U1 TECHNOLOGIES

AmbrosiaMQ™ – Feature Overview

U1 Technologies' AmbrosiaMQ™ is a messaging engine focused on addressing the three competing facets of performance, scalability and reliability – while ensuring an easy-to-use and easy-to-manage runtime environment for global deployments.

This document provides an overview of AmbrosiaMQ's features and how they support the following general objectives:

- Performance & Scalability
- Reliability & Fault Tolerance
- Usability
- Security
- Portability
- Global Deployments with Centralized Management

PERFORMANCE & SCALABILITY

Connection load balancing	AmbrosiaMQ performs weighted random selection, which simplifies the implementation and avoids trying to route all new connections to the least loaded broker.
Security event auditing	AmbrosiaMQ can be configured to write log events to flat files instead of a database, driving higher efficiencies.
Virtual circuits	Virtual circuits enable an application to simultaneously leverage the speed of reliable delivery and take corrective actions when its peer cannot take delivery of the message. Two processes establish a Virtual Circuit (VC) through which they receive notifications about each other's connectivity status (i.e., up or down).
Large data sets	AmbrosiaMQ provides very generalized, efficient flow control and error notification for large data sets.
Bandwidth limiter	Enhances performance of the overall system by limiting the amount of bandwidth that can be consumed by a client.
Fast serialization	Fast Serialization is a technique used to compress each message for higher performance and scalability. This technology supports multiple dictionaries and data types, and supports dictionary versioning, as well as the ability to modify a message without de-serializing it. Fast serialization can be used stand alone or outside of the messaging system.
Inter-broker acceptor addresses	AmbrosiaMQ allows multiple addresses for any inter-broker connection as well as client load balancing.
Peer-to-peer messaging	Enables ultra-high speed communication between two clients.

Intra-process messaging	Enables design of application whose components can run within a single process or across multiple processes. Allows switching from inter-process to intra-process and vice versa without recoding the application.
Wildcard publishing	Provides the ability to notify a select group of subscribers (based on subscription) without using broadcasts.
Regional route limits	Enables designation of geographical regions for brokers and collectives. Provides the ability to limit subscription and message propagation to a single geographical region.
Ability to get length/skip over serialized object	Optimizes message processing by allowing a message receiver to skip over objects without de-serializing them.
Embedded TCP proxy	Provides a highly efficient method for passing messages from one network zone to another.
Durable message discarding from clients queue upon disconnect	Optimizes clients by relieving them from processing duplicate messages.
Publisher option for setting discardable	Improves message processing by enabling the publisher to designate messages as discardable.
Client subscription manager	Reduces the number of network calls to a broker by providing subscription reference counting. Enhances application recovery semantics by enabling automatic re-subscription upon reconnect.
Concurrent SubjectSpace	Eliminates synchronized access to a SubjectSpace thereby significantly reducing the latency of multi-threaded applications that use a SubjectSpace.
Modify serialized messages	Reduces the overhead of message re-publishing by allowing modification without the need to de-serialize the message.
Proxy server support	AmbrosiaMQ supports proxy server load balancing, or ordered selection. In addition, it supports digest authentication and can tunnel SSL or TCP through a proxy server.
General discardability	AmbrosiaMQ implements a very fine-grained control by allowing messages at any priority to be designated as discardable or not.

Reliability & Fault Tolerance

Redundant durable subscriptions	Enhances the reliability of the system by allowing multiple locations at which durable messages can be collected on behalf of a client.
Redundant bridge brokers	Greatly enhances the reliability and load balancing of the overall broker network by permitting multiple brokers to act as a bridge between collectives.
Replicated queues	Enhances reliability of guaranteed queues by allowing multiple redundant queues.

Heartbeat monitoring	Supports heartbeat timeouts on inter-broker connections. Additionally, individual clients can set different heartbeat timeouts.
Zero-weight load balanced subscription	Facilitates a mechanism for establishing *stand-by* subscribers, which only receive messages if no other subscribers are available.
Separation of discardable queues from reliable queues	Enables consistent queue management based on reliability requirements.
Variable client queue sizes	Provides fine grain control of queue sizes based on client application's requirements.
Stoppable publishers	Enables applications to choose if a slow client subscriber should be terminated or cause the publisher to stop sending messages.
Durable subscriptions and integration with load balanced connections	Offers a standard implementation for guaranteed delivery. Allows a client to connect to any broker and receive guaranteed messages.
Multiple broker URLs for durable pools	Facilitates switching of brokers that host durable subscribers.
Lock access to broker transaction logs	Enhances broker's startup reliability by ensuring that the broker will not use an incorrect transaction log.
Topology discovery and connectivity verification	Offers a rapid and reliable method for discovering all brokers. Enables connectivity, subscription and publication testing to all brokers.
Advanced flow control	Enhances overall reliability of the system by implementing advanced flow control features. Provides ability to link flow control across client applications.
Database Auto-create and versioning	Provides better control over deployment and rollback of AmbrosiaMQ installations.
Dictionary versioning	Allows multiple versions of message schema to exist in the system thereby facilitating gradual upgrades of client applications.

USABILITY

Message trace routing and client-to client trace routing	Greatly enhances system diagnostics by providing message routing data, including latency at each hop. Facilitates a method by which a client can be instructed to send a message to another client and trace its route.
Health check web tool	Enables system managers to quickly determine the status of an AmbrosiaMQ broker.
Subscriber enumeration API	Improves system diagnostics by facilitating a way to determine who is currently subscribed to a subject.
Point-to-point messaging (queues)	Allows an application developer to use the queue paradigm, which enables them to design applications requiring guaranteed and load-balanced delivery of messages.

Selectors for POJOs, date/time and time zones	Provides application developers with easy and efficient message filtering capabilities.
Tool for retrieving bridge fail-over statistics	Enhances system diagnostics by providing information about bridge brokers
Last login time	Enables system administrators to determine the last time a user connected to a broker.
SubjectCache API	Facilitates management of single value objects through a very efficient mechanism.

SECURITY

JAAS Integration	Provides an industry standard method for plugging in any authentication mechanism.
Permission groups	Facilitates implementation of role-based access control.
Additional built-in security groups	Enhances the security of the system by providing fine grain access to security objects to designated groups.
Client authentication through SSL and certificates	Allows clients to use digital certificates to authenticate with the broker via SSL. The use of SSL and JAAS enables additional security enhancements such as Revocation Checking and Trust Management.
Support for NTLM V2.0	Increases secure deployment options for external users.
Account disable	AmbrosiaMQ has a group membership implementation and can disable an entire group in one action.

PORTABILITY

Complete JMS 1.1 implementation	Enables application developers to use the industry standard JMS API.
JMS integration with XA and JNDI	Extends JMS usability by integrating it with Java XA and JNDI compliant services.
AmbrosiaMQ to JMS bridge	Enables AmbrosiaMQ applications to exchange messages with JMS applications.
.NET API	Enables .NET applications to natively leverage AmbrosiaMQ.

GLOBAL DEPLOYMENTS WITH CENTRALIZED MANAGEMENT

Zones	AmbrosiaMQ supports multiple zones, with bi-directional connectivity rules. Any login (not just Administrators), can be restricted by zone.
Broker Admin Console	The Broker Admin Console provides administrators with complete control over a global deployment of brokers.
Configuration Servers	Configuration servers include security configuration and provides a Single point of administration.
Ganglia integration	AmbrosiaMQ integrates with this open source product and makes many broker statistics available for any Ganglia-aware monitoring tool.