Transaction Validation Rules Solution

Exercise 1: Investigation

Create a transaction validation rule named "Observation Rule". Cut and paste the following code into the CONDITION. Then, answer the question below.

return transactionSet.Claim.LossType==LossType.TC\_PR

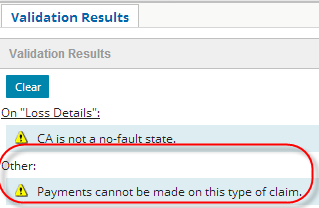
1. Under what circumstances will this rule execute? Write your explanation as if you were discussing the issue with a non-technical claims adjuster.

This rule executes whenever you create or change any type of transaction on a property claim.

2 Cut and paste the following incomplete line of code into the ACTION. Then, complete the code so that it displays a warning with the text message "Payments cannot be made on this type of claim." What is the complete line of code?

transactionSet.reject( null, null, "payment" , "Payments cannot be made on this type of claim." )

3. Reload changed classes in Studio (Run > Reload Changed Classes). Test your answer by logging on to ClaimCenter as Ronald Barnes (rbarnes) and creating a new reserve line on any claim he has access to.



4. The validation rule is a warning about payments. Why does it display when you create a reserve line?  
Can you add the reserve anyway? Go ahead and complete the reserve transaction.

The rule condition does not check to see if the transactions are payment transactions. The condition executes for all transactions. Therefore, the warning text does not match the rule logic.

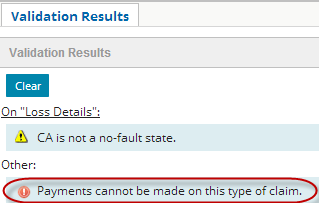
5. Now, modify the code in the Rule CONDITION so that the rule action executes only if the transactionSet contains payment transactions. Also modify the line of code in the Rule ACTION so that it displays an error with the same text message. What is the correct code?

CONDITION:

transactionSet.Claim.LossType == LossType.TC\_PR  
and transactionSet.Subtype =="CheckSet"  
// and transactionSet.AllTransactions.hasMatch(\ t -> t.Subtype =="Payment" )  
ACTION:

transactionSet.reject( "payment" , "Payments cannot be made on this type of claim.", null, null )

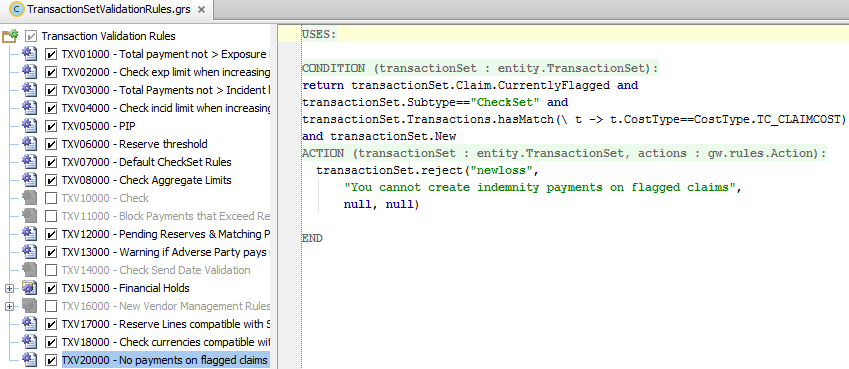
6. Reload changed classes. Test your answer by creating a new reserve line on a Ronald Barnes claim, and then by creating a payment against the reserve created above. What happens?  
  
The reserve is allowed, but the payment is not, and cannot be ignored.



**Exercise 2: Configuration**

Configure ClaimCenter to meet the following customer requirement from Acme Insurance.

**Requirement 1: Preventing New Indemnity Payments on Flagged Claims**



CONDITION:

return transactionSet.Claim.CurrentlyFlagged and

transactionSet.Subtype=="CheckSet" and

transactionSet.Transactions.hasMatch(\ t -> t.CostType==CostType.TC\_CLAIMCOST)

and transactionSet.New  
  
ACTION:

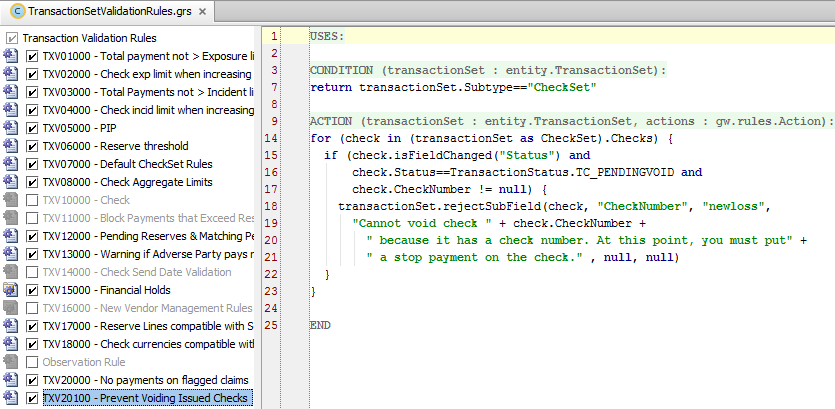
transactionSet.reject("newloss",

"You cannot create indemnity payments on flagged claims",

null, null)

**Requirement 2: Prevent the Voiding of Issued Checks**

Use rejectSubField() because Checks is a child array to the TransactionSet.



CONDITION:

return transactionSet.Subtype=="CheckSet"

ACTION:

for (check in (transactionSet as CheckSet).Checks) {

if (check.isFieldChanged("Status") and

check.Status==TransactionStatus.TC\_PENDINGVOID and

check.CheckNumber != null) {

transactionSet.rejectSubField(check, "CheckNumber", "newloss",

"Cannot void check " + check.CheckNumber +

" because it has a check number. At this point, you must put" +

" a stop payment on the check." , null, null)

}

}