# Overview

The document outlines the list of actions taken in the context of **high-severity** items highlighted in the R3 Report. Each item has been carefully considered during the process, with a strong focus on the recommendations shared by R3 as well as the system requirements governed by business.

# Activity Summary: Report Findings (High-Severity)

1. Command Constraints

* The constraints set associated with the mentioned commands are updated to reflect those highlighted validations, which would be aligned with the identified business requirements.
* While the constraints are added based on the points shared in the report, the values used for validations might differ based on the system requirements. For instance, the status associated with a freshly created service request would be “New” in the system, instead of the value “CREATED” as suggested in the report.
* The changes are available in the *ServiceRequestContract.kt* file.

1. Notary Selection

* A new utility file *(\workflows\src\main\kotlin\io\agriledger\flows\utils\NotarySelector.kt)* is introduced to perform the get-notary task.
* The suggested implementation approach is incorporated into the *getNotary* method present inside the *NotarySelector* class.
* The *getNotary* method is called from all the flows which are constructing transaction builder objects.
* The *deployNodes* task inside the *build.gradle (agriledger-v1)* file is updated to provide the config values for each *workflows* cordapp. The task, when executed, would generate the required config file for each node.
* The changes are available in the following files:
  + *NotarySelector.kt*
  + *build.gradle (agriledger-v1)*
  + *All the applicable flow files*

1. Conditional Flow Constraints

* A set of flows (modifying *ServiceRequestState*), which could use conditional flow constraints, has been identified.
* A method named *validateFlowConstraints* is introduced in each of the identified flow.
* The method is called by each identified flow before proceeding with the flow’s actual responsibility.
* The flow would proceed with its execution only if the flow constraints validation is successful.
* The changes can be viewed in *CreateServiceRequestFlow*, *AcceptServiceRequestFlow*, *FruitFlow* and other flows which modify *ServiceRequestState* before the batch creation stage.

1. CreateBatchFlow

* The *BatchCreation* command has been removed from the contract and a *BatchAssociation* command has been introduced.
* The new command incorporates the constraints governing the evolution of *ServiceRequestState* during the association of a batch with a service request.
* Appropriate constraints are added to both the commands (*CreateBatch* and *BatchAssociation*) to perform the necessary validations.
* The signatory-check constraint (present in both the commands) is updated accordingly to validate the signer properly.
* The changes can be viewed in the following files:
  + *CreateBatchFlow.kt*
  + *BatchContract.kt*
  + *ServiceRequestContract.kt*

1. NewSaleFlow

* The *SaleCreation* command has been removed from the contract and a *SaleAssociation* command has been introduced.
* The new command incorporates the constraints governing the evolution of *BatchState* during the association of a sale with a batch.
* Appropriate constraints are added to both the commands (*CreateSale* and *SaleAssociation*) to perform the necessary validations.
* The signatory-check constraint (present in both the commands) is updated accordingly to validate the signer properly.
* The changes can be viewed in the following files:
  + *NewSaleFlow.kt*
  + *BatchContract.kt*
  + *SaleContract.kt*

1. Role Based Flow Execution

* Since all the nodes are participants of every defined State in the system, each node gets to record all the changes associated with any transaction (which modifies one or more states) initiated by any defined flow. Consequently, the JAR file shared with each node contains all the flows defined in the system.
* RPC permissions are enabled to restrict the node privileges for starting any flow defined in the system.
* A flow constraint is introduced (inside each flow) to ensure that the node running a flow is the one which is expected to run it.
* Here are the impacted files/areas:
  + *build.gradle (agriledger-v1) (RPC Permissions)*
  + *All the applicable flows (flow constraint introduction)*