MODAPTO [101091996]: Modular Manufacturing and Distributed Control via Interoperable Digital Twins

13.6.2 MessageBus communication protocols and data exchange formats

2025-08-11

Introduction

The MessageBus communication protocols and data exchange formats within the MODAPTO system provide the essential infrastructure for enabling seamless, real-time communication between digital twins, physical assets, and system components. This sophisticated communication framework supports multiple protocols and formats to ensure interoperability, scalability, and reliability in complex manufacturing environments.

The MODAPTO MessageBus architecture implements a hybrid communication model that combines publish-subscribe patterns for event-driven communications with point-to-point messaging for direct coordination between specific components. The core components of a message bus architecture include the message sender, message receiver, message broker, and message channel, where the message sender sends a message to a message channel, and the message broker manages the message channel and ensures that the message is delivered to the appropriate receiver.

Protocol selection within the MODAPTO MessageBus is based on the specific communication requirements of different system components and operational scenarios. The primary protocols implemented include MQTT (Message Queuing Telemetry Transport) for lightweight IoT device communications, Apache Kafka for high-throughput streaming data, OPC UA for industrial automation integration, and HTTP/REST for web-based service interactions. Each protocol is optimized for specific use cases and communication patterns within the manufacturing environment.

MQTT implementation focuses on enabling efficient communication with resource-constrained devices and sensors throughout the production system. The protocol's lightweight nature and support for quality of service levels make it ideal for collecting real-time sensor data from production modules and distributing control commands to actuators and controllers. The MQTT broker manages topic subscriptions and message delivery, ensuring that all relevant system components receive necessary updates.

Apache Kafka integration provides high-throughput, fault-tolerant streaming capabilities for managing large volumes of production data and events. Kafka's distributed architecture and persistence capabilities make it suitable for scenarios requiring guaranteed message delivery, historical data replay, and high-volume data processing. The Kafka integration includes custom producers and consumers optimized for MODAPTO-specific data formats and communication patterns.

OPC UA integration enables seamless communication with industrial automation systems and programmable logic controllers (PLCs) throughout the production environment. OPC UA provides a framework for secure and reliable data transfer between different devices, systems and software applications in industrial environments, providing a standardised and functional communication tool that ensures seamless connectivity and information exchange between different platforms.

The OPC UA implementation supports both client and server modes, enabling MODAPTO digital twins to both consume data from existing automation systems and provide data to external control systems.

Data exchange formats within the MessageBus are standardized to ensure consistent interpretation and processing across all system components. The primary formats include JSON (JavaScript Object Notation) for lightweight, human-readable data exchange, Protocol Buffers for efficient binary serialization. Each format is selected based on the specific requirements of data volume, processing efficiency, and interoperability needs.

JSON-based messaging is used extensively for configuration data, status updates, and human-readable logging information. The JSON format provides excellent interoperability with web-based systems and enables easy debugging and monitoring of system communications. Custom JSON schemas are defined for MODAPTO-specific message types, ensuring consistent data structure and validation across all system components.

Protocol Buffers implementation focuses on high-performance, efficient serialization for high-frequency data exchanges such as sensor readings, control signals, and real-time status updates. The binary format significantly reduces message size and processing overhead compared to text-based formats, making it suitable for bandwidth-constrained or high-volume communication scenarios.

Message routing and filtering capabilities enable efficient distribution of messages to relevant system components while minimizing unnecessary network traffic and processing overhead. The routing system supports both topic-based routing for publish-subscribe patterns and content-based routing for more sophisticated message distribution strategies. Dynamic routing rules can be configured based on message content, recipient capabilities, and system load conditions.

Security implementations within the MessageBus ensure that all communications maintain appropriate confidentiality, integrity, and authenticity. This includes transport-layer security (TLS) encryption for network communications, message-level authentication and digital signatures, and access control mechanisms that restrict message publishing and subscription based on component roles and permissions.

Quality of Service (QoS) management provides guarantees for message delivery reliability and timing based on the specific requirements of different communication scenarios. The QoS system supports multiple delivery guarantee levels, from best-effort delivery for non-critical status updates to guaranteed delivery with acknowledgment for critical control commands and safety-related messages.

Event aggregation and correlation capabilities enable the MessageBus to process and combine related events from multiple sources, reducing message volume and providing higher-level abstractions for system monitoring and control. The aggregation system can combine sensor readings, generate derived events based on multiple inputs, and provide trend analysis for system optimization.

13.6.1 Integration with MODAPTO Digital Twin Manager architecture

Performance optimization features ensure that the MessageBus can handle the communication requirements of large-scale manufacturing systems with hundreds or thousands of connected components. This includes load balancing across multiple broker instances, message batching for improved throughput, and caching mechanisms for frequently accessed data.

Integration APIs provide standardized interfaces for connecting external systems and third-party components to the MODAPTO MessageBus. These APIs support common integration patterns and provide client libraries for multiple programming languages, enabling easy integration of existing manufacturing systems and custom applications.

References

1. Apache Software Foundation. (2025). “Apache Kafka Documentation: A Distributed Streaming Platform.” Technical Documentation.
2. OPC Foundation. (2023). “OPC Unified Architecture Specification Part 1: Overview and Concepts.” OPC 10000-1.
3. OASIS. (2019). “MQTT Version 5.0 OASIS Standard.” OASIS Message Queuing Telemetry Transport Technical Committee.
4. Alam, K. M., & El Saddik, A. (2017). “C2PS: A digital twin architecture reference model for cloud-based cyber-physical systems.” IEEE Access, 5, 2050-2062.
5. Message Bus Architecture supports different messaging protocols like AMQP, JMS, MQTT, and more, which define the message format, message delivery guarantees, and other messaging features.