### Service Statelessness

There are different levels of state management that can result in a significant resource burden when multiple services have access to different parts of an environment. State data management consumes system resources and can lead to the repeated ROI for the service. (Composition Member Design and Complex Compositions)

**Key Concepts**

- **Service Contract**: A service contract is the interface between service providers and service users, specifying the capabilities and constraints of a service.
- **Service Composition**: A service composition is a collection of services that work together to provide a larger, more complex service.
- **Service Statelessness**: A service is considered stateless if its statelessness is an inherent quality of its design. When a service is stateless, it can be treated as a unit of deployment.

**Implementation Considerations**

- **Service Statelessness**
  - Oracle WebLogic
  - IBM WebSphere
  - HP: Itanium server

**Service Statelessness in SOA**

- **Centralization**: The properties of SOA are always one to the cloud. SOA is a cloud to SOA.
  - Service families can be used to centralize the properties of services.
  - Service family can end up forming negative types of coupling upon the service logic may be implemented (and the service logic may be encapsulated.

**Service Statelessness in SOA**

- **Service Autonomy**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Service contracts only contain essential behavior.
  - Services are supplemented with communicative meta data.

**Service Statelessness in SOA**

- **Service Composability**
  - This principle is related to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Reusability**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Loose Coupling**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Abstraction**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Loose Coupling**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Autonomy**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Reusability**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Loose Coupling**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.

**Service Statelessness in SOA**

- **Service Abstraction**
  - This principle relates to the service contract. Based on its level of autonomy and the amount of state data.
  - Services are implemented with service-orientation design.
  - Services can be reused by service already provides.