Rule Engine API

The Application programmer Interface for the Mobile Rule Engine developed for Emdeon contains a JavaScript class by the name RuleEvaluator. The class is automatically generated by the XLS2SLANG tool.

1. The constructor for the RuleEvaluator

```
function RuleEvaluator()

Returns:-
      Constructor do not have any return value.
Usage:-
      var rule_eval = new RuleEvaluator();
```

All the data required to instantiate the engine is automatically populated by the tool which generated the API implementation. We can create multiple instances of RuleEvaluator and for each instance, there is a truth table.

2. The Rule dictionary (created by application programmer) has to be assigned to the Engine using the SetCurrentEnvironment API call

```
//////////////////////////////////////////////////
//
// This method gives a rule dictionary as parameter ...and
// The engine creates a clone of  the dictionary and an automatic.
//  rule evaluation will be triggered.

RuleEvaluator.prototype.SetCurrentEnvironment = function ( prule_dict );

Returns :- void

Usage :-
                   rule_dict = {};
                   rule_dict["a"]=10;
                   rule_dict["b"]=12;
                   rule_dict["c"]=20;
                   rule_dict["Insurance"]=false;
                   rule_dict["CitizenShip"]='American';

                   var rule_eval = new RuleEvaluator();
                   rule_eval.SetCurrentEnvironment(rule_dict);
```

3. EvaluateAll API will allow us to evaluate all the business rules ,generate truth tables and determine eligibility in one go. The routine returns the list of programs which maps program Name to a Boolean value as given below.

   Program_return_value[“T16”] = true;
   Program_return_value[“MSAA”] = false;

```
RuleEvaluator.prototype.EvaluateAll = function( );

Usage:-
                     rule_dict = {};
                    rule_dict["a"]=10;
                    rule_dict["b"]=12;
                    rule_dict["c"]=20;
                    rule_dict["Insurance"]=false;
                    rule_dict["CitizenShip"]='American';

                    var rule_eval = new RuleEvaluator();
                    rule_eval.SetCurrentEnvironment(rule_dict);
                   var program_return_value = rule_eval.EvaluateAll();
                  //------------------ Do whatever you want to do with the program
```

4. The RulEvaluator is a stateful API and hints has to be given to the engine to start keeping track of the changes made to the facts using ResetEvaluationContext API.

```
RuleEvaluator.prototype.ResetEvaluationContext = function()

Returns :- void

Usage:-
                     rule_dict = {};
                    rule_dict["a"]=10;
                    rule_dict["b"]=12;
                    rule_dict["c"]=20;
                    rule_dict["Insurance"]=false;
                    rule_dict["CitizenShip"]='American';

                    var rule_eval = new RuleEvaluator();
                    rule_eval.SetCurrentEnvironment(rule_dict);
                   var program_return_value = rule_eval.EvaluateAll();
```

```
                    //------------------ Do whatever you want to do with the program
                    Rule_eval.ResetEvaluationContext();
```

5. If we need to change the fact inside the RuleEngine,we can use ChangeFact API

```
RuleEvaluator.prototype.ChangeFact = function (key,value);

Returns:- void

Usage:-
                    rule_dict = {};
                    rule_dict["a"]=10;
                    rule_dict["b"]=12;
                    rule_dict["c"]=20;
                    rule_dict["Insurance"]=false;
                    rule_dict["CitizenShip"]='American';

                    var rule_eval = new RuleEvaluator();
                    rule_eval.SetCurrentEnvironment(rule_dict);
                    rule_eval.ResetEvaluationContext();
                    rule_eval.ChangeFact("a",-1);
                    rule_eval.ChangeFact("Insurance",true);
                    var program_return_value = rule_eval.EvaluateDelta();
```

6. EvaluateDelta API , allows incremental rule evaluation.

```
RuleEvaluator.prototype.EvaluateDelta = function() ;

Returns :- a dictionary which maps program name to a Boolean value
Usage :-
        Consult the program snippets of the previous entry
```

The driver program which was used to test the working of prototype API is given below.

```
////////////////////
//
// Driver program to Test the Rule Evaluator...
//
//
console.log("============================================================");

var rule = new RuleEvaluator();

console.log("============Populating Rule Dictionary");
rule_dict = {};
rule_dict["a"]=10;
rule_dict["b"]=12;
rule_dict["c"]=20;
rule_dict["Insurance"]=false;
rule_dict["CitizenShip"]='American';


console.log("===============Set the rule dictionary to the engine");

rule.SetCurrentEnvironment(rule_dict);

console.log("================ Evaluate All and Spit the eligibility ");

var program_dict = rule.EvaluateAll();
for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);

console.log("==============Reset Evaluation Context,Change Fact a = -1 ");

rule.ResetEvaluationContext();
rule.ChangeFact("a",-1);
program_dict = rule.EvaluateDelta();

for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);

console.log("==============Reset Evaluation Context,Change Citizenship to Asian");
rule.ResetEvaluationContext();
rule.ChangeFact("Citizenship",'Asian');
program_dict = rule.EvaluateDelta();

for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);

console.log("==============Reset Evaluation Context,Change a = 2");
```

```
rule.ResetEvaluationContext();
rule.ChangeFact("a",2);
program_dict = rule.EvaluateDelta();

 for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);
```

The output on the console is as given below

```
G:\PhilosophyLiterature\BusinessRules\BACKTRACK3>node Rules.js
============================================================
============Populating Rule Dictionary

===============Set the rule dictionary to the engine

================ Evaluate All and Spit the eligibility

Elligibility for ELLIGIBLE_PROGRAM_ONE= true
Elligibility for ELLIGIBLE_PROGRAM_TWO= true
==============Reset Evaluation Context,Change Fact a = -1

Elligibility for ELLIGIBLE_PROGRAM_ONE= false
Elligibility for ELLIGIBLE_PROGRAM_TWO= true
==============Reset Evaluation Context,Change Citizenship to Asian

Elligibility for ELLIGIBLE_PROGRAM_ONE= false
Elligibility for ELLIGIBLE_PROGRAM_TWO= false
==============Reset Evaluation Context,Change a = 2

Elligibility for ELLIGIBLE_PROGRAM_ONE= true
Elligibility for ELLIGIBLE_PROGRAM_TWO= false

G:\PhilosophyLiterature\BusinessRules\BACKTRACK3>
```

The JavaScript program which was used to identify the templates for code generation is given below.

```
///////////////////////////////////////////////////
//
//
// The Rule Engine ...Object ....!
//
//
//


function RuleEvaluator() {


  /////////////////////////////
  // Let the rule engine ...own the rule dictionary....
  //
  this.rule_dict = this.RetrieveRuleDictionary();

  /////////////////////////////////////////////////////////////////
  //
  // Elligibility is determined by the content of truth dictionary...
  this.truth_dict = this.RetrieveTruthDictionary();

  ////////////////////////////
  // This array keeps track of the variables to be evaluated
  //
  this.eval_array = [];
  this.invoke_dict = this.RetrieveInvocationDictionary();
  this.invoke_program_dict = this.RetrieveInvokeProgramDictionary();
  this.depend_dict = this.RetrieveDependencyDictionary();
}



///////////////////////////
//
// Rule Dictionary has to be supplied
// by Application developer..
//
//
// The Application Programmer has to create this
// data structure by consulting "VariableMappings"
// tab inside the spreadsheet... ( This has to be hand tweaked )
//
```

```
RuleEvaluator.prototype.RetrieveRuleDictionary= function() {
        rule_dict = {};
        rule_dict["a"]=10;
        rule_dict["b"]=12;
        rule_dict["c"]=20;
        rule_dict["Insurance"]=false;
        rule_dict["CitizenShip"]='American';
    return rule_dict;
}

////////////////////////////////////
//
//
// Invocation Dictionary ...
//    This dictionary will automatically be generated by the
//    XLS2SLANG tool... ( Automatic )
//
//
RuleEvaluator.prototype.RetrieveInvocationDictionary= function() {

        invoke_dict = {}
        invoke_dict["R11"] = "R11(a)";
        invoke_dict["R12"] = "R12(b,a)";
        invoke_dict["R13"] = "R13(a,b)";
        invoke_dict["R14"] = "R14(Citizenship)";
        invoke_dict["R15"] = "R15(Insurance)";
        return invoke_dict;
}



///////////////////////////////////////
//
//
// Dependency Dictionary....
//
//    Automatically generated by XLS2SLANG tool...
//    (Automatic)
//
RuleEvaluator.prototype.RetrieveDependencyDictionary = function(){
        depend_dict = {};
        depend_dict["a"] = ["R11","R12","R13"];
        depend_dict["b"] = ["R12","R13"];
        depend_dict["c"] = [];
        depend_dict["Citizenship"] = [ "R14" ];
        depend_dict["Insurance"] = ["R15"];
        return depend_dict;
```

```
}

////////////////////////////////////////////
// Truth dictionary - It is meant for Incremental
// rule evaluation...
// This is maintained by the program and will be automatically
// generated   ( Automatic )
//
//
//
//

RuleEvaluator.prototype.RetrieveTruthDictionary = function() {


             truth_dict = {};
             return truth_dict;

}

//////////////////////////////////////////////
//
// Rules ....which are generated by the
// XLS2SLANG program from Excel template...
//
//

function R11( a ) {

   return a > 0;
}

function R12( a, b  ) {

   return  ( a - b ) > 0;
}

function R13( a, b  ) {

   return  ( b - a ) > 0;
}

function R14(CitizenShip ) {

  return CitizenShip == 'Asian';
}

function R15(Insurance) {
  return Insurance == false;
}
///////////////////////////////////////////////
//
//
// Program to Rule mappings..To be generated by
// XLS2SLANG tool...
//

function ELLIGIBLE_PROGRAM_ONE( R11,R15 )
{
   return  ( R11 == true ) && (R15 == true) ;

}

function ELLIGIBLE_PROGRAM_TWO( R14 )
{
   return  ( R14 == false ) ;

}

///////////////////////////////////
//
// Helper methods to be generated by the XLS2SLANG tool
//
RuleEvaluator.prototype.RetrieveInvokeProgramDictionary = function() {
            //////////////////////////////
            //
            // invoking eligibility rules...
            //
            //
            //
            invoke_program_dict = {}
            invoke_program_dict["ELLIGIBLE_PROGRAM_ONE"] = "ELLIGIBLE_PROGRAM_ONE( R11,R15 )";
            invoke_program_dict["ELLIGIBLE_PROGRAM_TWO"] = "ELLIGIBLE_PROGRAM_TWO( R14 )";
            return invoke_program_dict;

}

///////////////////////////////////////////////////////
//
// API for the application programmers....
//
//
///////////////////////////////////////////////
//
// This method gives a rule dictionary as parameter ...and
// The engine creates a clone of it...
//
RuleEvaluator.prototype.SetCurrentEnvironment = function ( prule_dict )
{

        for (var cat in prule_dict )
        {
             this.rule_dict[cat] = prule_dict[cat];


        }

        this.EvaluateAll();
}

////////////////////////////////////////////////
//
//
RuleEvaluator.prototype.GetCurrentEnvironment = function ( )
{

        return this.rule_dict;
}




RuleEvaluator.prototype.ChangeFact = function (key,value) {

            this.rule_dict[key]=value;
            this.eval_array.push(key);
}


RuleEvaluator.prototype.ResetEvaluationContext = function()
{
    this.eval_array = [];

}

///////////////////////////////////////////
//
// Rule Engine .... Evaluate all method...
//
//

RuleEvaluator.prototype.EvaluateAll = function( )
{
   return this.Evaluate(this.rule_dict,null);


}

RuleEvaluator.prototype.EvaluateDelta = function() {

   return this.Evaluate(this.rule_dict,this.eval_array);
}


////////////////////////
//
// Private API , use with care
//
//
RuleEvaluator.prototype.EvaluateAllExternal = function( prule_dict ) {

    return this.Evaluate(prule_dict,null);
}


RuleEvaluator.prototype.EvaluateDeltaExternal = function( prule_dict ,param_array ) {

    return this.Evaluate(prule_dict,param_array);
}


/////////////////////////////////
//
//
//
//

RuleEvaluator.prototype.ProgramEvaluation = function()
{

 var R11 = this.truth_dict["R11"];
 var R12 = this.truth_dict["R12"];
 var R13 = this.truth_dict["R13"];
 var R14 = this.truth_dict["R14"];
 var R15 = this.truth_dict["R15"];

 var program_truth = {};

 for (var cat in this.invoke_program_dict )
 {
       if (  eval(this.invoke_program_dict[cat]) == true )
       {
            program_truth[cat]=true;
       }
       else
            program_truth[cat]=false;


 }

  return program_truth;


}

//////////////////////////////////
//
//
//  Evaluate all the rules.....
//
//
RuleEvaluator.prototype.Evaluate = function(prule_dict,param_array) {
   var  a  = prule_dict["a"];
   var  b  = prule_dict["b"];
   var  c  = prule_dict["c"];
   var Insurance = prule_dict["Insurance"];
   var Citizenship = prule_dict["Citizenship"];


   if ( param_array == null || param_array.length == 0)
   {
       for (var cat in this.invoke_dict )
           this.truth_dict[cat] = eval(this.invoke_dict[cat]);

       return this.ProgramEvaluation();
   }


   var index;
   for ( index = 0; index < param_array.length; ++index) {
       var str = param_array[index];


       var arr = this.depend_dict[str];
       if ( arr == null || arr.length == 0 )
                continue;
       for (var index2 = 0; index2 < arr.length; ++index2)
           this.truth_dict[arr[index2]] = eval(this.invoke_dict[arr[index2]]);
```

```
    }

    return this.ProgramEvaluation();

}





////////////////////
//
// Driver program to Test the Rule Evaluator...
//
//
console.log("============================================================");

var rule = new RuleEvaluator();

console.log("============Populating Rule Dictionary\r\n");
rule_dict = {};
rule_dict["a"]=10;
rule_dict["b"]=12;
rule_dict["c"]=20;
rule_dict["Insurance"]=false;
rule_dict["CitizenShip"]='American';


console.log("===============Set the rule dictionary to the engine\r\n");

rule.SetCurrentEnvironment(rule_dict);

console.log("n================ Evaluate All and Spit the eligibility \r\n");

var program_dict = rule.EvaluateAll();
for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);

console.log("==============Reset Evaluation Context,Change Fact a = -1\r\n ");

rule.ResetEvaluationContext();
rule.ChangeFact("a",-1);
program_dict = rule.EvaluateDelta();

for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);
console.log("==============Reset Evaluation Context,Change Citizenship to Asian\r\n");
rule.ResetEvaluationContext();
rule.ChangeFact("Citizenship",'Asian');
program_dict = rule.EvaluateDelta();

for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);

console.log("==============Reset Evaluation Context,Change a = 2\r\n");

rule.ResetEvaluationContext();
rule.ChangeFact("a",2);
program_dict = rule.EvaluateDelta();

 for( var el in program_dict )
  console.log("Elligibility for " + el + "= " + program_dict[el]);
```