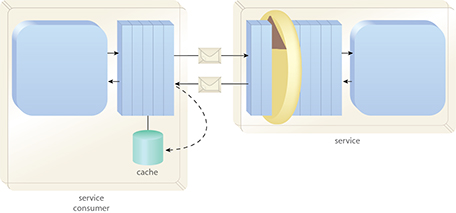
Six Constraints or REST

Client-Server

**Cacheable**: A service can define itself as catchable. Cache. Response messages from the service to its consumers are explicitly labeled as **cacheable** or non-**cacheable**. ... A common reason for incorporating caching as a native part of REST architecture is as a counterbalance to some of the negative impacts of applying the Stateless constraint.



*Figure 1 - Responses may be cached by the consumer to avoid resubmitting the same requests to the service. This diagram depicts a consumer-side cache. REST services can also use service-side caches for scalability purposes.*

**Stateless**: All the data needed for a request to the service method is contained in the request itself. The server does not store anything.

**Layered:** The client makes a request and it could come from a load balancer a CDN a cache server or the host web server etc. The client does not care as long as it gets its data. Where the data comes from in any given request is transparent to the client application.

**Code on demand: (Optional)** Service could send java script with the response to enhance the clients experience.

**Uniform Interface:**

* **Resource Identification** – A service resource can be identified via a URI. The data that is returned to the client is a representation of data. The client does not care where the data came from (Database, File System, CDN etc.) as long as the data is returned in the request.
* **Resource Manipulation with Representation** - The representation data is not the data itself it is just a representation. The representation of data has meaning and can be used to manipulate the resources on the server.
* **Self-Descriptive -** The service tells the client what kind of data it will send back (XML, JSON, CSS File, Html etc.) The point is that the service response is described to the client via Header information etc.
* **HATEOAS –** Hypermedia As The Engine Of Application State

REST != Good