**Best Practices in designing the Mulesoft Scenarios**

1)**Best practices while designing API:-**

* Understand why we build the API.
* Think long term for at least two to three years.
* Define functionality (Outside in + Inside out)
* Follow SPEC driven development
* Use authentication and throttling
* Use appropriate HTTP header status codes
* Use descriptive error messages
* Don’t break compatibility - version cautiously
* Use API proxy layer
* Prefer to use non-blocking architecture

2)**Mule Designtime Tuning:-**

* Plan small flows
* Maintain appropriate JVM Heap size, Garbage collection
* Use Conc Mark Sweep GarbageCollection
* Avoid using session variables.
* Java objects are faster in Payload formats compared to that of XML.
* VM transport is used to distribute load across nodes.
* Modularize (creating multiple configuration xml files bases on requirement)
* Environmentalism (Always use placeholders where ever we can, especially in case of outbound connectors database, http connectors)

**3)Infrastructure Tuning:-**

* The number of file descriptors should be more for high socket connections.
* If large machines (Cores or RAMs), consider more than one mule per machine.

4)**Data Extraction**:-

* Mule expression language is preferred to other scripting languages

5)**Integration patterns and Performance Implications**: -

* Scatter gatherer (results of operations are combined)
* Batch processing is used for bulk loads handling
* Cache scope reduces the data retrieval load on database.
* Use transaction course wherever necessary.

6)**Logging:-**

* Log the input and output messages appropriately.
* Log the error messages appropriately

7)**Throughput:-**

* In order to increase the throughput efficiency be sure to use persistent HTTP connections.
* Use the default Garbage collection rather than Concurrent Mark and Sweep.

8)**Hight Concurrency:-**

* It is important to choose the one that could support high concurrency.
* A non-thread based, non-blocking client such as API bench is recommended because it is lightweight.

9)**Reusability:-**

* Design the config XMLs in such a way that they can be reused in multiple scenarios.

10)**Exception Handling:-**

* Use appropriate messages for the errors such as validation errors, technical errors and business errors.
* Prefer to use global exception strategies