Enterprise Application Integration (EAI)

Chapter 7. An Introduction to EAI and Middleware

“All programmers are playwrights and all computers are lousy actors.”

Introduction

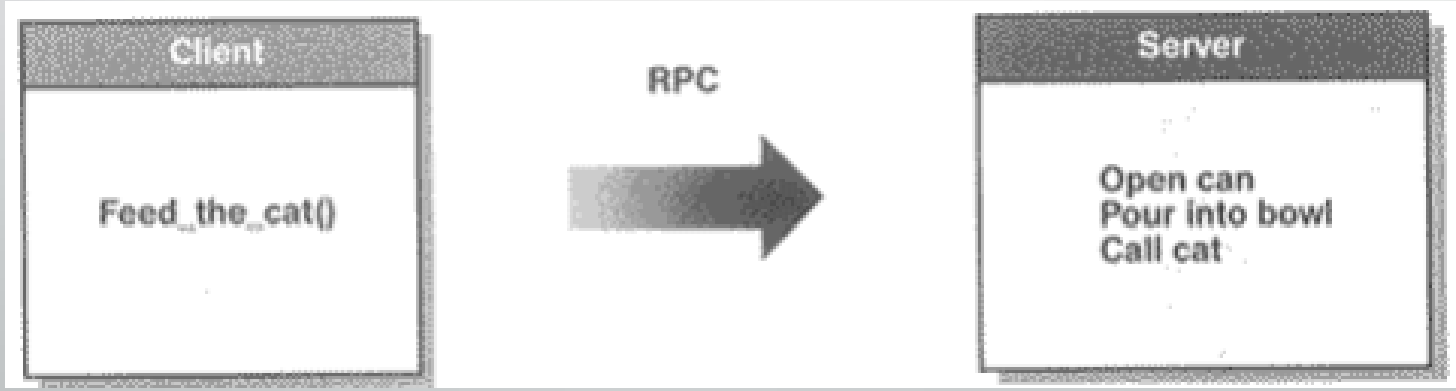
- The first six chapters have been devoted to EAI approaches and implementation.
- In the following chapters, we will concentrate on the technology that makes EAI possible: **middleware**.
- The next several chapters will describe several types of middleware technologies that may assist us in solving the EAI problem.

Middleware: The Engine of EAI

- A mechanism that allows one entity (application or database) to communicate with another entity or entities
- Middleware is basically any type of software that facilitates communications between two or more software systems

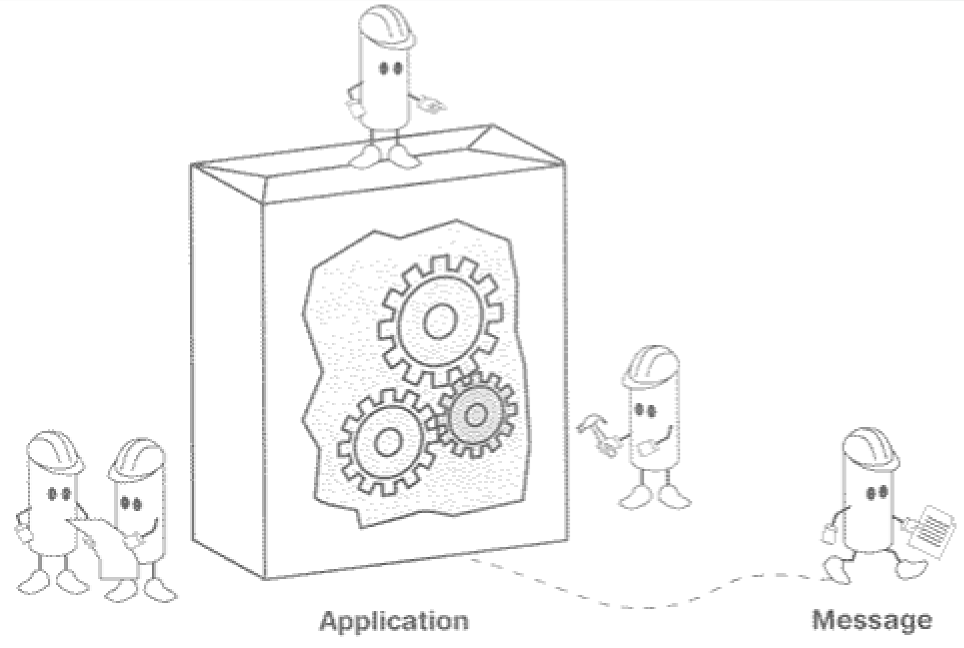
Types of Middleware

- **RPC**s are the oldest type of middleware.



Types of Middleware (cont.)

- **Message-Oriented**: MOM was created to address some shortcomings of RPCs through the use of messaging
- The asynchronous paradigm is much more convenient for developers and users because it does not block the application from processing

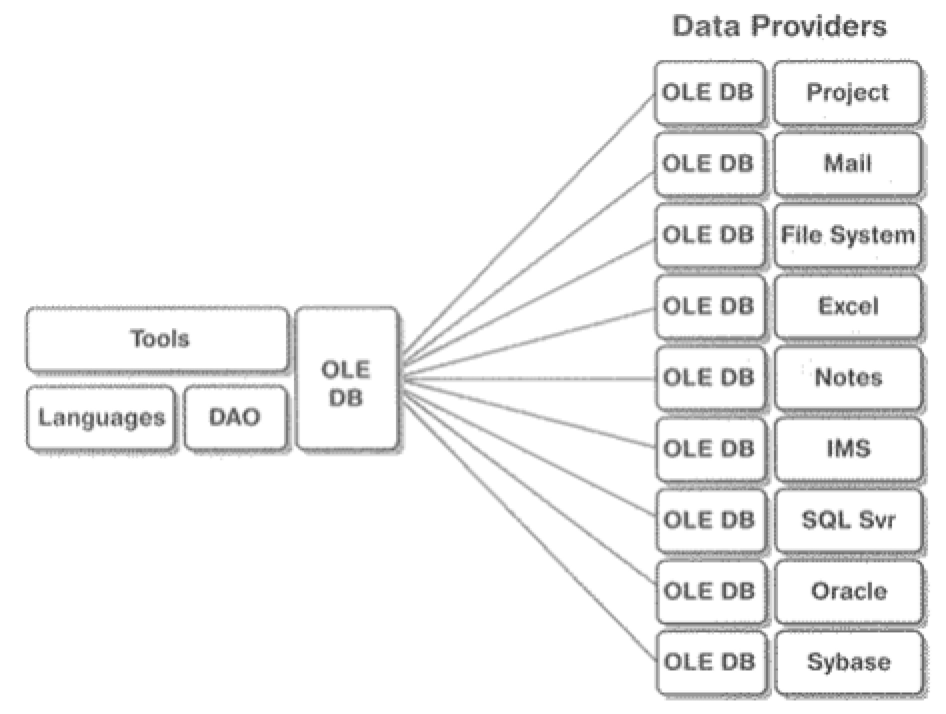


Types of Middleware (cont.)

- Distributed objects are also considered middleware because they facilitate interapplication communications.
- Distributed objects are really small application programs that use standard interfaces and protocols to communicate with one another

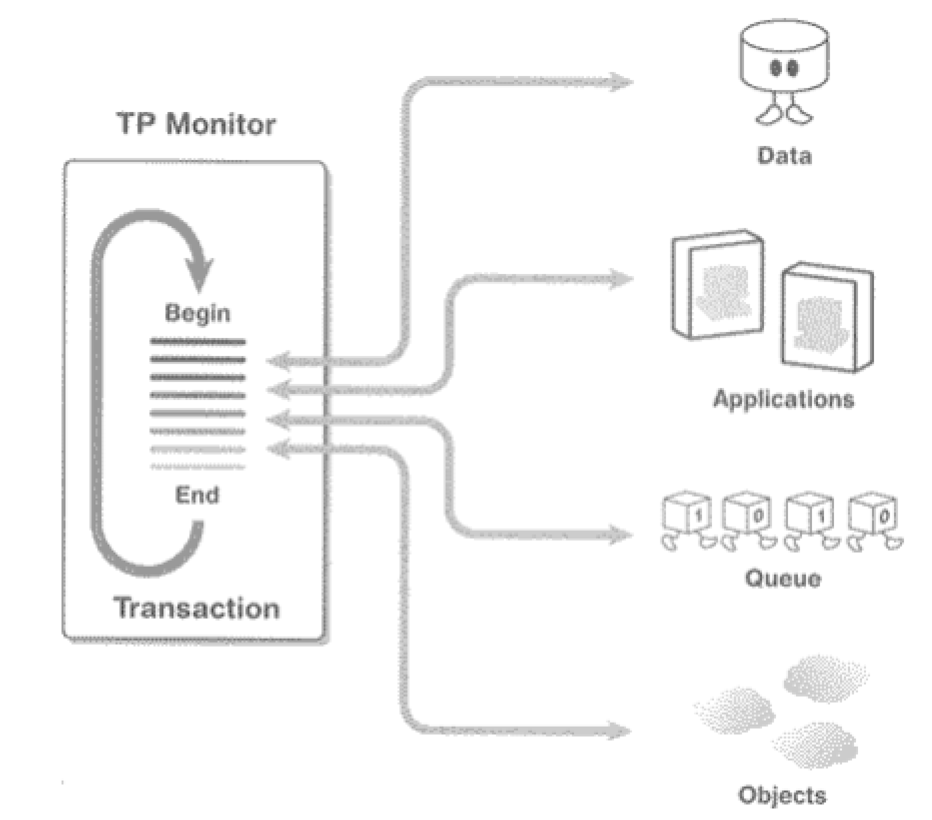
Types of Middleware (cont.)

- **Database-oriented** middleware is any middleware that facilitates communications with a database, whether from an application or between databases.



Types of Middleware (cont.)

- **TP Monitors**
- A transaction is a unit of work with a beginning and an end.
- The resources are integrated into the transactions and leveraged as part of the transaction.



Types of Middleware (cont.)

- **Message brokers** represent the nirvana of EAI-enabled middleware
- Can join many applications using common rules and routing engines
- Transform message schemas and alter the content of the messages

Middleware Models

- The **logical** middleware model depicts how the information moves throughout the enterprise conceptually. In contrast, the **physical** middleware model depicts how the information actually moves and the technology it employs.

Logical middleware model

- One-to-One versus Many-to-Many
  - **Point-to-point** middleware is middleware that allows one application to link to one other application—application A links to application B by using a simple pipe.

❖ Inability to properly bind together more than two applications

❖ Lacks any facility for middle-tier processing

✓ The great advantage of point-to-point middleware is its simplicity

Logical middleware model (cont.)

- **Many-to-many** middleware links many applications to many other applications.
- It is the best fit for EAI. This is the trend in the middleware world.
  - ❖ complexity of linking together so many systems

Logical middleware model (cont.)

Synchronous versus Asynchronous

- **Asynchronous:** The middleware software is able to decouple itself from the source or target applications, and the applications are not dependent on the other connected applications for processing.
  - Placing a message in a queue and then going about their business
  - ✓ The middleware will not block the application for processing
- **Synchronous** middleware is tightly coupled to applications
  - The applications are dependent on the middleware to process one or more function calls at a remote application.

Logical middleware model (cont.)

- The application is dependent on the middleware, problems with middleware, such as network or remote server problems, stop the application from processing.
- Synchronous middleware eats up bandwidth due to the fact that several calls must be made across the network in order to support a synchronous function call.
- **This disadvantage and its implications make it clear that the asynchronous model is the better EAI solution.**

Connection-Oriented and Connectionless

- **Connection-oriented** communications means that two parties connect, exchange messages, and then disconnect.
- **Connectionless** communications means that the calling program does not enter into a connection with the target process. The receiving application simply acts on the request, responding if required.

Direct Communications

- In direct communications, the middleware layer accepts the message from the calling program and passes it directly to the remote program.

Queued Communications

- Queued communications generally require a queue manager to place a message in a queue.
- The remote application then retrieves the message either shortly after it has been sent or at any time in the future.
- If the calling application requires a response (such as a verification message or data), the information flows back through the queuing mechanism.
- Most MOM products use queued communications.

Queued Communications (cont.)

- The queuing communication model's advantage over direct communications rests with the fact that the remote program does not need to be active for the calling program to send a message to it. What's more, queuing communications middleware typically does not block either the calling or the remote programs from proceeding with processing.

Publish/Subscribe

- Publish/subscribe (pub/sub) frees an application from the need to understand anything about the target application.
- All it has to do is send the information it desires to share to a destination contained within the pub/sub engine, or broker.
- The broker then redistributes the information to any interested applications.
- Publishers supply information about a topic, without needing to understand anything about the applications that are interested in the information

Request Response

- The request response model does exactly what its name implies. A request is made to an application using request response middleware, and it responds to the request.

Fire and Forget

- The fire and forget model allows the middleware user to "fire off" a message and then "forget" about it, without worrying about who receives it, or even if the message is ever received.
- This is another asynchronous approach.

Conversational-Mode

- A primary advantage of conversational-mode middleware is its ability to host complex business logic in order to maintain state and negotiate transactions with the application.

Tough Choices

- Although RPCs are slow, their blocking nature provides the best data integrity control.
- When using RPCs, updates are always applied in the correct order. If data integrity is more important than performance, RPCs may still be the best bet.
- The presence of easy-to-use interfaces will take the power of middleware—at one time the exclusive domain of the developer—and place it in the hands of the business user.