[Saga Usage Conventions](https://wiki.eisgroup.com/display/GRC/Saga+Usage+Conventions)

Saga usage has performance impact and this page describes Saga usage conventions that must be followed when developing in Genesis project. This page is focusing on remote Saga usage Cases. Local Saga may be considered.

## **General information**

* Command stream ensures durability, order, retry on failure.
* Command handler should be idempotent and should be able to execute same command multiple times and get to the same state.
* Command for same entity are ordered and executed in sequence therefore there is no performance gain (e.g.no parallel processing) to use remote Saga for single entity.
* Local Saga may provide fork join performance optimization however performance penalty of executing state transition check, validation and etc may exist. Therefore it is recommended to reuse business logic by reusing service beans but not chaining commands to Saga.

## **Do not use Saga**

| **Scenario** | **Description** | **Comments** |
| --- | --- | --- |
| Business transaction affects single entity | * Business logic handled by single command should not start Saga * Do not use Saga to chain existing commands for business logic reusability |  |
| Business scenario requires validation / check using other Service. Second service does not have logic which involves persistence. | * Validation / Check is mandatory before returning response * Use direct call if data which affects validation is changed |  |
| Centralized Service coordinates business logic execution cross multiple Services | * If business logic is orchestrated by centralize service then this service may coordinate invocations and failure without starting Saga * Other orchestration mechanism is already used | E.g. Purchase service |
| Entities from other domain can be updated eventually after business transaction execution | * Async behavior. It is not required to update other entity before returning response * One service reacts to other service business events and compensation is not required |  |
| Other Service/ System can be invoked eventually after business transaction execution | * Async behavior. It is not required to call other Service/ System before returning response * One service reacts to other service business events and compensation is not required |  |

## **Use Saga**

| **Scenario** | **Description** | **Comments** |
| --- | --- | --- |
| Business transaction affects multiple entities of the same domain. | * Local Saga and fork/join during single command execution is preferred. * Use Distributed Saga if Entities with different IDs should be updated |  |
| Entities from multiple domains must be updated in a single business transaction. | * Sync behavior. Multiple entities should be updated or reverted before returning response * Sophisticated compensation logic on Failure |  |