保誠人壽 InW

- 建檔子流程設計說明

Snowfish Liu

Apr. 2017

建檔子流程定義

Name	Value
ID	newcontract.createnbprocess
Imports	com.pca.bpms.model.ApplicationForm, com.pca.bpms.model.newbusiness.DataEntry, com.pca.bpms.model.IssueCase, org.slf4j.Logger, org.slf4j.LoggerFactory
Package	com.pca.bpms.newbusiness
Process Description	建檔子流程
Process Name	CreateNBProcess
Version	1.0

流程變數

Name	Define Type	Desciption
appform	com.pca.bpms.model.Application	流程共用 Datamodel
dataEntry	com.pca.bpms.model.newbusiness.DataEntry	建檔 Datamodel
issue	com.pca.bpms.model.IssueCase	問題件 Datamodel
procLogger	org.slf4j.Logger	流程所使用的 Logger
isException	Boolean	是否發生異常
restUrl	String	rest url
result	Object	外部呼叫回傳結果
exceptionType	String	異常種類
sender	String	建檔人員
createGroup	String	新契約建檔群組
issueGroup	String	問題件處理群組
isReturned	Boolean	是否屬於案件返回
reviewGroup	String	建檔覆核人員

流程變數

Name	Define Type	Desciption
rootProcessId	String	記錄主流程 Process ID
parentProcessId	String	記錄母流程 Process ID (若為第 2 層子流程，與 rootProcessId 相同)
procinsIDS	String	主母子流程歷程 process ins ID
subLevel	Integer	子流程層數
subject	String	Email 主旨
from	String	寄信者
to	String	收信者
body	String	Email 內容
inquiryLIAIds	java.util.List	記錄身份證號清單 (for 公會索引)
reviewer	String	記錄覆核人員
nbEntryHandler	String	記錄建檔問題件處理人員
nbColHandler	String	記錄報繳問題件處理人員

流程變數

Name	Define Type	Desciption
channel	String	通路別 (from 主流程)
sales	String	銷售單位代碼 (from 主流程)
payment	String	首期繳費方式 銷售單位代碼 (from 主流程)
paid	String	是否繳款 (from 主流程)
authorized	String	是否已有授權碼 (from 主流程)
divided	Boolean	是否授權分案

流程 Datamodel- ApplicationForm(共用)

Identifier	Label	Type	Note
id	TaskId	long	
processInstanceId	流程實例ID	Long	
area	區域	String	
caseLevel	案件核保等級	int	
caseAnnualPremium	案件年化保費	BigDecimal	
caseFaceAmount	案件保額	BigDecimal	
caseType	進件型態	String	ex. 紙本, e-submission...
salesNo	銷售單位代碼	String	ex. 09AM1(渣打) 09AJ1(玉山)
planCode	險種名稱	String	
payment	首期繳費方式代碼	String	ex. 轉帳, 信用卡, 現金...
priority	重要性	int	0, 1, 2 (0優先)
processName	流程名	String	
processDesc	流程中文名	String	
status	狀態	String	(控制流程預留用)
triggerTime	process啟動時間	Date	(先保留)
packageNum	包裹號碼	String	

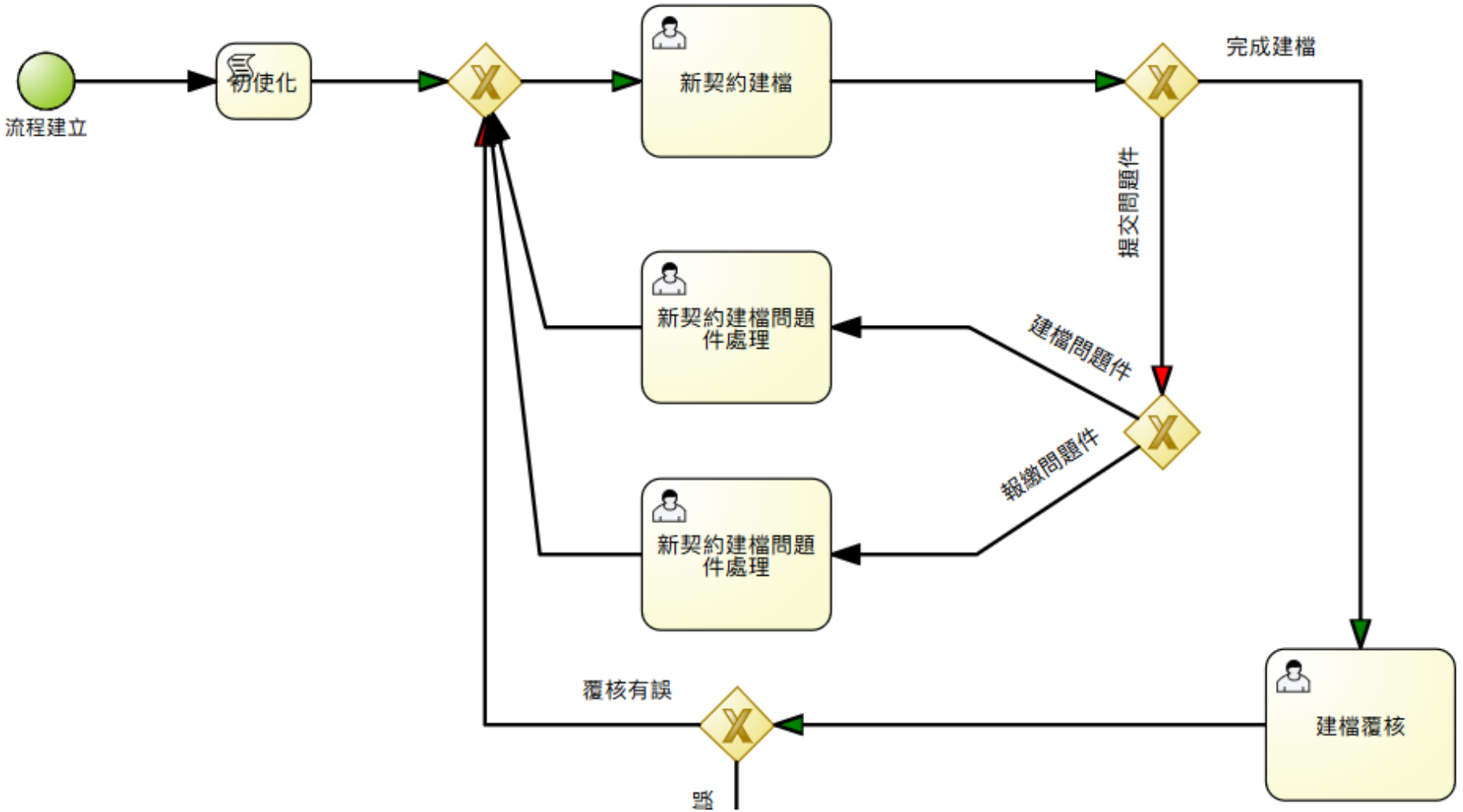
流程 Datamodel- IssueCase(問題件)

Identifier	Label	Type	Note
issueType	問題件型態	String	
sender	問題件提交人員	String	
handler	前一次的問題件處理人員	String	(預計移除)

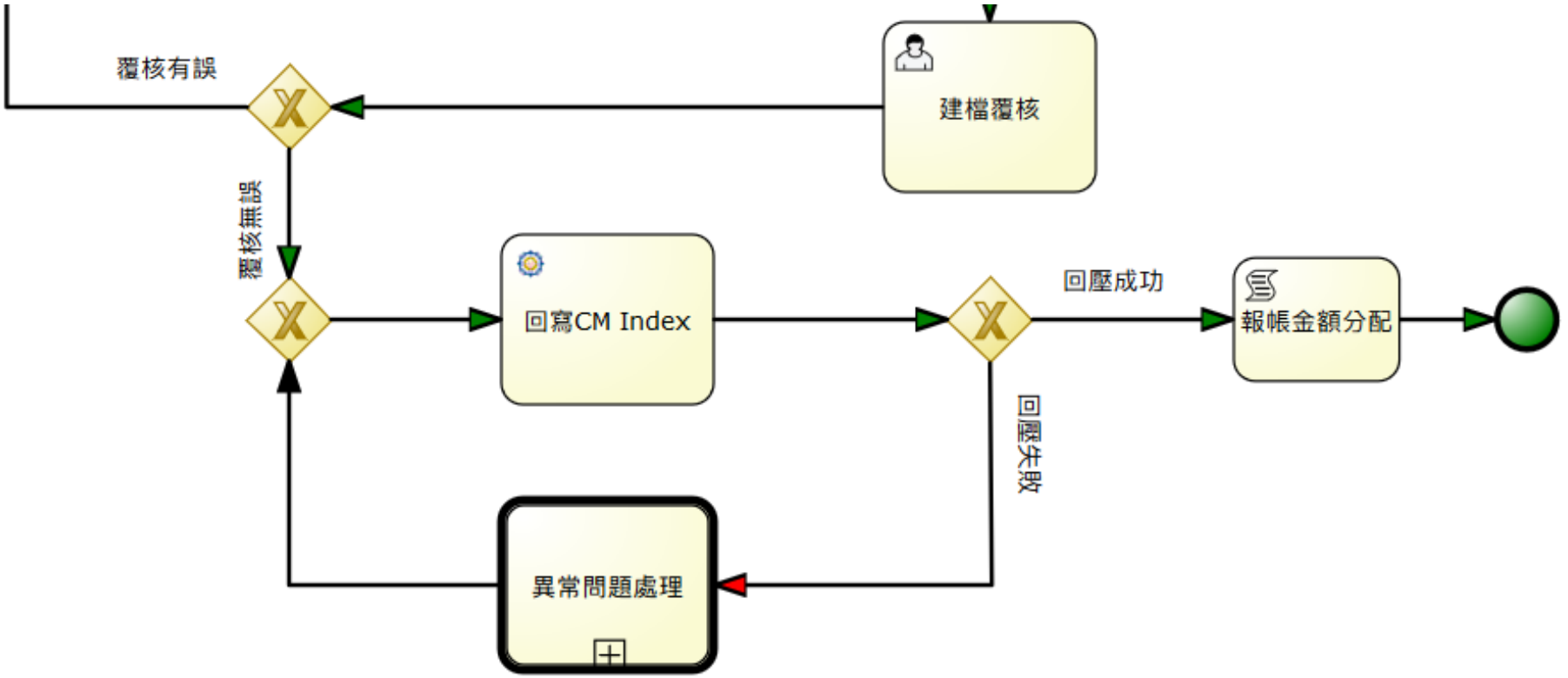
流程 Datamodel- DataEntry(建檔)

Identifier	Label	Type	Note
issueRaised	是否問題件	Boolean	
reviewed	覆核是否OK	Boolean	
~~divided~~	~~是否分案~~	~~Boolean~~	
creator	建檔人員	String	
reviewer	覆核人員	String	

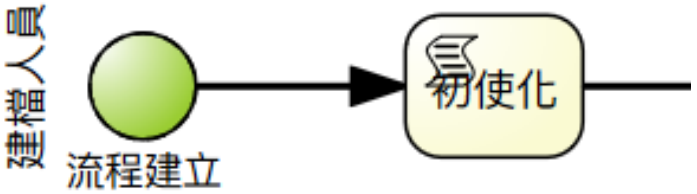
建檔子流程 BPMN 2



建檔子流程 BPMN 2



初使化作業



- 流程啟動後，第一個會執行的 task，使用 Script Task
- 初使化
  - 流程 Logger : procLogger
  - 初使化 Datamodel : appform, dataentry, issue
  - 初使化子流程層數 : subLevel
  - 紀錄 process instance id 歷程 : procinsIDS
  - 紀錄 rootProcessId, parentProcessId
  - 初使化 inquiryLIAIds

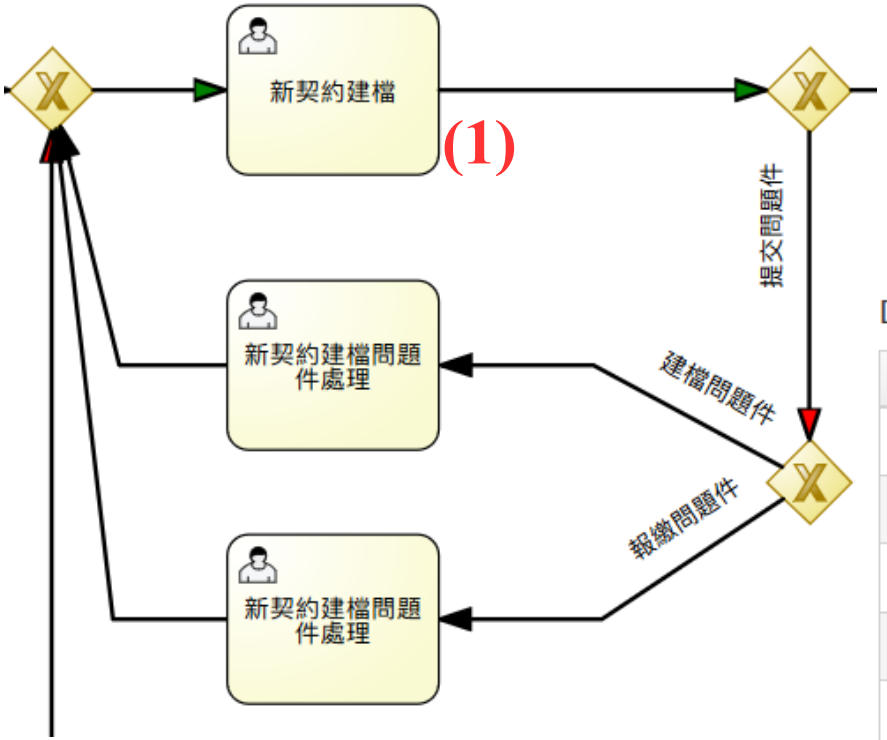
初使化作業

```
procLogger = (Logger) LoggerFactory.getLogger("Script Task Logger");
kcontext.setVariable("procLogger",procLogger);
procLogger.info("STAGE: "+kcontext.getNodeInstance().getNodeName());
appform = (ApplicationForm) kcontext.getVariable("appform");
if (null == appform) {  appform = new ApplicationForm();kcontext.setVariable("appform",
appform);}
dataEntry = new DataEntry();
dataEntry.setIssueRaised(false);
dataEntry.setReviewed(false);
dataEntry.setDivided(false);
kcontext.setVariable("dataEntry", dataEntry);
issue = new IssueCase();kcontext.setVariable("issue", issue);
```

初使化作業

```
subLevel = (Integer) kcontext.getVariable("subLevel");
subLevel +=1;
procinsIDS = (String) kcontext.getVariable("procinsIDS") + "," +
kcontext.getProcessInstance().getId();
rootProcessId = (String) kcontext.getVariable("rootProcessId");
parentProcessId = rootProcessId;
if (subLevel > 1) parentProcessId = kcontext.getProcessInstance().getProcessId();
kcontext.setVariable("subLevel", subLevel);
kcontext.setVariable("parentProcessId", parentProcessId);
kcontext.setVariable("procinsIDS", procinsIDS);
kcontext.setVariable("isException", false);
inquiryLIAIds = (java.util.ArrayList<String>) kcontext.getVariable("inquiryLIAIds");
if (null == inquiryLIAIds) inquiryLIAIds = new java.util.ArrayList<String>();
kcontext.setVariable("inquiryLIAIds", inquiryLIAIds);
```

1) 新契約建檔



➢ ***To-be***

執行建檔輸入的角色為”**新契約建檔人員**” **(createNB)**，在此人工作業可進行”提交問題件”或”建檔完成”

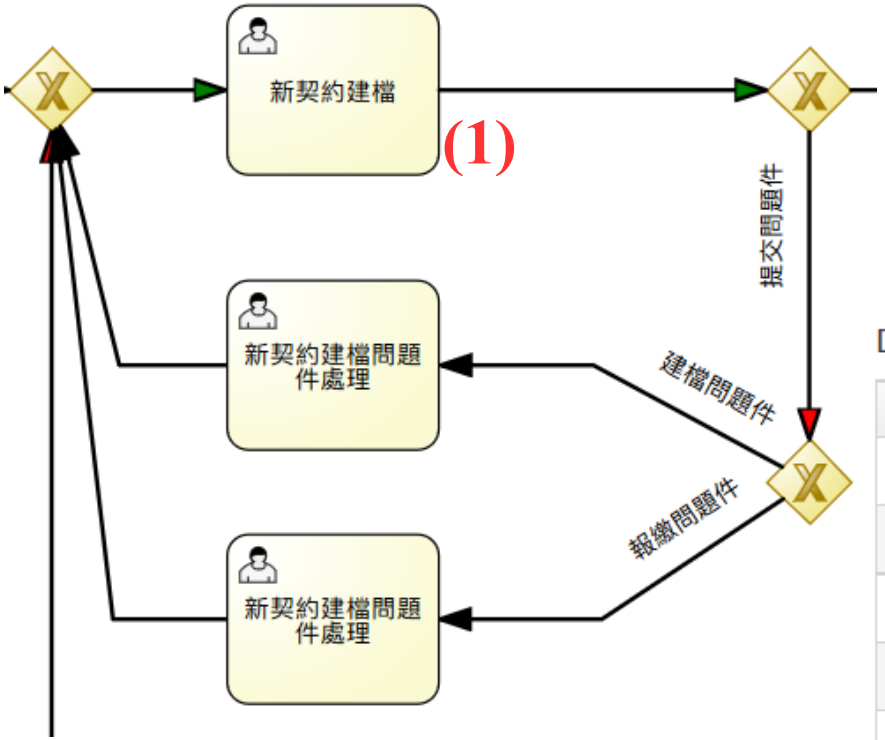
Data Inputs and Assignments

+ Add

Name	Data Type	Source
appform	ApplicationForm	appform
issue	IssueCase [com.ı	issue
rootProcessId	String	rootProcessId
parentProcessId	String	parentProcessId
procinsIDS	String	procinsIDS
subLevel	Integer	subLevel
dataEntry	DataEntry [com.ı	dataEntry
returnReason	String	returnReason

Name	新契約建檔
Actors	#{sender}
Groups	#{createGroup}
Task Type	User Task
Task Name	createNBEntry

1) 新契約建檔



Name	新契約建檔
Actors	#{sender}
Groups	#{createGroup}
Task Type	User Task
Task Name	createNBEntry

➢ ***To-be***

執行建檔輸入的角色為”**新契約建檔人員”** **(createNB)**，在此人工作業可進行

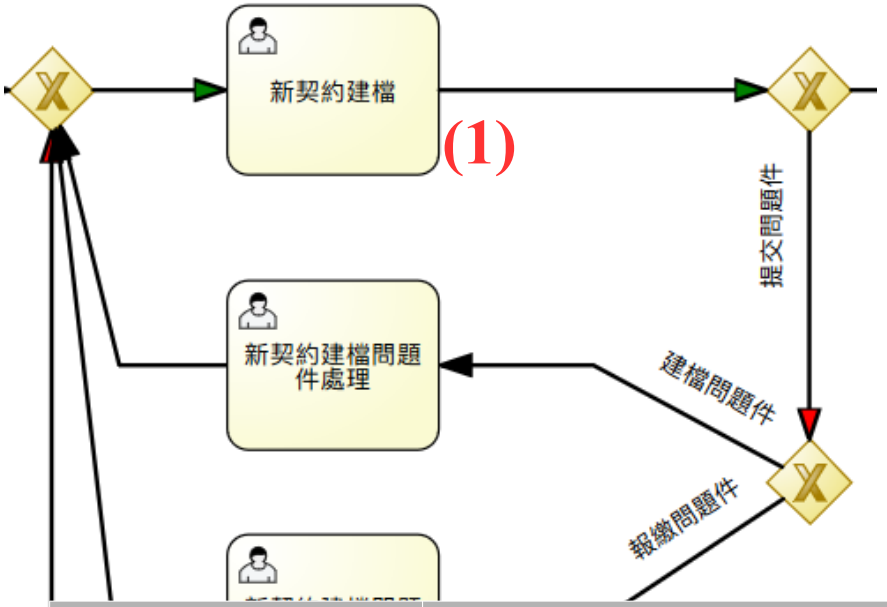
1. 提交問題件 **dataEntry.setIssueRaised(true)**
2. 判斷是否分案 divided
3. 完成建檔

Data Outputs and Assignments

+ Add

Name	Data Type	Target	
appform	ApplicationForm	appform	
issue	IssueCase [com.j	issue	
dataentry	DataEntry [com.j	dataentry	
channel	String	channel	
sales	String	sales	
payment	String	payment	
authorized	String	authorized	
paid	String	paid	

1) 新契約建檔



➢ ***To-be***

執行建檔輸入的角色為”**新契約建檔人員**” **(createNB)**，在此人工作業可進行” 提交問題件” 或” 建檔完成”

Name	新契約建檔
Actors	#{sender}
Groups	#{createGroup}
Task Type	User Task
Task Name	createNBEntry

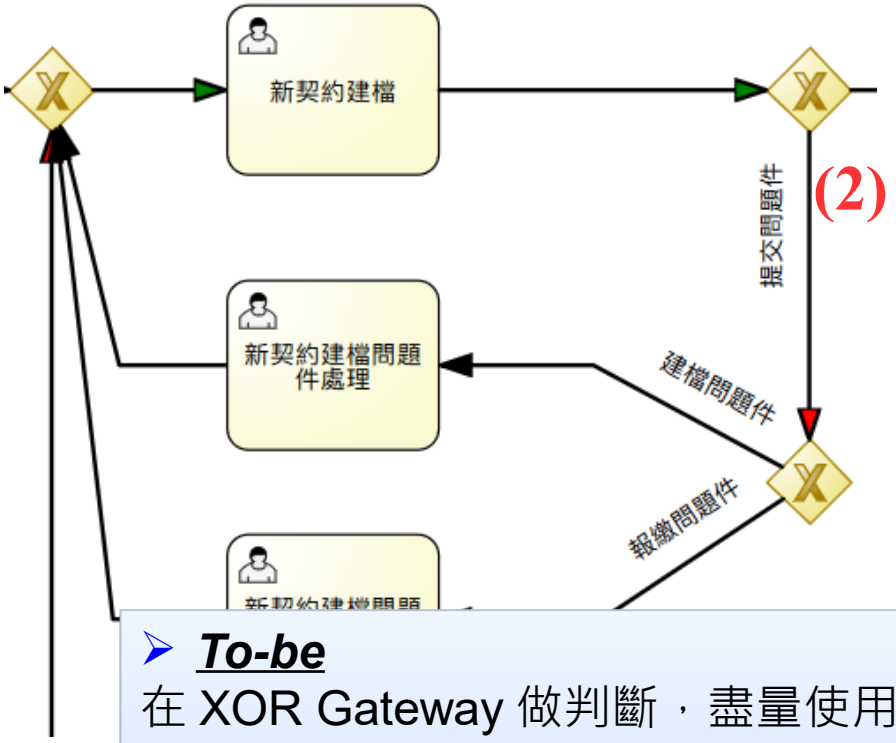
On Entry Action	if(KieFunctions.isEmpty(sender)) kcontext.setVariable("createGroup", "createNB");
On Exit Action	kcontext.setVariable("sender", dataEntry.getCreator());kcontext.setVariable("createGroup", null);

➢ ***To-be***

在進入建檔人工作業前先判斷 sender 是否存在，若為空則代表為第一次建檔或建檔人員不存在，則將 createGroup 填入 createNB

在離開建檔人工作業，將建檔人員填入 sender，然後將 createGroup 清空為 null

2) 提交問題件



Name	提交問題件
Type	Sequence Flow
Language	java
Expression	return KieFunctions.isTrue(dataEntry.getIssueRaised());

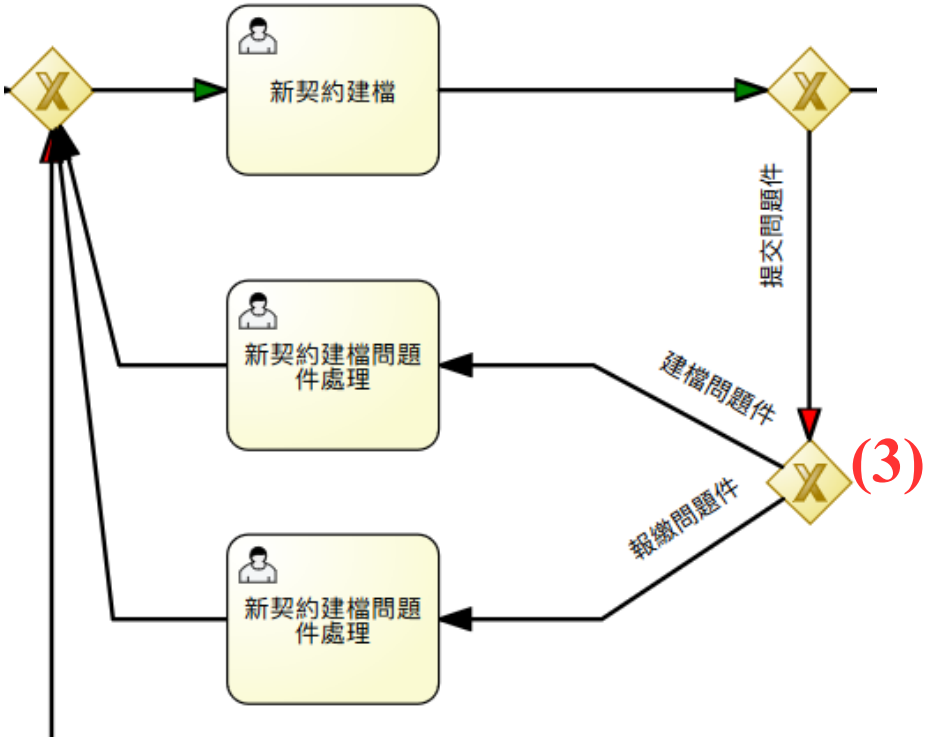
- ***To-be***

在 XOR Gateway 做判斷，盡量使用 Boolean 的型態做判斷

- ***To-be***

若在新契約建檔輸入人工作業，執行提交問題件，**會將 dataentry 的 issueRaised 變數設為 true** ，在 XOR Gateway 做判斷，進入問題件處理作業

3) 問題件類型判斷



Name	提交建檔問題件
Type	Sequence Flow
Language	java
Expression	return KieFunctions.equalsTo(issue.getIssueType(), "4");

Name	提交報繳問題件
Type	Sequence Flow
Language	java
Expression	return KieFunctions.equalsTo(issue.getIssueType(), "5");

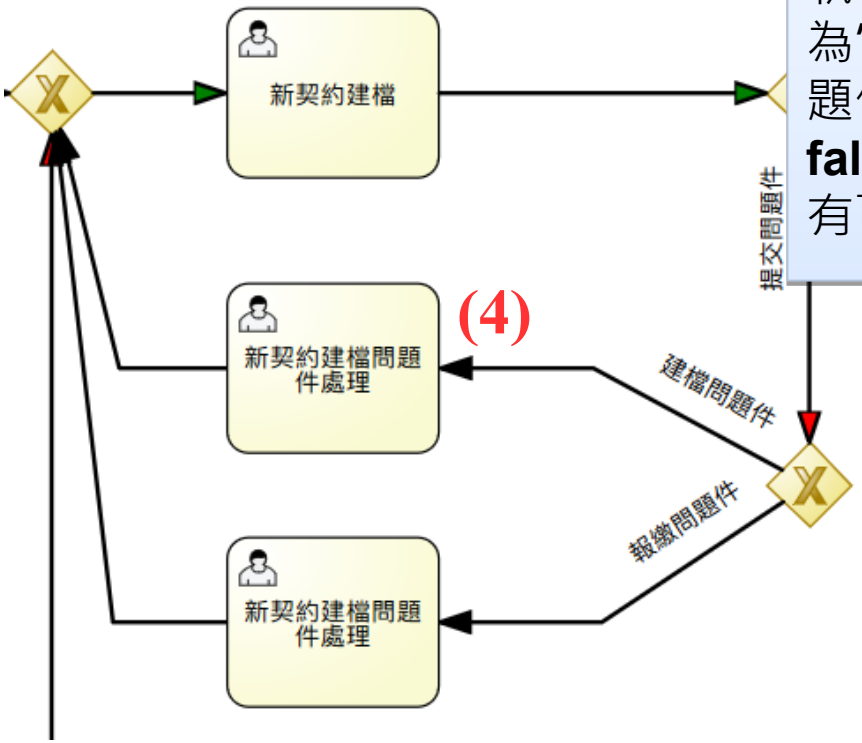
➢ ***To-be***

建檔問題件有兩種類型：

建檔問題件 issueType 為 4

報繳問題件 issueType 為 5

4) 建檔問題件作業



➢ ***To-be***

執行建檔問題件作業的角色為" **resolveIssueNBEntry"**，在此人工作業可進行" 問題件處理完成" ，**完成後流程須將 issueRaised 設為 false** ，回到新契約建檔輸入人工作業，再進行處理 (因有可能還會再提報問題件)

Data Inputs and Assignments

+ Add

Name	Data Type	Source
issue	IssueCase [com.]	issue
rootProcessId	String	rootProcessId
parentProcessId	String	parentProcessId
subLevel	Integer	subLevel
procinsIDS	String	procinsIDS
nbEntryHandler	String	nbEntryHandler
returnReason	String	returnReason

Data Outputs and Assignments

+ Add

Name	Data Type	Target
issue	IssueCase [com.]	issue
nbEntryHandler	String	nbEntryHandler
returnReason	String	returnReason

Name	建檔問題件作業
Actors	#{nbEntryHandler}
Groups	#{issueGroup}
Task Type	User Task
Task Name	createNBEntryIssue

4) 建檔問題件作業

On Entry Action	procLogger.info("On Entry Action, STAGE: "+
	kcontext.getNodeInstance().getNodeName());
	if(KieFunctions.isEmpty(nbEntryHandler))
	kcontext.setVariable("issueGroup", "resolveIssueNBEntry");

To-be

進入問題件前，先判斷 nbEntryHandler 是否為空，若為空，則代表為第一次問題件處理或問題件處理人員不存在，則須將 issueGroup 填件 resolveIssueNBEntry

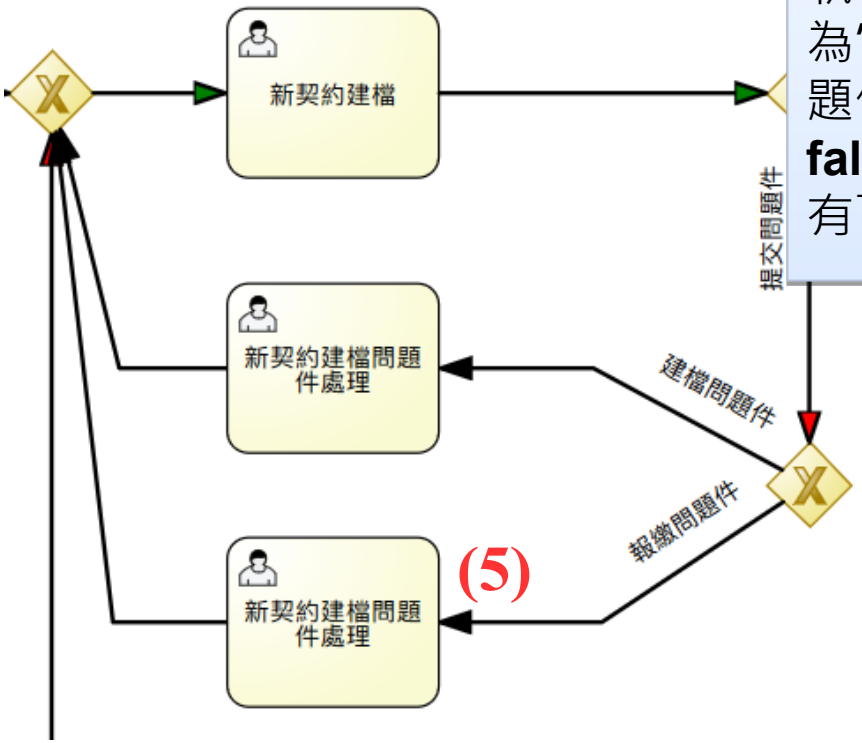
4) 建檔問題件作業

On Exit Action	procLogger.info("On Exit Actions - STAGE: "+kcontext.getNodeInstance().getNodeName());
	nbEntryHandler = (String) kcontext.getVariable("nbEntryHandler");
	if(!KieFunctions.isEmpty(nbEntryHandler))
	kcontext.setVariable("issueGroup", null);
	if(sender.equals(issue.getSender())) {
	  kcontext.setVariable("isReturned", true);
	} else {
	  kcontext.setVariable("sender", issue.getSender());
	  kcontext.setVariable("isReturned", false);
	}
	returnReason = (String) kcontext.getVariable("returnReason");
	dataEntry.setIssueRaised(false);

To-be

離開問題件，判斷 nbEntryHandler 是否有值，有值代表為問題件處理者已輸入，則須將 issueGroup 清空為 null, 因為可能此案件會再被提交二次問題件，處理人員都是同一個，並判斷返回新契約建檔的人員是否為同一名，將 isReturned 設為 true, 以及更新 returnReason, 並避免無指定建檔人員的情況，另外須將 issueRaised 設回 false

5) 報繳問題件作業



➢ ***To-be***

執行建檔問題件作業的角色為" **resolveIssueNBCol"**，在此人工作業可進行" 問題件處理完成" ，**完成後流程須將 issueRaised 設為 false** ，回到新契約建檔輸入人工作業，再進行處理 (因有可能還會再提報問題件)

Name	報繳問題件作業
Actors	#{nbColHandler}
Groups	#{issueGroup}
Task Type	User Task
Task Name	createNBColIssue

Data Inputs and Assignments

+ Add

Name	Data Type	Source
issue	IssueCase [com.j	issue
rootProcessId	String	rootProcessId
parentProcessId	String	parentProcessId
procinsIDS	String	procinsIDS
subLevel	Integer	subLevel
nbColHandler	String	nbColHandler
returnReason	String	returnReason

Data Outputs and Assignments

+ Add

Name	Data Type	Target
issue	IssueCase [com.j	issue
nbColHandler	String	nbColHandler
returnReason	String	returnReason

5) 報繳問題件作業

On Entry Action	procLogger.info("On Entry Action, STAGE: "+
	kcontext.getNodeInstance().getNodeName());
	if(KieFunctions.isEmpty(nbColHandler))
	kcontext.setVariable("issueGroup", "resolveIssueNBCol");

To-be

進入問題件前，先判斷 nbColHandler 是否為空，若為空，則代表為第一次問題件處理或問題件處理人員不存在，則須將 issueGroup 填件 resolveIssueNBCol

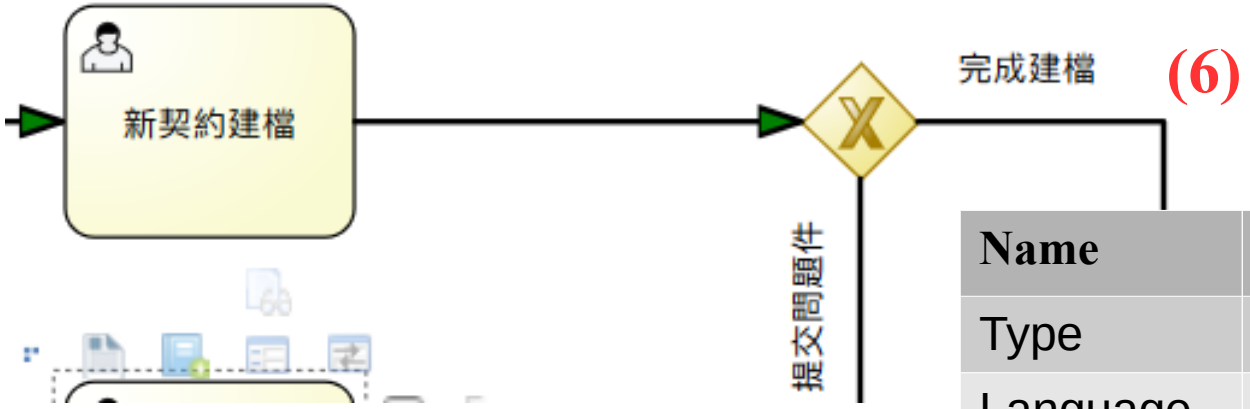
5) 報繳問題件作業

On Exit Action	procLogger.info("On Exit Actions - STAGE: "+kcontext.getNodeInstance().getNodeName());
	nbColHandler = (String) kcontext.getVariable("nbColHandler");
	if(!KieFunctions.isEmpty(nbColHandler))
	kcontext.setVariable("issueGroup", null);
	if(sender.equals(issue.getSender())) {
	  kcontext.setVariable("isReturned", true);
	} else {
	  kcontext.setVariable("sender", issue.getSender());
	  kcontext.setVariable("isReturned", false);
	}
	returnReason = (String) kcontext.getVariable("returnReason");
	dataEntry.setIssueRaised(false);

To-be

離開問題件，判斷 nbColHandler 是否有值，有值代表為問題件處理者已輸入，則須將 issueGroup 清空為 null, 因為可能此案件會再被提交二次問題件，處理人員都是同一個，並判斷返回新契約建檔的人員是否為同一名，將 isReturned 設為 true, 以及更新 returnReason, 並避免無指定建檔人員的情況，另外須將 issueRaised 設回 false

6) 建檔完成



Name	建檔完成
Type	Sequence Flow
Language	java
Expression	return KieFunctions.isFasle(dataEntry.getIssueRaised());

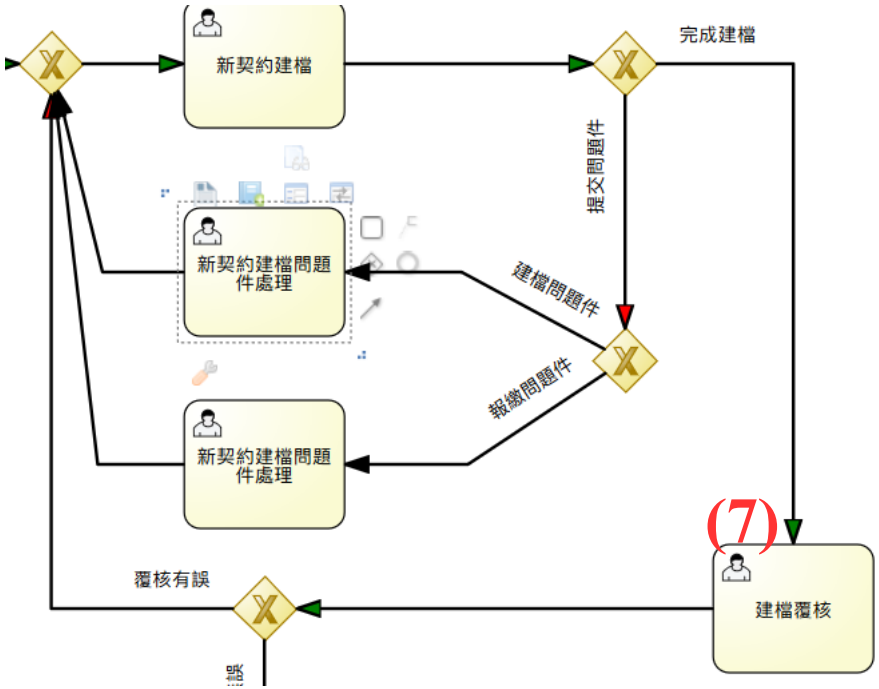
- ***To-be***

在 XOR Gateway 做判斷，盡量使用 Boolean 的型態做判斷

- ***To-be***

dataEntry 的 issueRaised 變數為 false ，在 XOR Gateway 做判斷，完成建檔作業

7) 建檔覆核



Name	建檔覆核
Actors	#{reviewer}
Groups	#{reviewGroup}
Task Type	User Task
Task Name	reviewNBEntry

Data Inputs and Assignments

+ Add

Name	Data Type	Source	
rootProcessId	String	rootProcessId	
parentProcessId	String	parentProcessId	
subLevel	Integer	subLevel	
procinsIDS	String	procinsIDS	
inquiryLIAIds	java.util.List	inquiryLIAIds	
dataentry	DataEntry [com.ı	dataentry	
appform	ApplicationForm	appform	

Data Outputs and Assignments

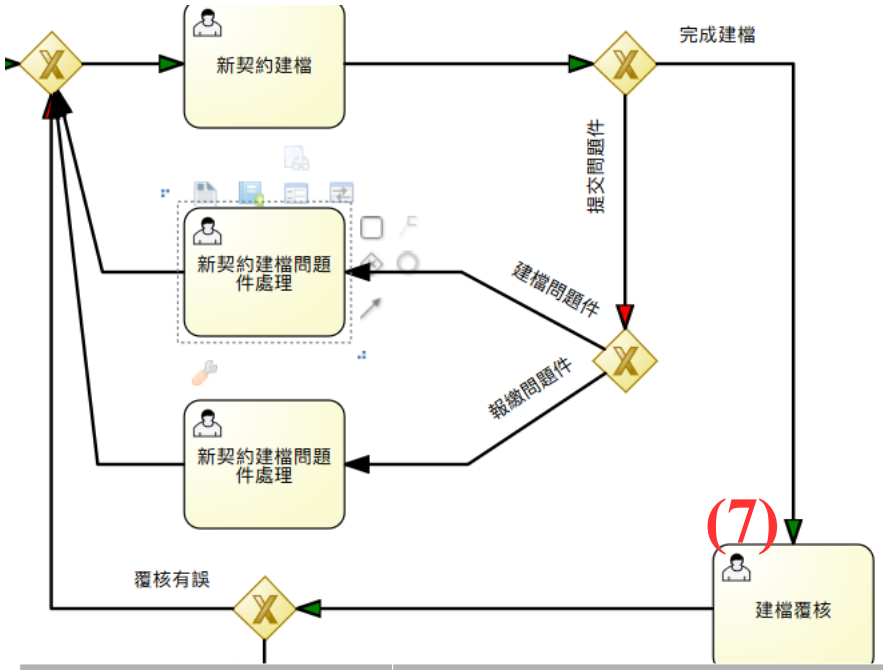
+ Add

Name	Data Type	Target	
inquiryLIAIds	java.util.List	inquiryLIAIds	
dataentry	DataEntry [com.ı	dataentry	
appform	ApplicationForm	appform	

➢ ***To-be***

執行建檔覆核的角色為”**建檔覆核人員”** **(reviewNB)**，在此人工作業由作業人員判斷建檔資料是否有誤，決定覆核結果，**此階段會產生身份證清單，將之存入 inquiryLIAIds 變數**

7) 建檔覆核



Name	建檔覆核
Actors	#{reviewer}
Groups	#{reviewGroup}
Task Type	User Task
Task Name	reviewNBEntry

➢ ***To-be***

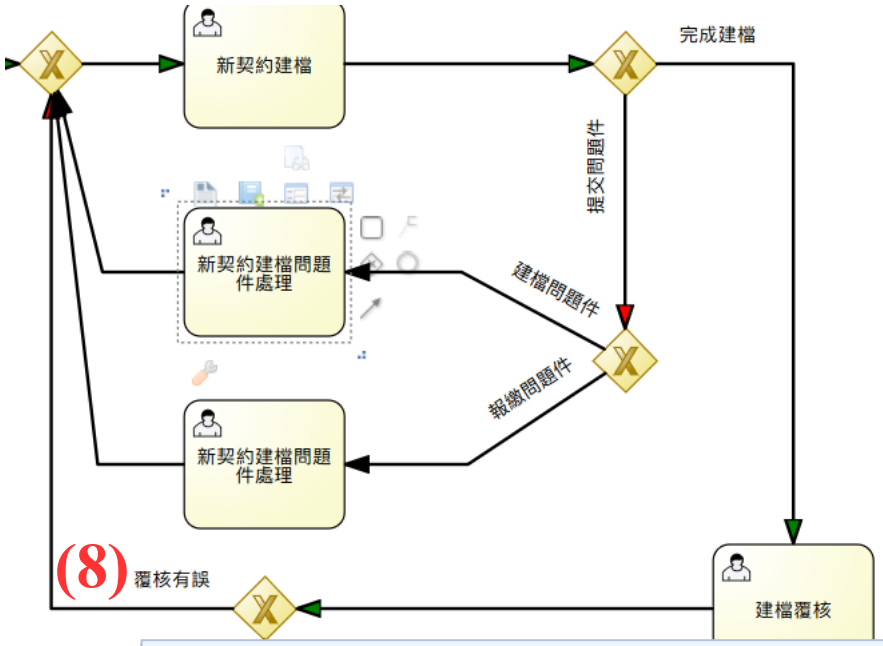
在進入建檔人工作業前先判斷 reviewer 是否存在，若為 null 則代表為第一次覆核或覆核人員不存在，則將 reviewGroup 填入 reviewNB

在離開建檔人工作業，將覆核人員填入 reviewer, 然後將 reviewGroup 清空為 null

On Entry Action	if(KieFunctions.isEmpty(reviewer)) kcontext.setVariable("reviewGroup", "reviewNB");

On Exit Action	if (!dataEntry.getReviewed()) {
	  if(dataEntry.getReviewer() != null) {
	  kcontext.setVariable("reviewGroup", null);
	  kcontext.setVariable("reviewer", dataEntry.getReviewer());
	  }
	  if(sender.equals(dataEntry.getCreator())) {
	  kcontext.setVariable("isReturned", true);
	  } else {
	  kcontext.setVariable("sender", dataEntry.getCreator());
	  kcontext.setVariable("isReturned", false);
	  }
	}

8) 建檔覆核有誤



Name	覆核有誤
Type	Sequence Flow
Language	java
Expression	return KieFunctions.isFalse(dataEntry.getReviewed());

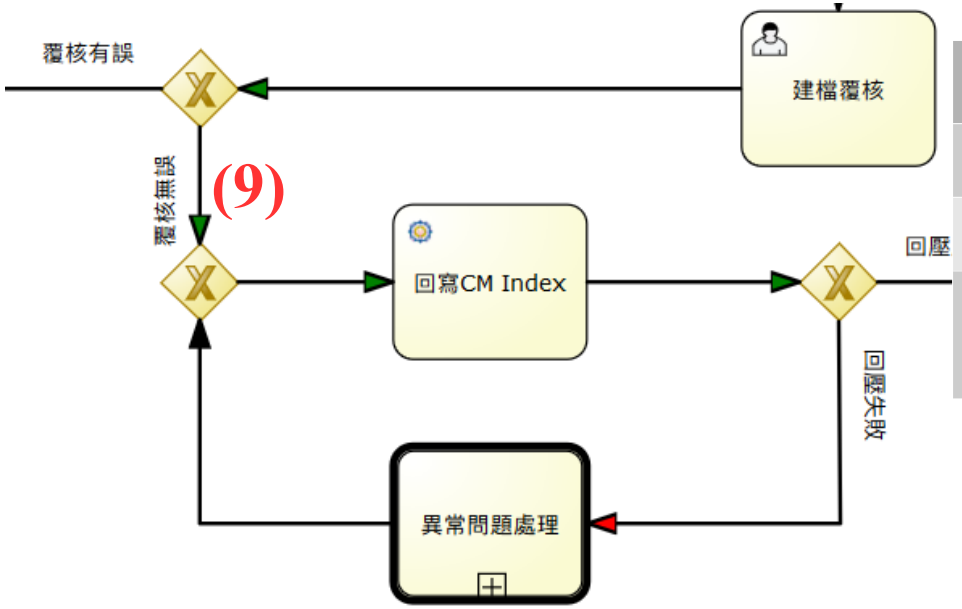
➢ ***To-be***

在 XOR Gateway 做判斷，盡量使用 Boolean 的型態做判斷

➢ ***To-be***

若在建檔覆核人工作業，判斷覆核結果有誤，**會將 dataEntry 的 reviewed 變數設為 false** ，在 XOR Gateway 做判斷，重新回到新契約建檔作業

9) 建檔覆核無誤



Name	覆核無誤
Type	Sequence Flow
Language	java
Expression	return KieFunctions.isTrue(dataEntry.getReviewed());

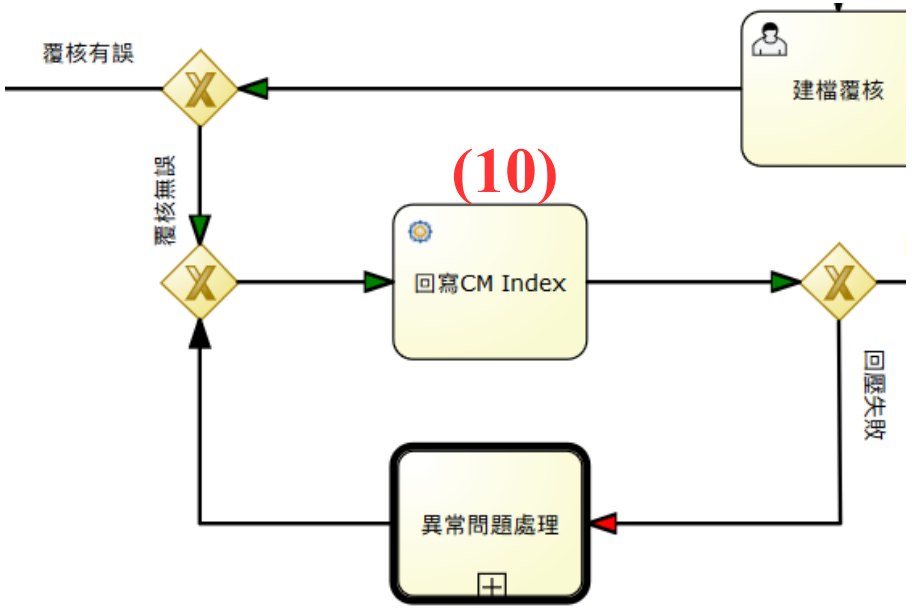
- ***To-be***

在 XOR Gateway 做判斷，盡量使用 Boolean 的型態做判斷

- ***To-be***

若在建檔覆核人工作業，判斷覆核結果無誤，**會將 dataEntry 的 reviewed 變數設為 true** ，在 XOR Gateway 做判斷，進入覆核完成後續作業

10) 回寫 CM Index



Data Inputs and Assignments

+ Add

Name	Data Type	Source
Url	String	restUrl
Method	String	"GET"
ConnectTimeout	String	
ReadTimeout	String	
Password	String	
Username	String	

Data Outputs and Assignments

+ Add

Name	Data Type	Target
Result	java.lang.Object	result

Name	回寫 CM Index
Type	Rest

➢ ***To-be***

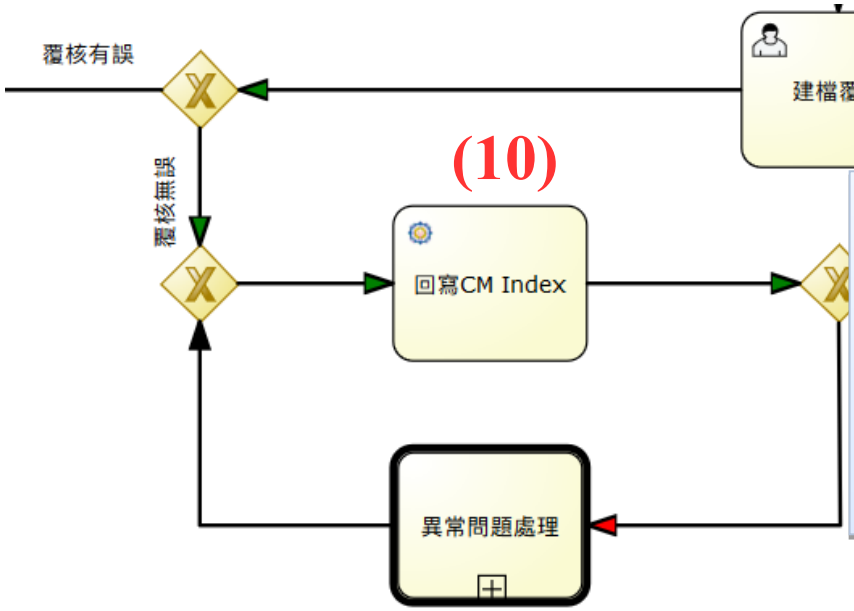
回壓文件類型的 CM index ，使用 Rest Task 來實現

➢ ***與通用性功能的關係或 Issue***

PCA 方提供回壓 CM Index 的 API, URL 範例 :

http://10.136.61.133:8080/InWCommon/rest/cm/udpateCreateNBCmIndex?pro_ins_id=123

10) 回寫 CM Index 結果處理

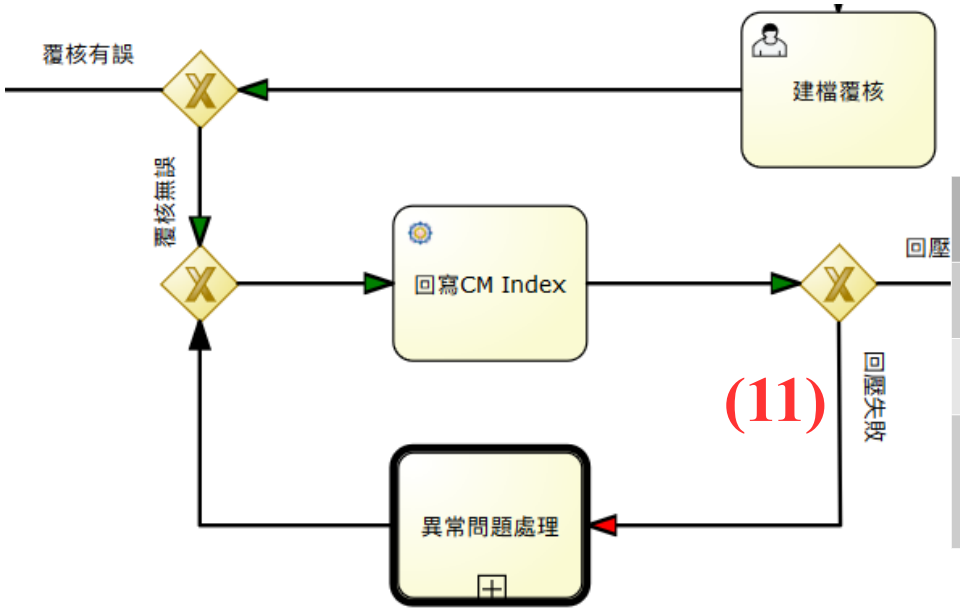


- ***To-be***

判斷呼叫回寫 CM Index 的結果，若成功 result 會回傳 true(字串)，失敗則將 isException 設為 true，並開始設定異常處理通知 email 的 subject 及 exceptionType, error stack 會寫入 result

On Exit Action	if (!String.valueOf(result).equals("true")) {
	kcontext.setVariable("result", result);
	kcontext.setVariable("isException", true);
	kcontext.setVariable("subject", " 回寫 CM index Error [" +
	kcontext.getProcessInstance().getId() + "] [" +
	kcontext.getProcessInstance().getProcessId() + "]");
	kcontext.setVariable("body", "ERROR Exception 內容 : " +
	result.toString());
	kcontext.setVariable("from", "pablo.sj.shen@pcalife.com.tw");
	kcontext.setVariable("to", "pablo.sj.shen@pcalife.com.tw");
	kcontext.setVariable("exceptionType", "NB");
	}

11) 回歷 CM Index 失敗



Name	CM Index 回歷失敗
Type	Sequence Flow
Language	java
Expression	return KieFunctions.isTrue(isException);

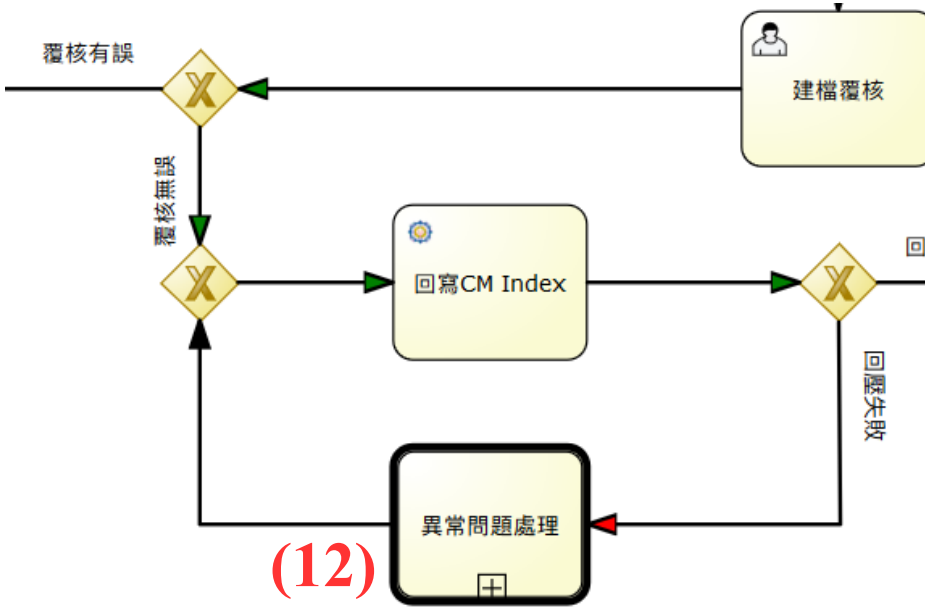
- ***To-be***

在 XOR Gateway 做判斷，盡量使用 Boolean 的型態做判斷

- ***To-be***

若回寫 CM Index 失敗，**會將 Process 的 isException 變數設為 true** ，在 XOR Gateway 做判斷，進入異常問題處理作業

12) 異常問題處理



Data Inputs and Assignments

+ Add

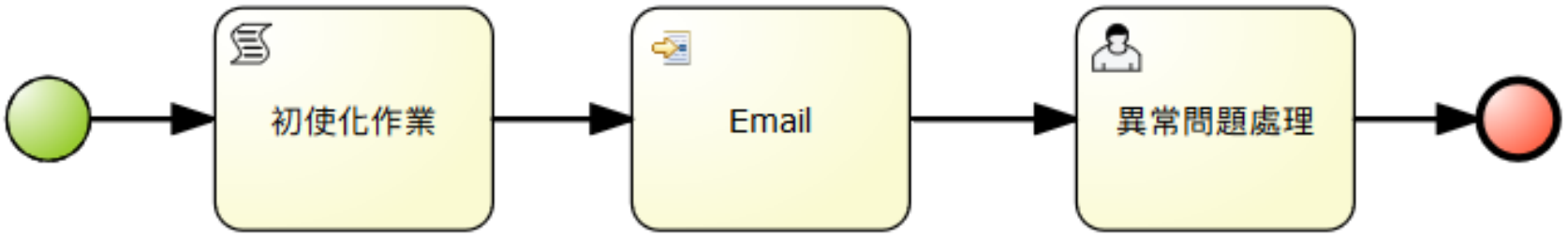
Name	Data Type	Source	
appform	ApplicationForm |	appform	
procLogger	org.slf4j.Logger	procLogger	
isException	Boolean	isException	
exceptionType	String	exceptionType	
result	Object	result	
rootProcessId	String	rootProcessId	
parentProcessId	String	parentProcessId	
subLevel	Integer	subLevel	
procinsIDS	String	procinsIDS	

Name	異常問題處理
Type	Sub-Process
Called Element	exceptionhandle

➢ ***To-be***

判斷回壓 CM Index 失敗後，進入異常問題處理，此處以起另一個子流程實作

異常處理子流程



- ***To-be***

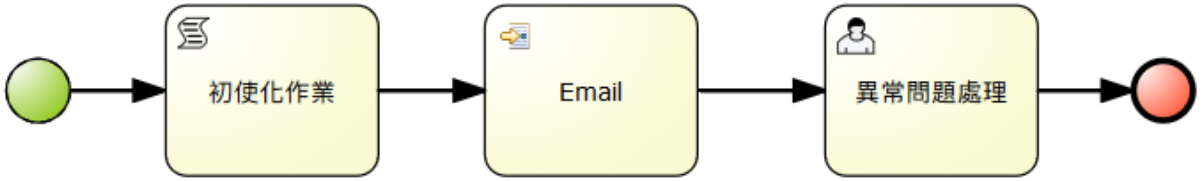
取得來自母流程的 Datamodel: ApplicationForm 及變數 result & exceptionType
設定異常問題處理的 Group 為 resolveExceptionNB
(Group 以變數 assignee 設計，保留未來異常問題處理子流程的彈性)

異常處理子流程

Name	初使化作業
Task Type	Script Task

Script	subject = (String) kcontext.getVariable("subject");
	body = (String) kcontext.getVariable("body");
	from = (String) kcontext.getVariable("from");
	to = (String) kcontext.getVariable("to");
	appform = (ApplicationForm) kcontext.getVariable("appform");
	result = kcontext.getVariable("result");
	exceptionType = (String) kcontext.getVariable("exceptionType");
	kcontext.setVariable("assignee", "resolveExceptionHC");
	subLevel = (Integer) kcontext.getVariable("subLevel");
	subLevel += 1;
	rootProcessId = appform.getProcessName();
	parentProcessId = rootProcessId;
	if (subLevel > 1) parentProcessId = (String)
	kcontext.getVariable("parentProcessId");
	procinsIDS = (String) kcontext.getVariable("procinsIDS") + "," +
	kcontext.getProcessInstance().getId();
	kcontext.setVariable("rootProcessId", rootProcessId);
	kcontext.setVariable("parentProcessId", parentProcessId);
	kcontext.setVariable("subLevel", subLevel);
	kcontext.setVariable("procinsIDS", procinsIDS);

異常處理子流程



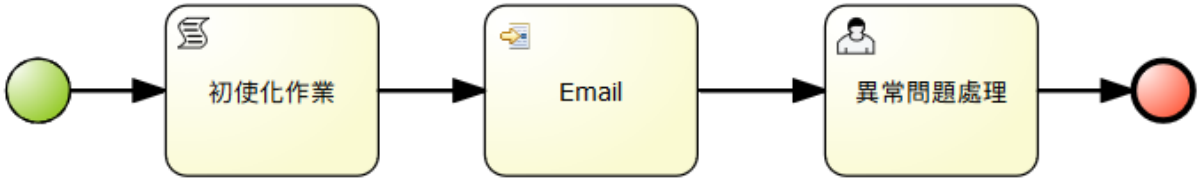
Name	Email
Task Type	Email Task

Name	Data Type	Source	
Subject	String	"call CM exceptio	
Body	String	"call CM exceptio	
From	String	"cloud_luster.tsa	
To	String	"cloud_luster.tsa	

- ***To-be***

發送 Email 通知處理人員

異常處理子流程



Name	異常問題處理
Task Type	User Task
Group	#{assignee}
Task Name	exceptionProc

Data Inputs and Assignments

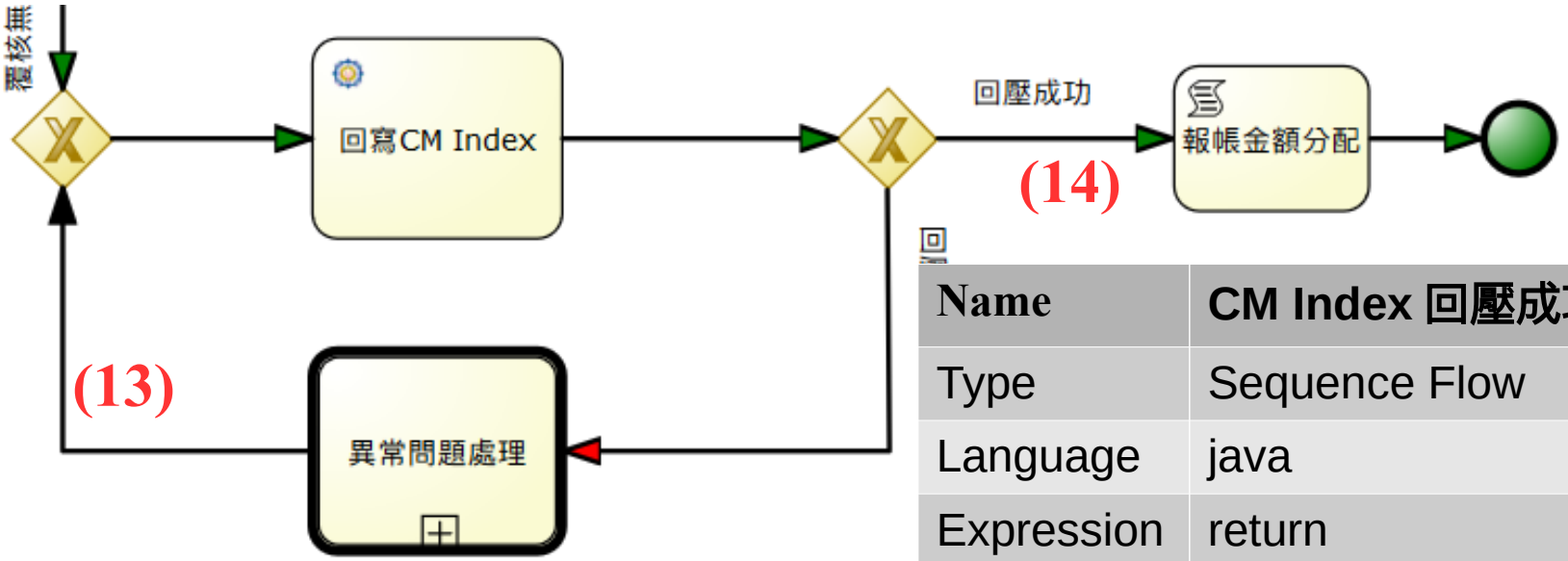
+ Add

Name	Data Type	Source
isException	Boolean	isException
appform	ApplicationForm |	appform
result	Object	result
exceptionType	String	exceptionType
rootProcessId	String	rootProcessId
parentProcessId	String	parentProcessId
procinsIDS	String	procinsIDS
subLevel	Integer	subLevel

➢ ***To-be***

異常問題處理人工作業，等待異常處理完成後送出結束異常處理作業

13) 結束異常問題處理 & 14) 回壓成功



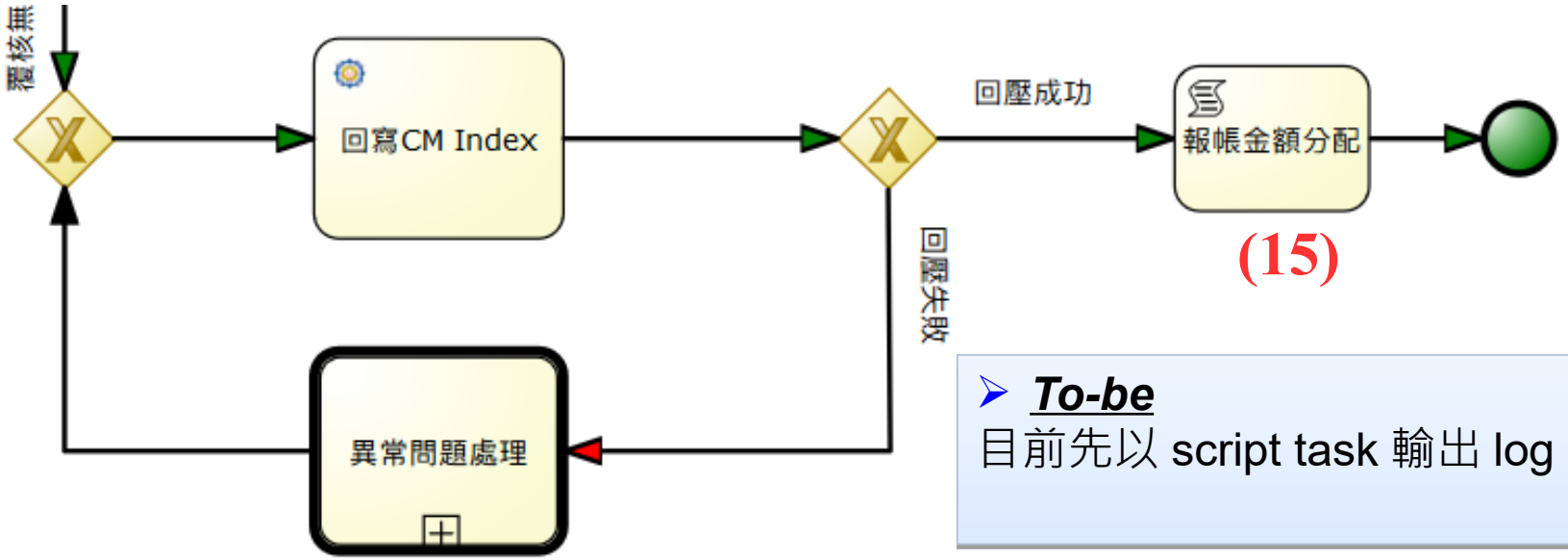
Name	CM Index 回壓成功
Type	Sequence Flow
Language	java
Expression	return KieFunctions.isFasle(isException);

Name	異常問題處理
Type	Sub-Process
On Exit Action	isException = false;

➢ ***To-be***
判斷 isException 設回 false，完成回壓作業

➢ ***To-be***
結束異常處理作業後將 isException 設回 false

15) 報帳金額分配 & 結束流程



➢ ***To-be***

目前先以 script task 輸出 log 處理

➢ ***與通用性功能的關係或 Issue***

1. **前端計算處理 or 呼叫 Rest ？**
2. **需要有異常處理 or 問題件 ?**

Name	報帳金額分配
Type	Script
Language	java
Script	logger.info("STAGE: "+kcontext.getNodeInstance().getNodeName());


Questions?