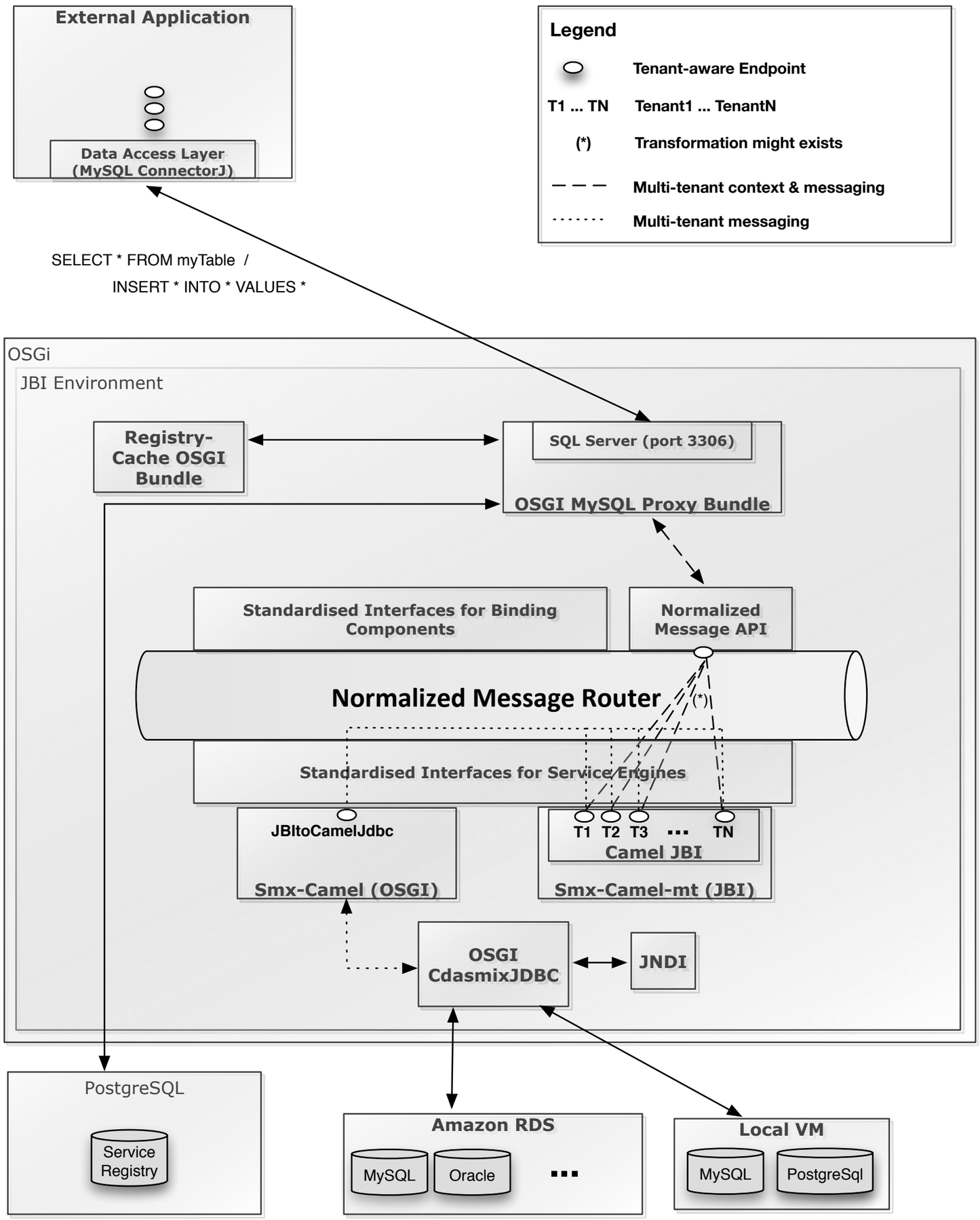

talk about why we were forced to use a single jbi endpoint for reception and delivery of all the messages. Because we are packing the multi tenant binding component as service assemblies, they get deployed into the jbi container and get modified in order to appear, and to be deployed also in the osgi container, but the zip of the service assembly is not osgi friendly, this means, the osgi deployer cant access the meta data information in order to register the bundle as a service in its registry. In JBI all the jars that a service unit may need to use need to be included as dependencies (and in the final zip file) in oder to access them. We could not access the osgi bundles from the multi tenant service assemblies, this means that if we strictly follow the jbi spec we would have service assemblies of a huge size, and each time we modify our cdasmix jdbc we would have to redeploy each of the tenant service assemblies. We cant add the dependencies in the binding component, but in the service unit which contains the route information, because each route has a differente context and needs the jar inside the service unit.

create a sql proxy server which interacts with the normalized message router and generates a normalized message to the tenant aware jbi endpoint which is deployed using camel. we send it to a jbi endpoint to make it compatible with what we have and to provide future compatibility by using camel, because jbi integration in servicemix is being depreceated from version 4.0.

(*) if the target datasource is a nosql datasource, both query and data must be transformed in the DA Transformer