Integration

# Integration

## Transact

### Key-points

* **API**s and Web Services for System **Integration**

Transact provides a comprehensive set of RESTful APIs using JSON payloads, adhering to semantic versioning and documented with OpenAPI specifications. These APIs cover most banking functionalities out-of-the-box and can be extended or customized via the Workbench low-code tool to create tailored APIs. The TransactAPI component acts as an API gateway, managing and securing API access.

* **Real-Time** Data **Streaming** and **Event** Processing Support

Event-based integration is supported through business events (triggered by business operations) and data events (triggered by table updates). Events follow the CloudEvents standard, ensuring schema consistency. The architecture employs the Transactional Outbox pattern and an Event Store microservice for event auditing, replay, and reconciliation. Integration with pub/sub systems like Apache Kafka, Azure Event Hubs, and AWS Kinesis enables scalable real-time event handling.

* Messaging and Queuing Capabilities for **Integration**

Communication with the core uses the OFS protocol via the Temenos Open Connectivity Framework (TOCF) with JMS/MQ queues for resilience and throughput management. The MQBroker component supports message queuing for legacy systems. The Adapter microservice uses Apache Camel templates for protocol transformations and message routing, facilitating integration with various messaging patterns.

* Data Synchronization and Consistency Handling

Transact employs a strongly consistent database for core transaction processing, ensuring all users see the same data simultaneously. For read-optimized scenarios, CQRS microservices synchronize data asynchronously using events, providing eventual consistency with low latency for reporting or user interfaces. The Transactional Outbox pattern ensures ordered and reliable event processing.

* **Protocol** Support and Communication Standards

Transact supports HTTPS for secure communication, REST APIs with JSON payloads, JMS/MQ for messaging, and OFS for core messaging. The Adapter microservice enables protocol transformations including XML with XSLT, bulking, and debulking of files. CloudEvents standardization ensures consistent event schemas across systems.

* Batch Processing and File-Based **Integration** Support

The Adapter microservice facilitates batch file processing, bulking, and protocol transformations, enabling integration with systems requiring bulk data transfers. Secure file transfers are supported via SFTP, which can be processed by the Adapter. Banks may also build custom ETL solutions leveraging the Outbox capability for data export, though Data Hub is the recommended product for downstream data integration.

### Details

APIs and Web Services: Temenos Transact provides a comprehensive RESTful API framework built on Apache Camel, enabling synchronous communication with external systems. These APIs use JSON payloads, adhere to semantic versioning, and are documented with OpenAPI specifications, ensuring ease of use and backward compatibility. Security is enforced via HTTPS transport, with authentication managed through Single Sign-On (SSO) using Keycloak supporting OIDC/OAUTH2 and SAML protocols. Use cases include real-time transaction processing, account inquiries, and product configuration. Performance benefits stem from lightweight JSON payloads and stateless REST design, facilitating scalable and responsive integrations.

Real-Time Data Streaming: Transact employs event-based integration using Apache Kafka for asynchronous data sharing. Two event types exist: Business Events (transactional context) and Data Events (table-level changes), both implemented via the Transactional Outbox pattern to guarantee ordered, reliable event delivery. The architecture supports high throughput and scalability, integrating with pub/sub systems like Kafka, Azure Event Hubs, and AWS Kinesis. Events conform to the CloudEvents standard, ensuring schema consistency and simplifying cross-system processing.

Messaging and Queuing: Messaging is primarily handled through Kafka-based event streams, providing durable, ordered, and fault-tolerant message delivery. The Adapter microservice leverages Apache Camel templates for protocol transformations and message routing, supporting patterns such as debulking and bulking. Queue management benefits from Kafka’s partitioning and replication features, ensuring resilience and high availability.

Data Synchronization: The core database is a strongly consistent, write-optimized repository without embedded business logic, supporting concurrent transactions with strict consistency. For read-heavy operations, Temenos offers CQRS microservices that synchronize data asynchronously via events, providing eventual consistency with low latency for reporting and user interfaces. This separation optimizes performance and scalability by isolating transactional and query workloads.

# Document Information

Generated by: Temenos RAG AI System

API Key: eyJhbGciOi...-wUIZ1Y8q0

API Calls Made: 2

Generated on: 2025-10-06 08:27:08

Product: ['Transact']

Pillar: Integration

Region: GLOBAL