Open Footprint™ Forum
Global Event
June 23-24, 2021
Open Footprint™ Calculation Engine Design

William Fox
Chief Product Officer
Data Gumbo Corporation

Javier Espinoza
VP of Engineering
Data Gumbo Corporation
# Scenarios of Interest

<table>
<thead>
<tr>
<th>Option</th>
<th>Implications</th>
<th>Pros &amp; Cons</th>
</tr>
</thead>
</table>
| **Scenario 1** Calculations can be created by user within OFP Application Platform | • Allowing OFP user to build (drag and drop) its own calculations within the Application platform  
• OFP to provide parameter options (e.g., conversion rates, fuel types etc.) based on OFP data model  
• Maintain data lineage by calculations made by user, there is no ‘correctness’ check on the calculation | + Flexible  
+ Low Maintenance  
= Weak Data Lineage  
= Low benefit |
| **Scenario 2** Integrate standard calculations into OFP Application Platform | • OFP to provide standard calculation functions (mathematical equations) based on available standards on the market and user selects appropriate calculation based on situation  
• OFP to update the calculations on a regular basis if standards change  
• Maintain data lineage from calculations made by user | + Data Lineage  
= High Maintenance |
| **Scenario 3** Integrate standard calculations into OFP Application Platform with reporting body’s responsibility to update | • OFP to provides standard calculation functions (mathematical equations) within Application Platform and user selects appropriate calculation based on situation  
• Reporting bodies are responsible for correctness and maintenance of the calculation’s formulas on the OFP platform. OFP will remain responsible for the release management  
• Maintain data lineage from calculations made by user, there is no ‘correctness’ check on the calculation | + Medium Maintenance  
+ Data Lineage |
| **Scenario 4** External calculations are being called to the OFP platform for output data creation | • All (including complex and industry specific) calculations can be retrieved from external sources via API calls  
• Calculations happen inside OFP platform  
• Maintenance of calculations will be performed outside OFP platform, OFP is not responsible  
• Maintain data lineage from calculations and produce all metadata | + Low Maintenance  
+ Data Lineage  
= Complexity  
= Availability |
| **Scenario 5** OFP uses external calculation engines via API calls | • OFP shares input data with external calculation engines and retrieves output data via API calls  
• Calculations happen outside of OFP platform  
• OFP is not responsible for the maintenance of the calculation engine  
• Sending data in and out (across countries) can have legal and security implications  
• Produce data lineage from calculations based on metadata |
Model Assumptions

» Activity parameters, Default Factors, Formulas have been properly stored in OFP data platform service

» Authentication and Authorization is not part of the solution at this moment
Main Attributes

» Decoupled
  – Compartmentalized Engine Modules

» Extensible
  – More calculations and features can be added

» Adaptable
  – Works with different interfaces
  – Utilizing adapters for compatibility

» Multitier
  – Engine architect includes multiple data transformation and calculation layers
Open-Source Libraries

Based on the requirements of the calculation engine being able to handle multiples use case scenarios, a business rule engine is to be used in conjunction with the calculation engine.

Calculation Engine Recommendation:

- Mathjs
  - https://mathjs.org/

- JavaScript Expression Evaluator (Silent Matt)
  - https://github.com/silentmatt/expr-eval

- Math-expression-evaluator (Bugwheels-94)
  - http://bugwheels94.github.io/math-expression-evaluator/
Business Rule Recommendation:

- Drools Engine (Java)
  - [https://www.drools.org/](https://www.drools.org/)

- NPM Node Rules
  - [https://github.com/silentmatt/expr-eval/](https://github.com/silentmatt/expr-eval/)
## Calculation Engine Capabilities

<table>
<thead>
<tr>
<th>Simple Linear</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition ‘+’</td>
<td>Trigonometry</td>
</tr>
<tr>
<td>Subtraction ‘−’</td>
<td>Exponential</td>
</tr>
<tr>
<td>Multiplication ‘*’</td>
<td>Square root</td>
</tr>
<tr>
<td>Division</td>
<td>Function Parsing</td>
</tr>
<tr>
<td>Grouping ‘( )’</td>
<td></td>
</tr>
</tbody>
</table>
Calculation Engine Design – Execution Model

- Trigger mechanism
- Factors, Activity, Formula
- Variable mapping
- Report preparation
  Support multiple formats

Math expression parser
Evaluator
Output Writer & Adapter

Rest API
Timer
Message Consumer

Read work product components
Write lineage data
Write work product
Write lineage data

Note: For a Math Parser and Evaluator, open-source libraries such as **math.js** (backend) and **MathJax** (UI) could be utilized.

Copyright © The Open Group 2021
Engine Design – Local Calculator

1. UI sends one or a list of activities and factors (optional)

2. Call Reader after validation

3. Fetch:
   - Calc Model / Formula
   - Default factors (optional)

4. Call Parser after mapping variables

5. Write lineage data

6. Pass results

7. Write GHG Statement

Diagram:
- Rest API
- Timer
- Message Consumer
- Input Reader & Adapter
- Math expression parser
- Evaluator
- Output Writer & Adapter
- OFP Data platform
Engine Design – Remote Calculator

1. UI sends one or a list of activities and factors (optional)

2. Call Reader after validation

3. Fetch:
   - Calc Model / Formula
   - Default factors (optional)

4. Call Remote Calculator w/ required payload

5. Get results

6. Write GHG Statement

- Rest API
- Timer
- Message Consumer
- Input Reader & Adapter
- Math expression parser
- Evaluator
- Output Writer & Adapter

OFP Data platform
Part 2 – Engine Examples
Part 2.1 – Business Rule Examples
Option 1: Drools Engine (Redhat - Java)

Define rules
e.g. Patient pays fee according his age
Option 2: NPM Node Rule Engine (JavaScript)

Decision Tree Example:

```
var RuleEngine = require('.../Index');

/* Set of Rules to be applied */
First blocks a transaction if less than 500
Second blocks a debit card transaction.*/
/* Note that here we are not specifying which rule to apply first.
Rules will be applied as per their index in the array.
If you need to enforce priority manually, then see examples with prioritized rules */
var rules = [{
    "condition": function(R) {
        R.when(this.transactionTotal < 500);
    },
    "consequence": function(R) {
        this.result = false;
        this.reason = "The transaction was blocked as it was less than 5000";
        R.stop(); // stop if matched. no need to process next rule.
    }
}];

R = new RuleEngine();
R.register(rules);
```

1st condition: Transaction Amount > 500?
2nd condition: Using Debit Card?
/* Fact with more than 500 as transaction but a Debit card, and this should be blocked */

var fact = {
    "name": "user4",
    "application": "MOB2",
    "transactionTotal": 600,
    "cardType": "Debit"
};

R.execute(fact, function(data) {
    if (data.result) {
        console.log("Valid transaction");
    } else {
        console.log("Blocked Reason:" + data.reason);
    }
});
Part 2.2 – Calculation Examples
Example 1: Stationary Combustion (Simple)

Example Equation for evaluator:

\[ CO_{2} \text{ Emission Amount} = \text{Fuel Consumption} \times CO_{2}\text{-Conversion Factor} \]

\[ CH_{4} \text{ Emission Amount} = \text{Fuel Consumption} \times CO_{2}\text{-Conversion Factor} \]
Example 2:– Transportation (Medium)

» Assumption:
  – ITS Berkeley Study Data, taking into effects of fuel weight change during flights and taking off burns a large amount of fuel comparing to overall flight
    • A = 114,256
    • B = 2,187,696
    • C = 1.000123

» Equation: \[ \frac{CO_2e}{passenger} = A + B \left[ C^{miles} - 1 \right] \]
DEMONSTRATION

» Live screen share
Part 3 – Engine Development
Business Rule + Calculation Engine

» RESTful APIs for UI (1 – 1.5 week)

» Engine Creation: (2 – 3 weeks)
  – Drools Engine + exp4J Engine (Java); or
  – NPM Node Rules + Math JS (JavaScript)

» Integration with OSDU (2 – 3 weeks)

» Total estimated time: 5 – 7.5 weeks
Thank You!