UNIVERSAL MESSAGING: FIT FOR 21ST CENTURY MESSAGING CHALLENGES

Message-oriented middleware has delivered a fast, secure and flexible approach to integrating enterprise applications for years and is now used to provide the same advantages in the cloud. The appeal of this architectural approach? Organizations can react to changing business requirements with relative ease. Platforms, applications, services and microservices exchange data or events at near real-time speed with the appropriate level of security and delivery guarantee. Messaging provides the underpinning to the event driven architecture used by major Platform-as-a-Service vendors, making message-oriented middleware as relevant now as it was in the '90s.

TABLE OF CONTENTS

Messaging is critical to integrating applications, devices and business partners but can be complex. One of the key challenges for enterprises is guaranteeing message delivery across distributed systems on a wide range of delivery channels, including mobile networks, the web, enterprise networks, between remote devices, applications and the Internet of Things (IoT).

Software AG Universal Messaging is a message-oriented middleware product that provides protocol transformation, routing and delivery across public, private and wireless infrastructures. Universal Messaging has been built from the ground up to overcome the challenges of securely delivering data across different networks, between applications, devices and the IoT, while ensuring the required performance.

Read on to learn more about the Universal Messaging platform, its rich features, scalability and support for the widest range of protocols and open standards.

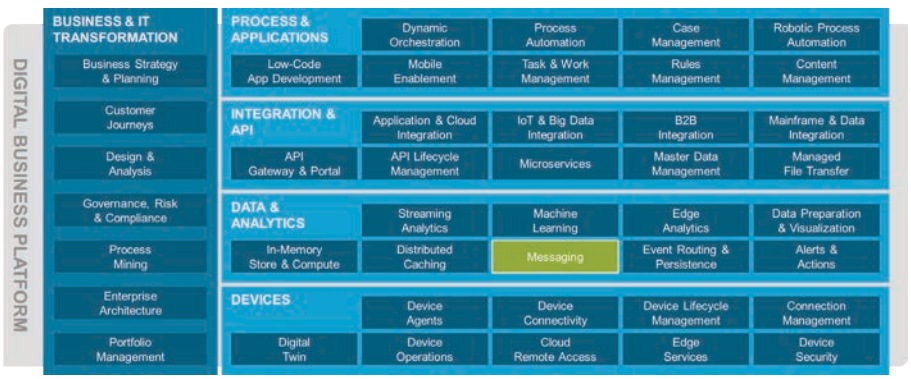
Overview Of Universal Messaging

Software AG Universal Messaging is a single solution for high-performance, low-latency messaging across a wide array of delivery channels, including all mainstream enterprise, web and mobile platforms. It delivers high-throughput messaging for resilient, secure and highly scalable applications, enabling enterprises to accelerate time-to-insight and action with ultrafast and highly scalable message delivery.

It can be used to meet the demands of a broad range of use cases, including: financial trading; online gaming; e-commerce/retail; the IoT, including data to and from remote devices and sensors; loyalty apps; real-time analytics; fraud detection; mobile POS and banking; price distribution; smart metering; web and mobile-enabled transactional systems.

Universal Messaging is a standards-based messaging platform, which offers protocol transformation, routing and delivery, with the widest possible compatibility for both your existing and future application, device and infrastructure requirements. It supports Message Queuing Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP) and Java® Message Service (JMS). Client API development can be conducted using C++, C#, Java and JavaScript®, ensuring your teams can use their existing skills and codebase to accelerate development.

Universal Messaging is the messaging service for the Software AG Digital Business Platform, providing the underlying framework that allows the components of the platform to exchange messages and events.Software AG's Digital Business Platform enhances, rather than replaces, the core IT systems of an enterprise and ties them together, improving the performance and efficiency of the component systems through its tightly integrated market-leading technologies.



Capabilities of the Digital Business Platform (messaging highlighted)

The benefits Of Universal Messaging

Universal Messaging streams data across enterprise, web and mobile platforms better and faster than any multi-product messaging platform on the market and has been benchmarked as the fastest universal messaging product available. It gives enterprises a number of distinct advantages when used as the foundation of a messaging infrastructure:

- **Flexibility** – Gain unrestricted access to your data across enterprise, web and mobile channels from the widest range of devices and platforms. Universal Messaging's feature rich platform ensures it can be configured to meet the needs of your environment
- **Simplicity** – Industry-leading messaging standards, wire protocols and transport options from a single solution, eliminating the complexity and latency incurred by bridging multiple messaging systems
- **Scalability** – Scale applications as new delivery channels require and respond to the demands of new devices or protocols. Support thousands of users, and scale with any increase in user numbers or data volumes
- **Security** – Communicate with a secure transactional system that includes encryption, pluggable authentication and entitlements services that exceed regulatory requirements
- **High availability** – Guaranteed message delivery using transactional semantics and persistence that provides unsurpassed levels of availability
- **Multi-language APIs** – Full interoperability between client APIs of your choice across enterprise, web and mobile platforms using C++, C#, Java and JavaScript
- **Multiple messaging paradigms** – Universal Messaging is able to support a number of messaging paradigms that an enterprise might need, including publish/subscribe and message queues.

Standards based from the ground up

In all areas of software, there are standards that allow different products and applications to work together. Universal Messaging has been designed to support the following proven messaging and message interface formats, the HTML5 standard, and a number of programming languages. Combined with the right Universal Messaging delivery paradigm, the protocols offer industry-leading, assured message delivery and performance in terms of low latency efficiency across low-bandwidth or intermittent networks, and on devices where processing or power resources may be limited.

- **MQTT** – MQTT (Message Queuing Telemetry Transport) is a publish/subscribe, simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth by IBM®/ Eurotech in 1999. The simplicity and low overhead of the protocol make it ideal for the emerging "machine-to-machine" (M2M) or IoT world of connected devices, and for mobile applications where bandwidth and battery power are at a premium. The protocol is openly published with a royalty-free license, and a variety of client libraries have been developed especially on popular embedded hardware platforms, such as Arduino/Netduino, Mbed and Nanode.
- **AMQP** – The AMQP protocol offers a layered model, consisting of transport and connection security, frame transfer and message transfer semantics without any assumptions on source/destination models or deployment topologies. This provides a portable, secure, binary, symmetric message exchange capability between applications whether involving a classic message-oriented middleware broker, a cloud messaging infrastructure instance, or a peer-to-peer message exchange service.
- **JMS** – Java® language clients and Java language middle-tier services must be capable of using message-oriented middleware systems. JMS provides a common way for Java language programs to access these systems. JMS is a set of interfaces and associated semantics that define how a JMS client accesses the facilities of an enterprise messaging product, such as Universal Messaging.

Of course, some third-party products struggle to cope with these standard messaging paradigms or the messages themselves. By combining Universal Messaging with Software AG's webMethods Integration Server, incoming data formats can be mapped to new formats recognised by your applications and devices – decisions can also be made on the data contained within those messages. The powerful combination of Universal Messaging with Integration Server allows organization to asynchronously connect applications and products, breaking down silos of data, giving the business the flexibility to react to changing business needs.

Client-side development

Universal Messaging client libraries are the mechanism by which users enable their applications to send and receive messages and events with Universal Messaging. The Universal Messaging servers provide the integration point between message senders and receivers, whether a device, cloud service or application. Universal Messaging offers three categories of enterprise clients with a wide range of languages supported for each API, giving you a choice of approaches based on the skills within your teams and the requirements for your messaging implementation:

- **Enterprise client APIs** – Universal Messaging Enterprise APIs allows developers to implement real-time publish/subscribe functionality into enterprise-class applications using a range of languages including Java, C++ and C# .NET.
- **Web client APIs** – Our web-based messaging solution allows web developers to implement real-time publish/subscribe functionality in browser applications or Rich Internet Applications (RIAs) using JavaScript. The API is a pure JavaScript solution that allows developers to use JavaScript and HTML to build HTML 5 clients. These can publish and subscribe to Universal Messaging stores, and asynchronously receive events in real time.
- **Mobile client API** – Implementing real-time publish/subscribe functionality within mobile phone applications on a range of devices, including Apple® iPhone® and Android®, is straightforward with these APIs. Written in C++, developers can use a combination of Objective-C and C++ to build applications for Apple devices. Android® developers can implement the functionality using the Universal Messaging Enterprise API for Java.

A paradigm for every message

When designing the messaging architecture for your enterprise, there are usually trade-offs to be made based on your business goal, the needs of the application, number of users, networking conditions, throughput and performance required. Each use case is different, and it is a balancing act to create the perfect messaging infrastructure which provides a platform that will not confine you to its own limitations. Universal Messaging is uniquely placed in the market through its combination of performance, functionality and standards support to help you build the infrastructure that your business needs.

Universal Messaging supports two broad messaging paradigms: publish/subscribe and message queues. Universal Messaging clients can use a mixture of these paradigms from a single session. In addition to this it is possible for clients to further control the conversation that takes place with the server by choosing particular styles of interaction. These styles are available to both readers and writers of messages and include asynchronous, synchronous, transactional and non-transactional.

- **Synchronous messaging** – Perfectly suited to situations where delivery of the message must be guaranteed. Here, a client sending the message waits for an acknowledgement from the messaging server to confirm it has received the message. This is often combined with persisting the message to permanent storage to ensure it is retained if anything untoward happens. The client does not delete the message until it has received the acknowledgement, this slows down the client and the server is slowed down by the disk write, but message delivery is assured.

- **Asynchronous messaging** – High throughput lends itself to asynchronous messaging, sometimes known as "fire and forget." In this instance, the client does not wait for an acknowledgement from the messaging server. Instead the messages are sent to the server as fast as the network will allow. In this scenario, the server will typically hold the messages in memory, to further support the need for speed. In this instance if there is any problem on either the client, network or server messages may be lost. Note that it is possible with Universal Messaging to persist messages to disk if required.
- **Transactional messaging** – Ensures the highest-possible quality of service for messages by placing the highest garantee on message delivery. On publishing, transactional messages are acknowledged by the server after being fully processed and replicated around the cluster to ensure message loss is impossible. On subscription, transactions allow explicit acknowledgement of message receipt to avoid duplication even in the event of failures.
- **Non-transactional** – This allows lower-latency, higher-throughput messaging by reducing the strong acknowledgement guarantees outlined for transactional messaging.
- **Persistent messaging** – Messages in Universal Messaging can be persisted to disk. This helps to prevent loss of messages even if servers go offline simultaneously. However, persistence does not compensate for using non-transactional messaging as the messages can only be written to disk once they have been received. Messages that are "in flight"at the point of the server failing will be lost if an asynchronous, non-transactional messaging architecture is used. Once the messages are on disk, they can be recovered on restart.
- **Non-persistent messaging** – Allows messages to be stored in-memory. For high velocity data, operating entirely in-memory can offer significant performance improvements. The specialised clustering technology in Universal Messaging can allow messages to be replicated in-memory throughout a cluster, ensuring resilience to node failure without requiring potentially slow persistence.

Messaging paradigms supported

- **Publish/subscribe (using channels/topics)** – This is a messaging model where the sender (publisher) of a message and the consumer (subscriber) of a message are decoupled. When using the channels/topics, readers and writers of events are both connected to a common topic or channel. The publisher publishes data to the channel. The channel exists within the Universal Messaging server. As messages arrive on a channel, the server automatically sends them on to all consumers subscribed to the channel. Universal Messaging supports multiple publishers and subscribers on a single channel.
- **Publish/subscribe (using datagroups and datastreams)** – Universal datagroups provide an alternative to channels (JMS topics) for the publish/subscribe messaging paradigm. Universal Messaging datagroups provide a very lightweight subscription grouping structure that allows you to manage user subscriptions remotely and transparently. Datagroups separate subscription logic from the client application, allowing subscription management to be performed from other applications. This can greatly simplify client application logic and mitigate the difficulties of changing system architecture. The process of managing subscribers can be carried out by publishers themselves or by some other process. Datagroups are designed to support large numbers of consumers whose subscriptions are typically fluid in nature. The addition or removal of consumers from datagroups can be entirely transparent from the consumer perspective.

- **Point to point using message queues** – Like publish/subscribe, message queues decouple the publisher or sender of data from the consumer of data. The Universal Messaging server manages the fan out of messages to consumers. Unlike publish/subscribe with channels, however, only one consumer can read a message from a queue. If more than one consumer is subscribed to a queue then the messages are distributed in a round-robin fashion.

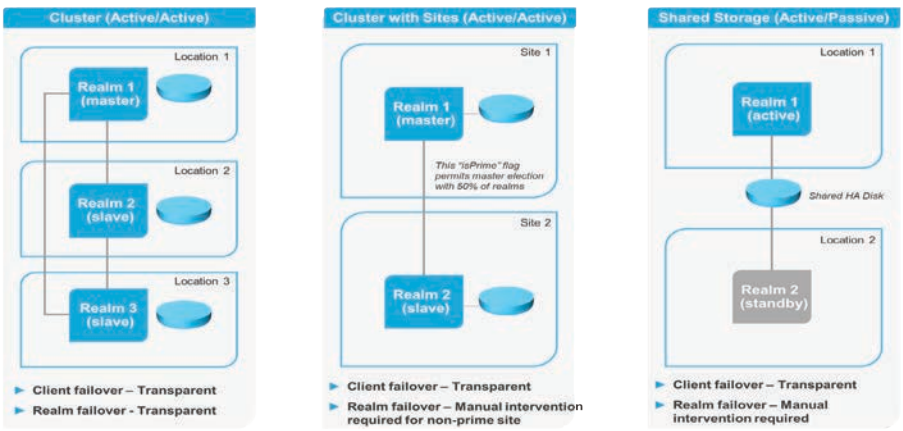
Scalability and resilience

Universal Messaging can scale to meet the messaging throughput of the most demanding applications and thousands of active users. It has proved itself to be a critical component for sectors including financial services, trading floors, ecommerce, global process businesses and manufacturing.

Equally, for any mission-critical messaging application, it is essential to be able recognize a failover quickly, and recover from it without any message loss or impact on the systems and stakeholders it supports. Universal Messaging protects against application and service failures, system and hardware failures, and site failures.

Universal Messaging addresses these challenges through its proven Universal Messaging cluster technology, which consists of Universal Messaging servers working together to provide increased availability and reliability. It also provides support for business contingency and disaster recovery.

Universal Messaging supports both active/active and active/passive clustering. In a Universal Messaging cluster, an individual Universal Messaging server is referred to as a cluster node. If one cluster node becomes unavailable, another cluster node takes over the messaging operations. Each cluster appears to be a single Universal Messaging server, greatly simplifying failover management with connected clients.



Universal Messaging leverages local disk storage to avoid contention associated with shared central storage and supports high availability through redundancy.

Active/active clustering

In an active/active cluster, multiple servers are active and working together to publish and subscribe messages. Universal Messaging clients automatically move from one server to another server in a cluster as required or when specific servers within the cluster become unavailable to the client for any reason. The state of all the client operations is maintained across the cluster to enable automatic failover.

In an active/active cluster, one of the nodes is designated as the master node. It is the role of the master node to be the arbiter of truth in the cluster. For all cluster operations that require coordination, such as destructive reads on queues, durable delivery, it is the master who is the controlling agent. If the master node leaves the cluster or goes offline due to power or network failure, the remaining active cluster nodes elect a new master, provided more than 50 percent of the cluster nodes are available to form the cluster.

The master node broadcasts the requests to the other cluster nodes to ensure that all the servers are in sync. If a cluster node disconnects and reconnects, all the states and data are recovered from the master node.

Active/passive clustering

Active/passive clustering is a solution that uses compatible third-party clustering software of your choice and special purpose hardware to minimize system downtime. Active/passive clusters are groups of computing resources that are implemented to provide high availability of software and hardware computing services. Active/passive clusters operate by having redundant groups of resources (such as CPU, disk storage, network connections and software applications) that provide service when the primary system resources fail.

In a high-availability active/passive clustered environment, one of the nodes in the cluster will be active and the other nodes will be inactive. When the active node fails, the cluster fails over to one of the inactive nodes automatically. As a part of this failover process, clustering software will start the resources on the redundant node in a predefined order (or resource dependency) to ensure that the entire node comes back up correctly.

Universal Messaging can run in an active/passive cluster environment, under Windows® or UNIX®.

Administration and monitoring tools

Universal Messaging includes various approaches to management and administration. Command line tools allow for the creation of scripts or manual administration, Enterprise Manager provides a product specific administration and monitoring tool, Software AG Command Central is a Digital Business Platform wide deployment, monitoring and administration tool, which also has a REST API, and lastly, Universal Messaging provides a rich proprietary API. Through these tools, and directly through the management APIs, access to a number of powerful monitoring and configuration features is possible, categorized into several key areas:

- **Statistical data** – Through the use of the Universal Messaging Administration API, clients can access a very detailed range of performance related data. Performance metrics can be gathered at many levels, ranging from server throughput statistics to individual client-connection, round-trip latency details.
- **Management event monitoring** – Most client and server induced actions in Universal Messaging result in a management event being created. Asynchronous listeners can be created using the Administration API that enables administration clients to capture these events. As an example, this could be used to dynamically create channel resources for a client, when it connects to a Universal Messaging server.
- **Resource creation** – Resources can all be created programmatically using the Universal Messaging Administration API. Coupled with statistical data and event monitoring, resources can be created "on the fly" based on rules defined by criteria based on the events and statistics.
- **Configuration management** – Every Universal Messaging Server has a number of configurable parameters. In addition, specific interfaces supporting specific protocols and plugins can be added to Universal Messaging servers. Universal Messaging's configuration management feature allows clients to snapshot configurations and generate configuration XML files. New servers can then be very quickly configured with the XML files enabling the very fast bootstrapping of new environments.

- **Third-party management support** – While Universal Messaging's Administration API can be used directly to integrate with third-party products, Universal Messaging servers also support JMX® (Java Management eXtensions), meaning any JMX management tool with the appropriate security permissions may be adopted.

Take the next step

Universal Messaging has been built from the ground up to allow a wide range of vertical to sectors to overcome the challenges of securely delivering data across different networks, between applications, devices and the IoT, while ensuring the required performance. It can be easily integrated into your existing infrastructure either on its own, or as part of the Software AG Digital Business Platform to further enhance your core IT systems. It is fully extensible, can act as the messaging backbone of your organization, and can be deployed both on-premises or in the cloud.

Other pure-play solutions stop at providing simple message transportation, which is not enough for the modern enterprise. Universal Messaging is a complete API-based platform designed to enable complex high-volume message translations and deep integration.

Universal Messaging is at work, right now, in some of the most challenging mission-critical messaging environments in sectors, such as retail, financial services and industrial applications. Whether as a standalone implementation or with the Software AG Digital Business Platform, Universal Messaging is improving the reliability of messaging and the performance of businesses across the globe.

With 50 years of software expertise, 300 partners and 4,000 employees worldwide, Software AG is a partner that can go the distance as your company grows and new challenges emerge. Our customers include nine of the 10 largest U.S. banks and leading online payment, credit card and investment firms; major airlines, train companies and transit authorities, even a space travel organization.

Learn more about Universal Messaging by downloading the 90-day free trial or speaking with your Software AG account team.

Check out these recommended resources:

- Universal Messaging website
- Universal Messaging community site (includes online documentation)
- Universal Messaging Free Trial
- Universal Messaging Fact Sheet
- Universal Messaging for webMethods Integration Fact Sheet

2018_10_Corporate_WP_Universal Messaging

software AG