# RuleMSX Documentation

### Overview

RuleMSX is a library which provides the core functionality of a Rule Engine. It is designed to inter-operate with the EasyMSX and EasyMKT libraries which use the Bloomberg API to access Bloomberg EMSX and market data.  
  
This functionality is provided in the shape of RuleSets, DataSets and Actions. By defining Rules and the conditions that must exist for these Rules to be triggered, the user can build complex reasoning based on the content of a DataSet, and how that DataSet changes over time. The Actions are the tasks performed as a result of a Rule being triggered

### RuleSets

## RuleSets

sdfdsgdsfsdf

## ExecutionAgent

When the application has completed the configuration of all the main elements (Rules, RuleConditions, Evaluators, Action, Executors, etc.), one or more RuleSets can be executed.  
  
This involves taking a DataSet and asking the RuleSet to be executed against that DataSet: -  
  
 myRuleSet.Execute(dataSet\_1);  
  
If this is the first time this RuleSet has been executed, a new ExecutionAgent will be created for the RuleSet. If the RuleSet already has an ExecutionAgent, it will be reused. The specified DataSet is then passed to the RuleSet's ExecutionAgent: -  
  
 executionAgent = new ExecutionAgent(myRuleSet, dataSet\_1);  
  
or  
  
 executionAgent.AddDataSet(dataSet\_1);  
  
Each ExecutionAgent has a DataSetQueue. Adding a DataSet to an ExecutionAgent simply adds the DataSet reference into the DataSetQueue. This is used to ensure that new DataSets are only ingested at the correct time, and not at the mid-point of a cycle.

A new ExecutionAgent will create a new internal thread that will operate a WorkingSetAgent. This WorkingSetAgent is the main loop that controls execution of the rules and actions for a RuleSet, and it continues to run until stopped by an external request (a call to the stop() method).   
  
Each cycle of the WorkingSetAgent begins with ingesting any DataSets in the ExecutionAgent's DataSetQueue. This is the process of creating a WorkingRule for each Rule in the RuleSet and the specified DataSet.

To create a WorkingRule, a Rule and a DataSet are required. A process known as dereferencing takes place, which has two steps. The first step is to take each Action associated with the Rule, and add the ActionExecutor references to the WorkingRule’s Executors collection.  
  
The second part of the dereferencing process is to iterate each RuleCondition of the Rule, and add it’s RuleEvaluator to the Evaluators collection of the WorkingRule. Each RuleEvaluator has a collection of DataPoint names that it depends on. For each of these dependant data point names, we find the actual DataPoint in the DataSet that matches the name. The WorkingRule is then added to the AssociatedWorkingRules collection of the DataPoint’s DataPointSource object.  
  
The reason for doing this is that when a DataPointSource’s value changes, its SetStale() method is (should be) fired. This forces each WorkingRule dependency of the DataPointSource to be added to the OpenSetQueue in the WorkingSetAgent for execution in the next cycle, unless the WorkingRule is already in the OpenSetQueue.   
  
Following the ingestion process, the current OpenSetQueue becomes the OpenSet, and the OpenSetQueue is then reset to empty.  
  
The OpenSet is now iterated, and each WorkingRule in the queue is processed. Each Evaluator in the WorkingRule is fired, passing it the WorkingRule’s DataSet. If all Evaluators in the WorkingRule return true, then the action process begins. Each action associated with the WorkingRule is executed.