CORBA – Common Object Request Broker Architecture

CORBA (Common Object Request Broker Architecture) is a distributed object standard defined by the Object Management Group (OMG) in the early 1990s. Its goal was to enable software components written in different programming languages and running on different platforms to communicate seamlessly.

CORBA introduced the concept of an Object Request Broker (ORB), which acts as middleware between clients and distributed objects. In essence, CORBA extended the object-oriented programming paradigm into distributed systems by allowing so-called “distant objects” (remote objects) to interact as if they were local.

This was a major step in software integration before REST APIs and microservices became dominant.

Object Request Broker (ORB)

The **ORB** is the central component of CORBA. It is responsible for:

- locating remote objects,
- forwarding client requests,
- handling communication between distributed components,
- managing object references.

From the programmer’s perspective, calling a remote object method in CORBA looks very similar to calling a local method. The ORB hides the complexity of network communication.

Conceptually:

Client → ORB → Remote Object

Remote Object → ORB → Client

The ORB handles all low-level details such as transport protocols and data encoding.

Distributed Objects Concept

CORBA extends object-oriented principles to distributed environments.

Remote objects:

- expose methods,
- encapsulate data,
- can be invoked across networks.

Unlike REST (which is resource-based) or JSON-RPC (which is procedure-based), CORBA is object-based. Objects can share both state (data) and behavior (methods).

This design reflects classic OOP thinking in distributed systems.

4. IDL – Interface Definition Language

CORBA introduced a language-independent interface definition mechanism called IDL (Interface Definition Language).

IDL allows developers to define object interfaces independently of programming languages.

Example IDL:

```
interface Calculator {
   long add(in long a, in long b);
};
```

From this IDL definition, language-specific stubs and skeletons are generated automatically (e.g., for C++, Java, Python).

This ensures:

- language interoperability
- strict interface contracts
- platform independence

IDL was one of the early solutions to cross-language service integration.