

DMCommunity Challenge “Soldier Payment Rules”

A Solution with OpenRules Decision Manager

This [challenge](#) asks to “assemble a single timeline for the soldier over a given service period that shows his/her hourly pay rate in any given time.” There are several interesting solutions that consider different time intervals and possible database organization.

This solution assumes that there are two services:

- Service 1. “Get Data for a given soldier and date”
- Service 2. “Calculate Pay Rate for a given soldier and date”.

Service 1 can be implemented differently using SQL or other means, but it is important that for a given soldier and service date it will produce the soldier’s rank, profession, service type, unit, and combat. Service 2 will use this data to calculate the proper pay rate.

Here is the implementation of Service 2 as a very simple decision service using OpenRules Decision Manager. When it’s deployed as AWS Lambda function, we may execute it as any RESTful service, for example using POSTMAN:

The screenshot displays a Postman REST client interface. At the top, a POST request is configured to the URL `https://81n1qmlrqf.execute-api.us-east-1.amazonaws.com/test/soldier-payments`. The request body is a JSON object with the following structure:

```
1  {
2    "problemRequest": {
3      "soldier": "Bill",
4      "serviceDate": "2023-10-16",
5      "rank": "private",
6      "profession": "fighter",
7      "serviceType": "active",
8      "unit": "marines",
9      "combat": "yes"
10   }
11 }
```

The response body is also shown in JSON format, indicating a successful 200 OK status with a response time of 46 ms and a body size of 412 B. The response structure is as follows:

```
1  {
2    "decisionStatusCode": 200,
3    "rulesExecutionTimeMs": 0.611522,
4    "response": {
5      "problemResponse": {
6        "rate": 13
7      }
8    }
9  }
```

Here is how it was implemented using a few Excel-based tables. First of all, here is the corresponding business glossary:

Glossary glossary					
Variable	Business Concept	Attribute	Type	Default Value	Used As
Soldier	ProblemRequest	soldier	String		in
Service Date		serviceDate	Date		in
Base Rate		baseRate	Integer	\$1	in
Rank		rank	String		in
Profession		profession	String		in
Service Type		serviceType	String		in
Unit		unit	String		in
Combat		combat	String		in
Pay Rate	ProblemResponse	rate	Integer		out
Errors		errors	String[]		out
Rank Rate	Rates	rankRate	Integer		
Profession Rate		professionRate	Integer		
Service Type Rate		serviceTypeRate	Integer		
Unit Rate		unitRate	Integer		
Combat Rate		combatRate	Integer		

As you can see, we expect that ProblemRequest contains all soldier's characteristics. Even "Base Rate" could come from outside or we will use the default value of \$1.

Here is the main decision table:

DecisionTable DeterminePayRate		
Condition		Conclusion
Errors		Pay Rate
Is Empty	TRUE	Base Rate + Rank Rate + Profession Rate + Service Type Rate + Unit Rate + Combat Rate
Is Empty	FALSE	\$0

The main goal of this decision service is defined as "Pay Rate" and OpenRules will automatically define that is depends on other characteristics mentioned in the formula "Base Rate + Rank Rate + Profession Rate + Service Type Rate + Unit Rate + Combat Rate". Here are other decision tables that calculate these characteristics:

DecisionTable DetermineRankRate			
Condition	Conclusion	Conclusion	
Rank	Rank Rate	Errors	
private	\$1		
corporal	\$2		
sergeant	\$3		
lieutenant	\$4		
captain	\$5		
	0	Add	Unknown Rank for {{Soldier}}

DecisionTable DetermineProfessionRate			
Condition	Conclusion	Conclusion	
Profession	Profession Rate	Errors	
fighter	\$2		
driver	\$1		
cook	\$1		
officer	\$3		
	0	Add	Unknown Profession for {{Soldier}}

DecisionTable DetermineServiceTypeRate			
Condition	Conclusion	Conclusion	
Service Type	Service Type Rate	Errors	
active	\$2		
reserve	\$1		
retired	\$0		
	0	Add	Unknown Service Type for {{Soldier}}

DecisionTable DetermineUnitRate			
Condition	Conclusion	Conclusion	
Unit	Unit Rate	Errors	
HQ	\$1		
paratroopers	\$2		
marines	\$2		
infantry	\$2		
	0	Add	Unknown Unit for {{Soldier}}

DecisionTable DetermineCombatRate			
Condition	Conclusion	Conclusion	
Combat	Combat Rate	Errors	
yes	\$5		
no	\$0		
	0	Add	Unknown Combat for {{Soldier}}

If some characteristics are not defined or not expected, the proper Errors will be produced. Nothing else is required. To deploy this decision model as an AWS Lambda (see the image above), we used the standard OpenRules bat-file "deployLambda.bat".

Probably, these rules are too simple, and if we keep the pay rates for different characteristics in a database the problem can be completely solved using SQL only. However, we may expect that in real-

world accounting rules could be much more complex. For instance, we may add the rule “If Service Type is retired, then Pay Rate should be \$0”. It is easier to do it in rules:

DecisionTable DeterminePayRate				
Condition		Condition	Conclusion	ActionPrint
Errors		Service Type	Pay Rate	Result
Is Empty	TRUE	retired	\$0	Pay Rate = {{Pay Rate}}
Is Empty	TRUE		Base Rate + Rank Rate + Profession Rate + Service Type Rate + Unit Rate + Combat Rate	
Is Empty	FALSE		\$0	Please fix the errors of service data: {{Errors}}

Here we also added the column “ActionPrint” to print the resulting pay rate or all errors. Similarly, we may easily add more complex rules like “Give additional \$2 to active fighters who are marines participating in combat” but it could be not so simple to do it in SQL.