MAXTRANSIT (MAXTOUCH POS)  ERP

Universal Interface

1. Introduction

The POS-ERP interface enables bidirectional data flow to support retail operations, including product management, sales reporting, inventory control, and transaction processing. Key benefits include:

- Real-time or near-real-time data synchronization.
- Reduced manual data entry errors.
- Improved inventory accuracy and financial reconciliation.

This interface uses XML for structured data exchange, adhering to standards like XML 1.0 with UTF-8 encoding. It is designed for scalability in multi-store environments.

Scope

- Systems Involved
    - POS Cash System: Maxtouch (the client-facing point-of-sale application running on checkout terminals).
    - ERP System: Central enterprise backend responsible for inventory management, financial accounting, procurement, master data, and reporting.
- Primary Data Flows
    - From ERP to POS (Maxtouch):
        - Product catalog synchronization (new items, updates, deactivations)
        - Pricing and discount updates
        - Current inventory/stock levels
        - Promotional rules and configuration changes
    - From POS (Maxtouch) to ERP:
        - Completed sales transactions (including items, quantities, discounts, taxes, payments)
        - Inventory movements and adjustments (returns, damages, manual corrections, stock receipts at store level)
        - End-of-day / end-of-shift financial and operational summaries (cash reconciliation, tender totals, voids, refunds)
- Scalability & Extensibility
  The interface is designed to be highly flexible and scalable:
    - Additional fields can be added to the XML structure at any time without breaking existing integrations.

- New or custom fields introduced in the XML message can be mapped in the Maxtransit agent.
- In the Maxtransit configuration, each received XML field can be explicitly assigned to the corresponding Maxtouch internal parameter, screen field, database column, or business logic handler.
- This on-demand extensibility allows rapid adaptation to new business requirements (e.g., new loyalty attributes, serial/lot tracking, multi-currency support, custom taxes, nutritional info, or regulatory fields) without modifying core POS or ERP application code.

2. Architecture

2.1 Communication Mechanism

MAXTOUCH POS system runs a local agent called Maxtransit, which handles communication with the ERP. Maxtransit operates as a middleware service on the POS side, responsible for:

- Polling: Periodically querying ERP endpoints for updates (e.g., new products or price changes) at configurable intervals (default: every minute).
- Pushing: Sending accumulated data (e.g., sales transactions) to ERP endpoints in batches at defined intervals (default: every minute for transactions, end-of-shift for summaries).
- Queueing: Buffering data in case of network issues, with retry mechanisms (exponential backoff).
- Endpoints: Data is pushed/pulled via RESTful APIs on the ERP side. Example endpoints:
  - POST `/erp/api/receive-pos-data` (for POS to ERP messages).
  - GET `/erp/api/get-updates?storeId={storeId}&lastSync={timestamp}` (for ERP to POS polling).
- Transport Protocol: HTTPS with TLS 1.2+ for security. Authentication via API key in XML
- Authentication: HTTP Bearer Authentication via the `Authorization` header on every request.

  Format:

  ```
  Authorization: Bearer <API-KEY>
  ```

- Intervals Configuration: Defined in Maxtransit setting
  Error Logging: Maxtransit logs failures to a local file (e.g., `/var/log/maxtransit.log`) and sends alerts via email/SMS if configured.

2.2 Data Exchange Format

- XML Structure: All messages use a root `<DATAPACKET>` element with attributes:
  - `command`: Message type (e.g., "ProductUpdate").

    - `version`: Interface version (e.g., "1.0").
    - `timestamp`: ISO 8601 format (e.g., "2024-03-11T14:30:00Z").
    - `sender`: "POS_System" or "ERP_System".
    - `receiver`: Opposite of sender.
- Validation: Use an XML Schema Definition (XSD) for validation (schema provided in Appendix A).
- Compression: Optional GZIP for large batches.
- Batch Support: Messages can contain multiple records (e.g., multiple products in one update).

3. Commands (Message Types)

Below are detailed XML structures for key message types.

3.1 Product Update (ERP to POS)

This message synchronizes product catalogs from ERP to POS, including additions, updates, or deletions. Polled by Maxtransit at intervals.Extended fields for retail:

- EAN: European Article Number (barcode).
- Discount: Percentage or fixed amount (with type indicator).
- Other Important Fields:
    - UPC (Universal Product Code).
    - Brand and Manufacturer.
    - Size, Color, Weight, Dimensions.
    - CostPrice (wholesale cost).
    - MinStockLevel (reorder threshold).
    - ImageURL (for POS display).
    - ActiveStatus (boolean for availability).
    - VariantID (for product variants).

```
xml
<?xml version="1.0" encoding="UTF-8"?>

<DATAPACKET command="ProductUpdate" version="1.0"
timestamp="2024-03-11T14:30:00Z" sender="ERP_System" receiver="POS_System">

  <Products>

    <Product id="PROD001" action="update">

      <SKU>ABC-123</SKU>

      <EAN>4006381333931</EAN>

      <UPC>071662011000</UPC>

      <Name>Wireless Mouse</Name>

      <Description>Ergonomic wireless mouse with 5 buttons.</Description>
```

```
      <Category>Electronics</Category>
      <Size>Medium</Size>
      <Color>Black</Color>
      <Weight unit="g">120</Weight>
      <Price currency="EUR">29.99</Price>
      <CostPrice currency="EUR">15.00</CostPrice>
      <Discount type="percentage">10</Discount>
      <TaxRate>1</TaxRate>      <!-- 1=19% 2=7% 3=0%  -->
      <StockQuantity>150</StockQuantity>
      <MinStockLevel>20</MinStockLevel>
      <SupplierID>SUPP001</SupplierID>
      <ImageURL>https://example.com/images/prod001.jpg</ImageURL>
      <ActiveStatus>true</ActiveStatus>
    </Product>
    <!-- ... -->
  </Products>
  <Status code="200">Success</Status>
</DATAPACKET>
```

3.2 Sales Transaction (POS to ERP)

Pushed by Maxtransit in batches. Reports sales for inventory deduction and accounting.

```
xml
<?xml version="1.0" encoding="UTF-8"?>
<DATAPACKET command="SalesTransaction" version="1.0"
timestamp="2024-03-11T15:45:00Z" sender="POS_System" receiver="ERP_System">
  <SerialNumber>123</SerialNumber>
  <Transactions>
    <Transaction id="12345">
      <StoreID>STORE001</StoreID>
      <Date>2024-03-11</Date>
      <Time>15:30:00</Time>
      <Customer>
```

```
        <ID>CUST001</ID>
        <Name>John Doe</Name>
        <Email>johndoe@example.com</Email>
        <LoyaltyPointsEarned>50</LoyaltyPointsEarned>
      </Customer>
      <Items>
        <Item>
          <ProductID>PROD001</ProductID>
          <Quantity>2</Quantity>
          <UnitPrice>29.99</UnitPrice>
          <Discount>5.00</Discount>
          <Tax>4.80</Tax>
          <Subtotal>55.18</Subtotal>
        </Item>
        <!-- ... -->
      </Items>
      <TotalAmount>65.97</TotalAmount>
      <Payments>
        <Payment>
          <Method>CreditCard</Method>
          <Amount>65.97</Amount>
          <TransactionRef>CC-987654</TransactionRef>
        </Payment>
      </Payments>
    </Transaction>
    <!-- Batch: more transactions possible -->
  </Transactions>
  <Status code="200">Success</Status>
</DATAPACKET>
```

3.3 Inventory Adjustment (Bidirectional)

Handles stock changes like returns or receipts. Can be pushed from POS or polled from ERP.

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<DATAPACKET command="InventoryAdjustment" version="1.0"
timestamp="2024-03-11T16:00:00Z" sender="POS_System" receiver="ERP_System">

  <SerialNumber>45</SerialNumber>

  <Adjustments>

    <Adjustment id="ADJ001">

      <StoreID>STORE001</StoreID>

      <ProductID>PROD001</ProductID>

      <QuantityChange>-1</QuantityChange>

      <Reason>Customer Return</Reason>

      <Date>2024-03-11</Date>

    </Adjustment>

    <!-- ... -->

  </Adjustments>

  <Status code="200">Success</Status>


</DATAPACKET>
```

3.4 Acknowledgment/Response (Bidirectional)

Sent as a reply to confirm processing.

xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<DATAPACKET command="Acknowledgment" version="1.0"
timestamp="2024-03-11T16:05:00Z" sender="ERP_System" receiver="POS_System">

  <ReferenceMessageID>TXN12345</ReferenceMessageID>

  <SerialNumber>123</SerialNumber>  <!-- Echo back the received serial -->


  <Errors>

    <!-- Empty if success; otherwise list issues -->

  </Errors>

</DATAPACKET>
```

3.5 Additional Message Types

- Price Update: Similar to Product Update but focused on pricing changes.
- Store Configuration: ERP to POS for settings like tax rates or promotions.
- End-of-Day Report: POS to ERP summarizing daily sales, cash balances.

4. Error Handling

- Status Codes: Use HTTP-like codes (200: Success, 400: Bad Request, 500: Internal Error).
- Error Elements: Include `<Errors>` with `<Error code="..." description="..."/>`.
- Retries: Maxtransit handles automatic retries for transient errors.

5. Security

- Authentication: HTTP Bearer Authentication via the `Authorization` header.
- Access Control: Role-based (POS can only read updates, not modify ERP core data).
- Auditing: Log all exchanges with timestamps and IPs.
- POS always operates in offline mode with local caching.

6. Testing and Deployment

- Test Scenarios: Simulate polling/pushing with sample XML.
- Deployment Steps:
    1. Install Maxtransit on the POS machine.
    2. Configure intervals and endpoints.
    3. Perform end-to-end tests.