departure time | Boundary test | 481 | Sender should receive an error message stating that the departure time is not valid | - | - |
| 38 | Estimated departure time | Equivalence Partitioning | 5 | Sender should receive an error message stating that the departure time is not valid | - | - |
| 39 | Estimated departure time | Equivalence Partitioning | 150 | The departure time should be added onto the time which the message is received in order to obtain the estimated departure time | - | - |
| 40 | Estimated departure time | Equivalence Partitioning | 560 | Sender should receive an error message stating that the departure time is not valid | - | - |
| 41 | New ticket | Unit testing | Carn hill, Enercon, John smith, T1 & T2, 250 | Site visitor should receive a text message with a 4-digit code. New entry should be made in sign in table. Clickatell should report 201 response | - | - |

## Sign out

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** Sign out | | |  | **Page:** 1 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input/conditions** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 1 | Sign out | Unit testing | 1001 (Open ticket number) | The ticket should be closed and the visitor signed out. Represented by Boolean entry in sign in table | - | - |
| 2 | Sign out | Unit testing | 1000 (Closed ticket number) | The visitor should receive a text message saying that this ticket had already been signed out. | - | - |
| 3 | Sign out | Unit testing | 23 (Ticket number does not exist) | The visitor should receive a text message saying that this ticket does not exist | - | - |
| 4 | SMS Alert | Unit testing | Ticket not closed before estimated departure time | Text message sent out to Gaelectric number. | - | - |
| 5 | SMS Alert | Unit testing | Ticket closed before estimated departure time | No alerts should be sent out | - | - |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **SYSTEM NAME**: Gaelectric Sign in / Sign out  **PROGRAM NAME:** com.vogella.jersey.todo  **MODULE NAME:** Remote Sign out | | |  | **Page:** 1 |  |  |
| **Test No** | **Test Item** | **Type of test** | **Input/conditions** | **Expected Outcome** | **Actual Outcome** | **Corrective Action** |
| 1 | Ticket No | Boundary test | 999 | Error message | - | - |
| 2 | Ticket No | Boundary test | 1000 | Entry accepted | - | - |
| 3 | Ticket No | Boundary test | 1001 | Entry accepted | - | - |
| 4 | Search button | Unit test | 1000 | Sign in details should be displayed in text fields below the buttons. | - | - |
| 5 | Search button | Unit test | 2301 (non-existent ticket) | Error message stating that the entered ticket no does not exist | - | - |
| 6 | Sign out button | Unit test | 1000 | Boolean field for the corresponding record should be set to one. Indicating that the ticket has been resolved and the visitor signed out. | - | - |

## Remote sign out