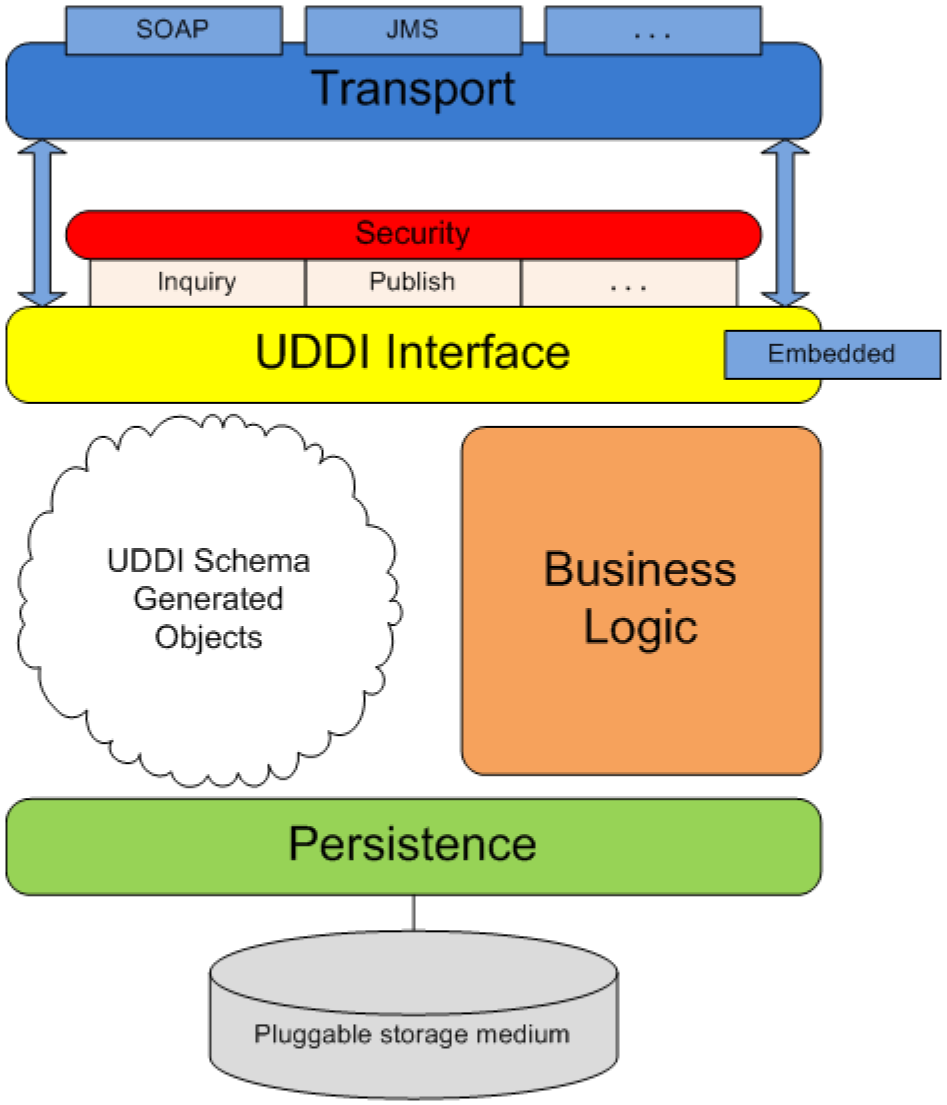
jUDDI Architecture for UDDI v3 Project



The diagram above shows the software layers and components employed in the jUDDI project implementation for UDDI v3. Here is a brief description of each item in the diagram and how they all work together to create the UDDI-compliant registry that is jUDDI:

Persistence - the persistence layer, as its name implies, is responsible for persisting registry data to a storage medium. To this end, a third-party persistence service that implements the Java Persistence API (Apache Hibernate) is utilized to manage transactions with the storage medium and also to facilitate the plugging-in of various storage types.

UDDI Schema Objects - The UDDI specification comes equipped with an XML schema for its many data structures. jUDDI employs XML-binding technology (JAXB) to generate objects from the schema that are then used as the interface to both the persistence layer and the UDDI interface layer. The logic layer works with these objects to perform the business logic associated with a UDDI call.

Business Logic - the business logic layer is responsible for performing all the business logic associated with the UDDI calls. UDDI-based requests received from the UDDI interface layer are injected into a schema object and then utilized by this layer to invoke the necessary logic. This usually entails interacting with multiple schema objects - taking advantage of the persistence services provided by these objects to retrieve and store data.

UDDI Interface - the UDDI interface layer accepts incoming UDDI-based requests and ummarshals these requests to the appropriate schema object. This object is then served to the logic layer so the necessary logic can be performed to fulfill the request. After the request is fulfilled, this layer is responsible for marshalling the result and sending the response to the requesting party.

It should be noted that jUDDI can be accessed via an "embedded" mode. In this scenario, an interface is provided to directly interact with objects, forgoing the need to marshal and unmarshall XML requests. This allows users to embed jUDDI directly into their application without having to deploy a full-blown jUDDI server.

Security - security is provided by the UDDI specification and is based on policies defined in the specification. jUDDI implements all the mandatory policies and can be extended to support the optional policies. Chief among these policies is access control to the UDDI API exposed by jUDDI. jUDDI fully implements this policy, per the specification, which allows users to easily plug in their own third-party authentication framework.

Transport - the transport layer provides the means to receive and send out requests via a network or messaging service. The UDDI specification details an interface where XML messages are exchanged between client and server but is agnostic as to how those messages are relayed. By default, jUDDI uses Apache Axis2 to transport messages via SOAP over HTTP, however, the system is designed so other methods of transport can be easily plugged in (for example, JMS).