|  |  |
| --- | --- |
| Use-Case Description | Composability Manager---Delete Resource Edge |
| Actors | Composability Manager, Janusgraph database, Resource Control Operations, Resource Events Framework, Composition Policies, Authorization Block |
| Description | Delete an edge to the graph database with Sunfish details, as a graph Vertex |
| Input Data | ResourceID |
| Pre Conditions | Composability Manager is running, Janusgraph database is running, Sunfish service is running |
| Post Conditions | A resource edge is deleted from the graph database |
| Trigger | Call is made for a graph database path deletion |
| Normal Flow | * Gather resource name and input data details * g.E(MessageID).drop   gremlin> graph = TinkerFactory.createModern()  ==>tinkergraph[vertices:6 edges:6]  gremlin> g = graph.traversal()  ==>graphtraversalsource[tinkergraph[vertices:6 edges:6], standard]  gremlin> g.V(1).bothE()  ==>e[9][1-created->3]  ==>e[7][1-knows->2]  ==>e[8][1-knows->4]  For purpose of the example, let's say we want to drop edges between vertex 1 and vertex 2. We could find those with:  gremlin> g.V(1).bothE().where(otherV().hasId(2))  ==>e[7][1-knows->2]  and then remove it with:  gremlin> g.V(1).bothE().where(otherV().hasId(2)).drop()  gremlin> g.V(1).bothE()  ==>e[9][1-created->3]  ==>e[8][1-knows->4]  If you have the actual vertices, then you could just do:  gremlin> g.V(v1).bothE().where(otherV().is(v2)).drop()  gremlin> g.V(1).bothE()  ==>e[9][1-created->3]  ==>e[8][1-knows->4]   * return success |
| Alternate Flow 1 |  |
| Alternate Flow 2 |  |
|  |  |