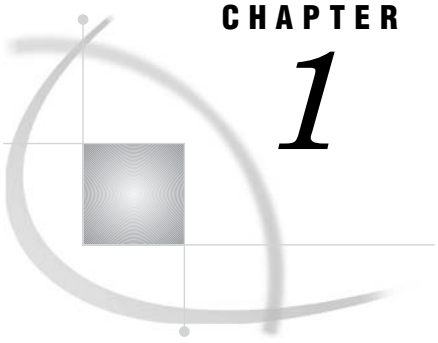


CHAPTER

1

Overview of Application Messaging

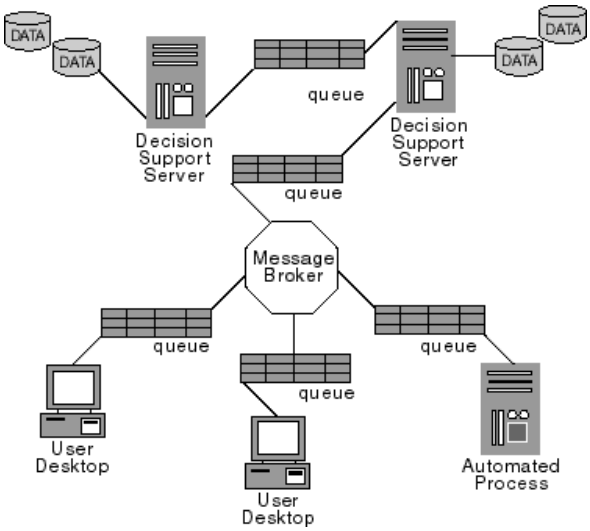
Application Messaging Overview

Application messaging architectures provide a platform that supports interoperability among loosely coupled applications over a message passing bus. When the targeted scope of interoperability is broad (for example, spanning multiple application systems and organizational boundaries), application messaging architectures might be required. This is because the likelihood of conformance in the software implementation base (for example, the selected distributed object standard) across the set of participating applications is diminished. Additionally, the set of participating applications can exhibit asynchronous, disconnected operation. These applications execute with no direct point-to-point communication session. However, they require guaranteed fulfillment of requests for service or event delivery.

This degree of operational heterogeneity introduces several requirements that are reflected in the application messaging infrastructure. Heterogeneity in the implementation base of the various applications (including perhaps, retrofitted legacy applications) suggests a need for a reasonably nonintrusive integration mechanism. The semantics of application messaging satisfy this need, generally expressing open, close, send, and receive functionality with flexible application-defined message structures. Heterogeneity with respect to the asynchronous, disconnected execution and notification modes of end-point participants introduces requirements for service qualities that include routing, assured just-once delivery, and retained sequencing. The architecture that has emerged within commercial application messaging products to express these quality-of-service properties is *store-and-forward queuing*.

In a store-and-forward model, messages are sent to named queues, which are in turn hosted at specific destination network addresses. The navigation of messages from their origin occurs through a transmission network that ensures the integrity of message delivery to the destination queue and presentation to the recipient process.

Display 1.1 The Store and Forward Messaging Model



Ever more frequently, the simple design pattern of two identifiable applications that interoperate over a message passing bus is inconsistent with the realities of an event-driven enterprise. Interdependencies across multiple applications with respect to events that occur within an enterprise combined with an ever-changing topology of event supplier and consumer applications are often present. Decision-makers require information pertinent to their domain of responsibility regardless of the reporting applications. Automated business processes require modification in rapid response to changing operational conditions. The ability to satisfy these requirements in a timely manner, and thereby reduce the latencies too common in information interchange, is critical to efficient and effective enterprise performance.

To support such dynamism, extended application messaging infrastructure facilities in the form of message brokers are emerging. Message brokers are being effectively positioned as enterprise application integration and event-management focal points, which function as hub processes that manage the information flow throughout an enterprise. Operationally, message brokers provide rules-based message routing and distribution as well as message transformation and augmentation capabilities that enable the removal of this aspect of implementation logic from participating applications.

Interfaces to three principal commercial messaging platforms, IBM WebSphere MQ (previously named MQSeries), Microsoft MSMQ, and TIBCO Software TIB/Rendezvous (including the Certified Message Delivery transport) are provided with SAS Integration Technologies. Support for these platforms enables SAS software's information delivery capabilities to be leveraged within various enterprise solution scenarios, including application integration, asynchronous and mobile synchronization, and event notification.

Support for client environments is broad. IBM provides WebSphere MQ on a vast array of operating system platforms with programming language support including C/C++, Java, and Cobol as well as ActiveX control support that enables Visual Basic participation. The Enterprise Java JMS facility also anticipates a provider for WebSphere MQ. Likewise, Microsoft provides full language support for MSMQ.

Supported Platforms for the SAS Messaging Interfaces

WebSphere MQ Functional Interface and the SAS Common Messaging Interface

The SAS interfaces to WebSphere MQ version 6 and later are supported for the following platforms:

- all supported UNIX platforms
- all supported 32–bit Windows platforms
- Windows on x64 (WebSphere MQ version 7 and later only)
- z/OS

The SAS interfaces to WebSphere MQ Client version 6 and later are supported in the following platforms:

- all supported UNIX platforms
- all supported 32–bit Windows platforms
- Windows on x64 (WebSphere MQ version 7 and later only)

MSMQ Functional Interface

The SAS interfaces to Microsoft Message Queuing Services (MSMQ) are supported on all of the 32–bit and 64–bit Windows platforms that are supported by SAS.

TIB/Rendezvous

SAS Integration Technologies supports both the reliable and certified message delivery features of TIB/Rendezvous Release 7.5.4 and later.

Support for using TIB/Rendezvous with the SAS Common Messaging Interface is available in the following operating environments:

- all supported UNIX platforms
- all supported 32–bit Windows platforms