Integration of GNU Taler with ERP systems

Bohdan Potuzhnyi, Vlada Svirsh

Berner
Fachhochschule

2024-2025

Abstract

This report explores the theoretical integration of the GNU Taler Merchant Backend with different ERP systems such as SAP and Dolibarr, aiming to streamline transaction processing, financial management, resource planning, and customer relationship management. GNU Taler — a digital payment system designed for secure and private transactions, is examined in the context of ERP systems like SAP and Dolibarr. The report provides a comprehensive theoretical framework for the integration, including an analysis of system architecture, methodology, and data flow. Emphasis is placed on real-time data synchronization, the automation of manual processes, and the security protocols necessary for ensuring data integrity. While the focus is primarily on the theoretical aspects of integration, this report also outlines potential practical implications for future implementations in various ERP environments, particularly SAP S/4HANA and Dolibarr. The report lays the groundwork for further development and testing by offering detailed insights into the technical requirements and challenges of such an integration.

Contents

List of Figures

Terminology

1. **Taler** — A protocol for digital cash.

2. **GNU Taler** — Software supporting the Taler protocol.

3. **Consumer** — The person or company interacting with a business. This includes individual customers and corporate clients buying goods or services. In the context of GNU Taler integration, consumers represent the end users of the payment system.

4. **Merchant** — The business entity that uses the GNU Taler Merchant Backend for processing transactions, selling goods, and issuing refunds.

5. **Goods** — A general term describing the items in inventory, including both physical and digital products, used in sales or refund processes.

6. **ERP** — Enterprise resource planning software system that helps organizations(merchants) streamline their core business processes.

7. **SPAA (Single-Page Administration Application)** — The web-based administration tool for using the GNU Taler Merchant Backend.

8. **CRM** — Customer Relationship Management.

9. **SMEs (Small and Medium-sized Enterprises)** — Businesses with limited resources compared to larger corporations.

1 Introduction

1.1 Background

In recent years, the need for secure and efficient digital payment solutions has increased, particularly as central banks consider digital alternatives to physical cash. The **integration of GNU Taler** with enterprise resource planning (ERP) systems such as **SAP** and **Dolibarr** reflects this evolution, offering privacy-focused, free software payment solutions in modern business environments.

GNU Taler is a free software [1] digital cash system [2] designed to provide privacy for users while offering transparency of merchants. Its unique feature is the use of cryptographic techniques to ensure payment privacy while adhering to anti-money laundering (AML) regulations. Unlike digital currencies such as Bitcoin, Taler prioritizes transactional privacy without relying on distributed ledger technology (DLT) [3].

The **integration of Taler with SAP** highlights the demand for efficient payment systems in large-scale ERP environments. **SAP**, a globally dominant ERP system, provides businesses with comprehensive tools for financial management, supply chain optimization, and customer relationship management (CRM). Integrating GNU Taler within SAP's financial modules offers businesses the potential to automate and secure their financial transactions, enhancing operational efficiency and ensuring compliance with privacy standards. This integration could modernize how large enterprises handle digital payments, while reducing risks associated with traditional banking channels.

On the other hand, **Dolibarr**, as an open-source ERP and CRM system [4] aimed at small to medium-sized enterprises (SMEs), offers a simpler but equally adaptable platform for managing business operations, including billing, payments, and CRM. The addition of GNU Taler to Dolibarr enhances its appeal by incorporating secure digital payments, allowing SMEs to benefit from the same transaction security and privacy features as larger corporations using SAP.

As central banks explore digital currency issuance, such as the proposal discussed by Chaum, Grothoff, and Moser (2021) on **Central Bank Digital Currencies (CBDCs)** [5], the importance of privacy-preserving payment methods like GNU Taler is underscored. Their research outlines how token-based digital currencies, such as those based on GNU Taler, can offer privacy, scalability, and compliance with regulations. This kind of integration mirrors the ongoing global discourse on digital payments and CBDCs, emphasizing how innovative digital cash systems can be used in modern enterprise environments.

By integrating Taler into ERPs, businesses can streamline transaction processing, reduce manual interventions, and enhance data accuracy, while ensuring a scalable solution for future financial operations. This integration aligns with the broader trend of digital currency adoption and provides a theoretical framework for businesses aiming to improve their digital payment infrastructure.

1.2 Problem Statement

In today's digital economy, businesses face a variety of operational challenges when using isolated systems for transaction processing, financial management, and customer relationship management (CRM). Platforms such as GNU Taler, SAP, and Dolibarr provide powerful functionalities individually but present significant inefficiencies and risks when operated as standalone solutions without integration. These issues become particularly pronounced for businesses requiring streamlined financial management, enhanced customer interaction, and real-time synchronization between transactional, financial, and operational data.

1.2.1 Key Challenges

Integrating GNU Taler with ERP systems such as SAP and Dolibarr introduces significant opportunities for businesses to streamline operations and enhance efficiency. However, several challenges must be addressed to realize these benefits. The following points outline the critical challenges to effective integration and highlight their implications:

1. **Data Silos and Fragmentation:** GNU Taler's Merchant Backend, which acts as a service managing inventory, categories, orders, and payments, remains limited in scope when businesses rely on it in isolation. While it can handle basic e-commerce functionalities, for many businesses it is essential to integrate these operations with more powerful ERP systems like SAP and Dolibarr. Without integration, businesses operate in fragmented silos where transaction data from Taler's backend does not automatically synchronize with financial records or customer management in SAP or Dolibarr. This data fragmentation leads to difficulties in acquiring a unified view of business performance, making resource planning and financial forecasting more challenging.

2. **Manual Data Entry, Errors, and Redundancy:** The lack of integration between Taler's Merchant Backend, SAP, and Dolibarr necessitates the manual transfer of data across systems, such as re-entering order and payment details from Taler into SAP for inventory management or Dolibarr for financial reporting. This manual input process introduces human error and redundancy, which can result in mismatched data between platforms, leading to inaccurate reports, incomplete inventory counts, or payment processing issues. Manual data entry not only wastes valuable time but also increases the risk of non-compliance with tax regulations, which require accurate and up-to-date financial records.

3. **Limited Automation and Inefficient Workflows:** Businesses using GNU Taler, SAP, and Dolibarr in isolation lose opportunities for automating key workflows. For example, when a payment is processed through Taler's Merchant Backend, the inventory in SAP or financial ledgers in Dolibarr are not automatically updated. Additionally, tax reporting and financial statements, which are critical for business compliance, must be manually generated by cross-referencing data from

separate systems. By integrating Taler with SAP and Dolibarr, businesses can automate the flow of data from the point of transaction to financial reconciliation, tax reporting, and inventory management, significantly improving operational efficiency and reducing reliance on manual processes.

4. **Inconsistent Financial Management and Reporting:** One of the limitations of using GNU Taler without integration is that it lacks the financial management capabilities of comprehensive ERP systems like SAP. For example, tax reporting, auditing, and detailed financial analytics will require integration with more powerful ERP systems that offer these advanced features. Businesses need access to bank account data for more efficient cash flow management, and integration of it with comprehensive ERP system which helps to achieve proper handling of tax reports, legal requirements, and regulatory compliance.

5. **Security, Compliance, and Governance Risks:** Separate systems increase the difficulty of ensuring consistent security protocols and governance across platforms. For instance, while GNU Taler focuses on secure payment processing, it may not inherently align with SAP's security governance for enterprise-wide financial data. Furthermore, integrating all systems enables better compliance with industry regulations, such as GDPR, and provides clearer audit trails for tax and accounting purposes. Additionally, automated synchronization between systems can contribute that sensitive information, such as payment details and tax reports, is uniformly managed under strict governance policies, reducing the likelihood of security breaches or data leaks.

6. **Delayed and Fragmented Decision-Making:** The inability to integrate GNU Taler's Merchant Backend with SAP or Dolibarr causes delays in decision-making, as managers lack real-time visibility into financial data, customer interactions, and inventory status. This delay can lead to missed business opportunities and slower responses to market changes. An integrated system would allow businesses to see financial transactions, customer orders, and inventory levels in real time, improving decision-making capabilities. For example, when payments are processed through Taler, integration would ensure that this data is reflected in SAP's financial module and Dolibarr's CRM for more accurate, up-to-date insights.

1.3 Criteria of Working System

To fully leverage the benefits of integrating GNU Taler with ERP systems like SAP and Dolibarr, the solution must meet several key criteria. These elements ensure the system is efficient, reliable, and valuable to businesses:

- **Seamless Integration of GNU Taler's Merchant Backend with ERP:** The primary objective is to create an integrated ecosystem where GNU Taler's Merchant Backend, SAP, and Dolibarr

communicate seamlessly. This will enable businesses to utilize the privacy and security features of Taler for transaction processing while benefiting from the extensive financial management, customer relationship management (CRM), and enterprise resource planning (ERP) capabilities offered by SAP and Dolibarr. Integrating these systems will streamline workflows, reduce operational complexity, and allow for smoother data exchange between payment processing, inventory management, and financial accounting.

• **Real-Time Data Synchronization for Payments, Sales Orders, and Financial Records:** A key goal of this integration is real-time data synchronization between the systems. When payments are processed through the Taler platform, the system should automatically update sales orders, inventory, and financial records in SAP and Dolibarr. This real-time synchronization ensures accurate financial management, providing businesses with an up-to-date overview of their transactions, available stock, and customer orders. Moreover, it facilitates timely reporting and compliance with tax regulations, offering significant value to both small and large enterprises.

• **Automation of Manual Processes to Improve Operational Efficiency:** This integration aims to enhance operational efficiency by automating manual processes such as data entry, inventory tracking, and financial reconciliation. GNU Taler's Merchant Backend currently acts as a service managing payments, orders, and inventory, but without integration, much of this data needs to be manually transferred into larger ERP systems like SAP and Dolibarr. Automation will reduce human error, enhance productivity, and enable faster processing times, leading to cost savings and better resource management for businesses.

• **Integration with Bank Account Information and Tax Reporting:** Another objective is to establish a clear path for businesses to access bank account information, automate bank reconciliations, and generate comprehensive tax reports within the integrated system. By integrating GNU Taler's platform with ERP systems that have advanced financial modules, businesses can access vital financial data, conduct efficient cash flow analysis, and ensure proper tax compliance. This objective aims to facilitate smoother financial management for businesses, especially when handling multiple accounts and complying with legal and tax obligations.

1.4 Conclusion

The integration of GNU Taler with ERP systems like SAP and Dolibarr is a significant step forward in modernizing business operations and embracing the evolving digital payment landscape. By combining the privacy and security features of GNU Taler with the powerful financial and resource management tools of ERP platforms, businesses can streamline transactions, reduce manual interventions, and ensure compliance with regulatory standards. As central banks explore digital cash initiatives, solutions like GNU Taler align with the global push for secure and privacy-preserving payment systems, paving the way for more efficient financial ecosystems.

2 Technology Overview of Existing Components

2.1 Overview of Taler Merchant Backend

GNU Taler is a free software digital payment system designed with privacy, security, and simplicity at its core. It is built with **privacy-preserving design** that ensures that consumer remains anonymous, while merchants transactions are still open and clear for tax authorities. The GNU Taler merchant backend is a central component in facilitating these transactions, providing a structured framework for managing orders, payments, and refunds. [3]

2.1.1 Free Software Nature and Architecture

GNU Taler is developed as part of the GNU project for the GNU operating system [2], adhering to the objectives of transparency and security. As free software, it aligns with the principles of the four essential freedoms defined by the Free Software Foundation [1]:

1. Freedom to Run the Program
2. Freedom to Study and Modify
3. Freedom to Redistribute Copies
4. Freedom to Distribute Modified Versions (for GNU Project only with copyleft licenses [6])

Its modular design includes several key components, including the **merchant backend**, **exchange**, and **consumer wallet**. The merchant backend serves as the intermediary between the merchant's website and the exchange (which processes payments), abstracting the complexity of network communications and cryptographic operations.

The architecture of Taler consists of multiple distinct processes that handle various aspects of digital transactions:

1. **Merchant Backend**: This handles the core functions such as creating orders, managing payments, and verifying consumer transactions, as well as running the GNU Taler merchant single-page administration application.

2. **Exchange**: The exchange manages the issuance and redemption of digital coins. It is responsible for handling consumer payments in predefined currency, as well as managing refunds and auditing functions. This service is typically created and maintained by local authorities or financial institutions such as banks.

3. **Wallet**: The wallet is a browser extension or app that consumers use to hold digital coins and interact with merchants or other consumers (e.g. for the peer-to-peer payments).

The whole system ensures that merchants can accept payments without learning the identities of the consumers, thus maintaining the anonymity promised by the system. Merchants only interact with **order** and **contract terms**, which include the payment details and resources being purchased.

2.1.2 Handling Digital Transactions

The GNU Taler merchant backend handles digital transactions by abstracting the complexities of **cryptographic operations** involved in Taler. When a consumer makes a purchase, the backend facilitates:

1. **Order Creation**: The merchant frontend creates a digital contract with the necessary details, such as product information and the amount to be paid.

2. **Payment Processing**: The backend interacts with the consumer's wallet, which submits the payment information in the form of **deposit permissions**. The merchant backend ensures that these permissions are cryptographically valid.

3. **Order Fulfillment**: Once the payment is confirmed, the backend gives confirmation of the payment, and business can serve the appropriate resource (such as a digital product or confirmation page) to the consumer.

The backend operates through a RESTful API, making it easy to integrate with various e-commerce platforms or custom-built websites. Merchants can use this API to track payment statuses, view transaction histories, and handle refunds seamlessly. This makes Taler a flexible and powerful solution for digital businesses that require a secure and privacy-respecting payment system.

In summary, GNU Taler's merchant backend is a robust system that streamlines the process of handling digital transactions while ensuring anonymity for consumers and transparency for merchants. The free software nature of the project allows developers and businesses to adapt the system to their specific needs while adhering to strong privacy and security standards.

2.1.3 Business Tools of the Taler Merchant Backend

The Taler merchant backend offers a robust suite of tools designed to facilitate secure and efficient e-commerce transactions while adhering to privacy and regulatory frameworks. The system's architecture supports various business functionalities, enabling merchants to manage financial operations, maintain security, and ensure compliance with industry standards. [7]

1. **Order Management System**: The backend provides comprehensive order management tools. Merchants can generate, track, and manage consumer orders via API integrations with their e-commerce platforms or through manual creation within the backend itself. Once an order is

created, it transitions into a contract once payment is received. The backend also supports automatic reconciliation of orders with bank settlements, which streamlines operational workflows.

2. **Inventory Control and Management**: A key feature of the Taler Merchant backend is inventory management. This feature is particularly useful for businesses selling physical goods or limited digital products. Merchants can define stock levels, and the backend ensures that available inventory is accurately reflected in consumer orders, preventing overselling and ensuring that order fulfillment is aligned with stock availability.

3. **Payment and Transaction Security**: Taler's payment system is built on strong cryptographic principles, ensuring both the privacy of consumers and the accountability of merchants. The backend manages all payment processes securely, handling sensitive data like signing keys and banking information internally. The payment process is designed to be compatible with established banking systems, allowing merchants to receive funds in traditional currencies, such as EUR or USD, directly into their bank accounts.

4. **Refund Processing**: The system includes support for processing refunds. If an order cannot be fulfilled, merchants can issue refunds to consumers. However, refunds are only possible before the funds have been fully transferred from the Taler exchange to the merchant's account, adding a layer of protection for exchange.

5. **Automated Financial Settlements**: By leveraging GNU Taler merchant backend APIs, the backend automates the process of settlement reconciliation. Incoming wire transfers from exchanges are automatically matched with the corresponding orders. This reduces the need for manual intervention, enhances accuracy, and ensures that financial records remain consistent and compliant with auditing standards.

6. **Customization and Reporting Capabilities**: The Taler Merchant backend offers significant flexibility, allowing businesses to customize contracts, payment terms, and reporting structures. Merchants can adjust key parameters such as payment deadlines, currencies accepted, and legal conditions (e.g., terms of service and privacy policies). The system also supports detailed financial reporting, making it easier for merchants to comply with regulatory requirements, such as GDPR, anti-money laundering (AML), and know-your-customer (KYC) regulations.

7. **Webhook Integration for Business Automation**: The Taler Merchant backend is equipped with webhook capabilities, enabling businesses to automate processes such as updating inventory, notifying consumers, or triggering fulfillment workflows. Webhooks can be configured to respond to specific events like payment completions or refunds, providing seamless integration with external business systems and improving operational efficiency.

While providing overall good base for the businesses to operate, the Taler Merchant Backend lacks the comprehensive tools that is present in more detailed ERP systems like SAP and Dolibarr. This is

where the integration of Taler with these systems can provide a significant value to the businesses, allowing them to leverage the privacy and security features of Taler while benefiting from the advanced financial management, resource planning, and consumer relationship management capabilities of advanced ERP systems.

2.1.4 GNU Taler API & Webhooks

Integrating GNU Taler with an ERP system requires developers to work with two primary components: the **GNU Taler API** and the **Taler Webhooks System**. These components form the foundation of the communication framework between the GNU Taler Merchant Backend and the ERP system, ensuring seamless synchronization of key business operations such as order management, payment processing, and inventory updates.

2.1.4.1 GNU Taler API   The GNU Taler API provides a comprehensive set of endpoints to manage merchant backend operations. These endpoints enable developers to:

- Manage product categories (e.g., create, update, delete).
- Handle inventory updates, including adding, editing, or deleting products.
- Process orders, including creating orders, checking payment status, and issuing refunds.
- Configure point-of-sale (PoS) settings.
- Query and manage wire transfers.

Developers can reference the detailed API documentation on the official website [8] for a complete list of available endpoints and their descriptions. Also as part of this project we have created the filtered list of APIs to the one that most developers would need in this integration. It is available in 2 forms:

1. OpenAPI 3.0 [9]
2. Postman collection [10]

For instance:

- To create a new order, developers can use the endpoint `POST /instances/{INSTANCE}/`**`private`**`/orders` and provide order details in the request body.
- To issue a refund, the endpoint `POST /instances/{INSTANCE}/`**`private`**`/orders/{ORDER_ID}/refund` can be used, specifying the refund amount and reason.

2.1.4.2 GNU Taler Webhooks System   The Taler Webhooks System facilitates event-driven communication by notifying external systems, such as the ERP, when specific events occur. Key features of the webhook system include:

- **Event Types**: Supports various events, such as `order_pay` (payment received), `order_refund` (refund issued), and `inventory_updated` (inventory changes). See Appendix A for a full list of event types and their triggers.
- **Retry Mechanism**: Webhooks are designed to automatically retry requests with exponential backoff in case the target server is temporarily unavailable. This ensures reliable delivery even during downtime.
- **Customization**: Webhook payloads can be customized using Mustache templates to include event-specific data fields such as `order_id`, `refund_amount`, or `product_serial`.

Developers can configure webhooks using endpoints such as:

- `POST /instances/{INSTANCE}/`**`private`**`/webhooks` to create a new webhook.
- `GET /instances/{INSTANCE}/`**`private`**`/webhooks` to inspect existing webhooks.
- `PATCH /instances/{INSTANCE}/`**`private`**`/webhooks/{WEBHOOK_ID}` to update a webhook.
- `DELETE /instances/{INSTANCE}/`**`private`**`/webhooks/{WEBHOOK_ID}` to delete a webhook.

For example, to receive notifications about completed payments, a webhook can be configured for the `order_pay` event. The webhook will send the `order_id` and `contract_terms` to the specified target URL whenever a payment is completed.

2.2 Overview of SAP

SAP (Systems, Applications, and Products in Data Processing) is a globally leading enterprise resource planning (ERP) software designed to support business operations across multiple functions, including finance, supply chain management, production, and human resources. SAP ERP integrates various business processes into a unified system, enabling departments within an organization to collaborate efficiently and share data seamlessly [11].

2.2.1 History and Evolution of SAP

Systemanalyse Programmentwicklung, founded in 1972, started developing ERP software for companies, and in 1973, 'SAP R/1' was born [12]. Over the following years, SAP evolved into one of the most comprehensive ERP systems globally, capable of managing complex business operations across multiple industries [13]. Its flagship product, SAP ERP, has undergone several iterations, with SAP S/4HANA as the latest version. S/4HANA represents a next-generation ERP platform, leveraging in-memory computing through the SAP HANA database to enable real-time data processing and analytics [14]."

This modern iteration integrates advanced technologies such as machine learning, artificial intelligence, and predictive analytics, allowing businesses to make more informed decisions, reduce operational costs, and enhance overall efficiency [15]. SAP S/4HANA's architecture is particularly suited for real-time data processing, making it highly compatible with digital payment systems like GNU Taler.

2.2.2 Key Modules in SAP

SAP ERP is composed of modular components, each designed to address specific business needs. Several key modules are particularly relevant to integrating with GNU Taler:

- **Financial Accounting (FI):** This module manages all financial transactions within a business, including general ledger accounting, accounts payable, accounts receivable, and tax accounting [13]. It would be instrumental in recording and processing transactions made through GNU Taler.

- **Controlling (CO):** This module supports internal financial reporting and cost management. It is closely integrated with the FI module and provides functionality for budget planning and variance analysis [11]. Integration with GNU Taler could provide real-time insights into cost management based on payment data.

- **Sales and Distribution (SD):** Responsible for handling sales orders, pricing, shipping, and billing, the SD module would play a crucial role in integrating GNU Taler by managing consumers payments and synchronizing sales orders with financial data [16].

- **Materials Management (MM):** This module deals with procurement and inventory control, ensuring that stock levels are maintained based on sales orders [14]. Integrating this module with GNU Taler could automate inventory updates in real-time based on consumer transactions.

- **Customer Relationship Management (CRM):** CRM helps businesses manage consumer interactions and sales processes. Integrating GNU Taler with CRM would enhance consumer experience by synchronizing payment data with consumer profiles, streamlining support and sales processes [11].

2.2.3 Integration Capabilities of SAP

SAP offers robust integration capabilities, allowing external systems to interact with its modules. The two most common methods for integration are:

- **BAPIs (Business Application Programming Interfaces):** These predefined functions in SAP allow external applications to access business objects and processes. BAPIs provide a consistent method for integrating SAP with third-party systems like GNU Taler, ensuring stable and secure interactions [17].

- **IDocs (Intermediate Documents):** IDocs are standard data structures in SAP used for trans-ferring data between systems. They support asynchronous communication and can be used to transmit payment data between GNU Taler and SAP ERP [18]. This is particularly useful for ensuring that financial transactions and sales orders are recorded accurately and in real time.

2.2.4 Benefits of using SAP

SAP is a powerful ERP system that helps businesses streamline their operations and stay competitive. Some of its key benefits include:

- **Automated Transaction Processing:** SAP simplifies complex workflows, such as financial management and sales order handling, saving time, reducing errors, and boosting productivity [19].

- **Real-Time Insights:** With the advanced S/4HANA platform, SAP delivers real-time analytics, enabling businesses to make faster, smarter decisions and adapt quickly to changes in the market [15].

- **Better Financial Management:** The financial tools in SAP, like the Financial Accounting (FI) and Controlling (CO) modules, make it easier for businesses to manage budgets, ensure compliance, and generate accurate reports [20].

- **Strong Security and Compliance:** SAP provides built-in features to meet international regulations, like GDPR and AML, while protecting sensitive data and ensuring secure business operations [13].

2.3 Overview of Dolibarr

Dolibarr is a comprehensive, open-source ERP and CRM system, primarily designed for small and medium-sized enterprises (SMEs). Known for its modular architecture and user-friendly interface, Dolibarr allows businesses to streamline their operations by integrating essential functions such as billing, payments, inventory management, and customer relationship management into a single platform. Its open-source nature makes it highly customizable, enabling businesses to adapt the system to their specific needs without incurring high licensing costs [21].

2.3.1 Key Features and Modules in Dolibarr

Dolibarr offers a range of core modules, each designed to manage specific business functions. The following modules are particularly important for enterprises looking to handle billing, payments, and customer management:

- **Billing and Invoicing**: Dolibarr provides a robust invoicing system that enables businesses to generate, send, and track invoices. The module also handles recurring invoices and offers comprehensive reporting features. The integration with payment gateways allows businesses to track paid and unpaid invoices and manage overdue payments more efficiently [22].

- **Payments**: This module supports the processing of payments from multiple sources, including credit cards, bank transfers, and online payment systems. By automating the reconciliation process, Dolibarr helps reduce manual data entry errors and ensures that financial transactions are recorded accurately. Additionally, the system supports the management of partial payments and refunds [23].

- **Customer Relationship Management (CRM)**: Dolibarr's CRM module enables businesses to manage their interactions with current and prospective consumers. The module includes tools for tracking consumer communications, managing sales pipelines, and handling after-sales support. Integration with the billing and payments modules allows consumer data to be updated automatically when financial transactions occur, improving overall consumer experience and operational efficiency [24].

2.3.2 Integration Capabilities of Dolibarr

Dolibarr offers strong integration capabilities with other systems through APIs and webhooks. Its REST API enables seamless communication between Dolibarr and external applications, allowing for real-time data synchronization across various platforms. This feature is crucial for businesses that need to integrate Dolibarr with e-commerce platforms, payment gateways, or other ERP systems [23]. Additionally, webhooks allow for the automation of workflows by triggering specific actions, such as updating inventory or generating invoices when a payment is processed [22].

Also, one of the key advantages of Dolibarr is its modular design, which allows businesses to activate only the modules they need. This makes it highly scalable, catering to both small companies with minimal needs and larger enterprises that require more complex functionalities. The system is also highly customizable due to its open-source codebase, allowing developers to modify or extend functionalities according to the specific needs of the business [21].

2.3.3 Benefits of Using Dolibarr

- **Cost-Effective**: As an open-source platform, Dolibarr eliminates the high costs associated with licensing fees found in many proprietary ERP systems, making it an affordable option for SMEs [24].

- **Ease of Use**: The user-friendly interface of Dolibarr, combined with its modular structure, makes it easy for businesses to implement and scale the system as their needs evolve [21].

- **Improved Business Efficiency**: By centralizing key business functions such as billing, payments, and CRM, Dolibarr helps businesses reduce redundancy, streamline operations, and improve accuracy in financial and consumer management tasks [23].

Challenges and Considerations

Although Dolibarr offers many advantages, it may not provide all the advanced features required by larger enterprises or highly specialized industries. In such cases, businesses may need to invest in additional development or third-party integrations to extend Dolibarr's capabilities [24].

2.4 Existing Integration Solutions

Currently, there are no existing integration solutions that incorporate GNU Taler into comprehensive ERP systems, particularly in the areas of inventory management, order management, and financial accounting. The primary focus of existing integrations has been on payment processing within e-commerce platforms such as WooCommerce, Pretix, and Joomla!, where GNU Taler serves as a privacy-friendly payment option for online merchants. These integrations offer essential payment processing functionalities, but they do not extend to the broader business operations typically managed by ERP systems, such as resource planning, financial reporting, and supply chain management. [25]

For instance, the integration of GNU Taler with WooCommerce allows merchants to accept Taler payments for online sales, while Pretix and Joomla! integrations provide similar capabilities for event ticketing and content management platforms, respectively. However, these integrations primarily focus on the transaction phase, ensuring secure and private payments through GNU Taler's backend, without directly addressing the need for broader ERP functionalities such as synchronizing payments with inventory levels or automating financial reconciliation with general ledgers.

Additionally, the existing solutions do not provide seamless integration with comprehensive ERP systems like SAP or Dolibarr, which are designed to handle end-to-end business processes. SAP and Dolibarr support extensive modules for managing financial transactions, order fulfillment, customer relationship management (CRM), procurement, and more. The lack of integration with ERP systems limits the potential for businesses to automate workflows beyond payment processing, such as automatic updates to stock levels, order status tracking, or real-time financial reporting based on transactions processed through GNU Taler.

Integrating GNU Taler with ERP systems like SAP or Dolibarr would be a pioneering step, as it would enable businesses to synchronize their digital payment transactions with their overall operations, including supply chain management, sales order fulfillment, and financial accounting. For example,

businesses could use such an integration to automate the process of updating inventory based on sales processed through GNU Taler, while ensuring that payments are reflected in their financial records in real-time.

This type of integration would create a unified business environment where data flows seamlessly between the payment gateway and other business systems, significantly improving operational efficiency, reducing manual data entry, and enhancing the accuracy of financial reporting. It would also extend the value proposition of GNU Taler beyond e-commerce platforms to more complex business environments that rely on ERP systems to manage their operations holistically.

In conclusion, while existing integrations with platforms like WooCommerce, Pretix, and Joomla! serve as valuable starting points for using GNU Taler in e-commerce, they fall short in addressing the broader business needs managed by ERP systems. The broder integration of GNU Taler with SAP and Dolibarr would be a groundbreaking development, opening the door for future integrations with other ERP systems and providing businesses with the tools they need to manage their entire operations more efficiently, securely, and with enhanced privacy.

3 Technical Design of Integration Solution

3.1 Design Overview

The integration of GNU Taler with ERP systems requires updates to the existing system architecture to accommodate new functionalities and ensure seamless interaction between components. This section provides a detailed view of the existing Taler and ERP infrastructure, proposed infrastructure, and the components involved, as represented through component and package diagrams.

Before delving into the subsequent diagrams, packages, and their details, it is important to note that ERP systems vary significantly in their configurations. This report aims to propose solutions that are applicable to the majority of ERP systems. As a result, certain components commonly found in specific ERP setups may be excluded to ensure the proposed solution is easier to understand and to keep this report concise.

3.1.1 High-level Architecture

The high-level architecture of the integration between GNU Taler and ERP can be visualized through the Business Process Flow Diagram (BPFD) depicted in Figure 1. This diagram outlines the interaction between the various systems involved in the payment and order reconciliation process, specifically focusing on the communication between the user, POS, Taler Merchant, Exchange, Bank, and ERP system. Parts of the existing system is highlighted with black and parts that are painted in blue is what we are discussing in this report, and what is supposed to be implemented.
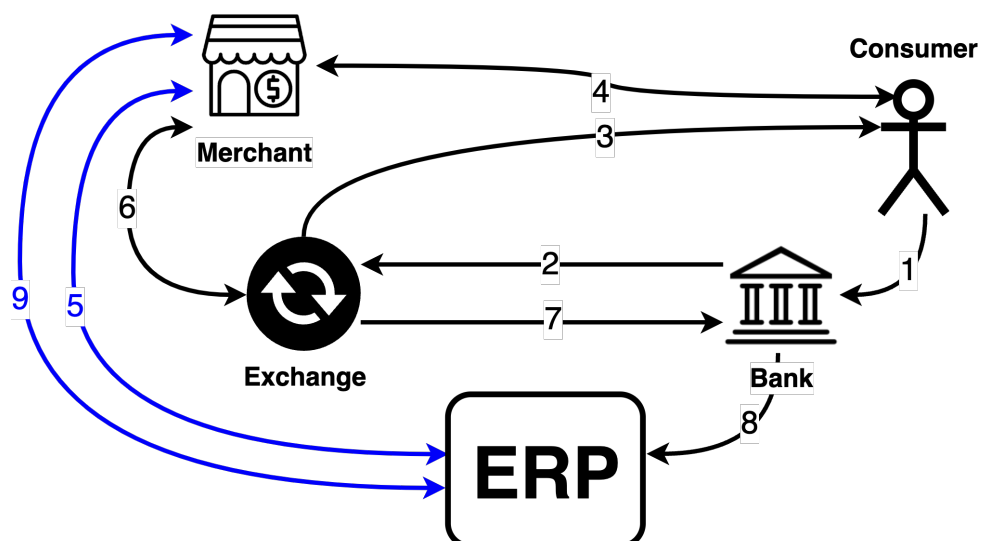
Figure 1: Business Process Flow Diagram: High-level Overview of Typical Sales Order

Step-by-Step Process Description:

1. **User Withdraws Funds to the Exchange Bank Account**:
   The process begins with the user withdrawing funds from their personal bank account into the GNU Taler system. This step ensures the user has sufficient digital funds available to make future purchases.

2. **Bank Processes the Withdrawal and Sends Confirmation to the Exchange**:
   After receiving the user's request, the bank processes the withdrawal and sends a confirmation to the Taler Exchange. This step ensures that the bank confirms the availability of the funds before they are transferred into the GNU Taler digital wallet.

3. **User's Wallet Withdraws the Funds from the Taler Exchange**:
   Upon confirmation from the bank, the Taler Exchange credits the user's GNU Taler wallet with the corresponding funds. These funds are now available for the user to complete transactions with merchants that accept GNU Taler payments.

4. **User Interacts with the Merchant**:
   The user initiates a purchase at the merchant's system. This step involves the selection of products for purchase and the initiation of the payment process via GNU Taler. The Taler Merchant backend generates a unique payment link or request for the user, which the user uses to confirm the payment via the Taler wallet. The Taler Merchant waits for payment confirmation from the GNU Taler system before continuing with the order processing.

5. **Taler Merchant Communicates with ERP**:
   Once the payment is confirmed, the Taler Merchant communicates with the ERP system. This interaction involves notifying ERP of the order's payment status. The ERP system records the financial transaction and updates internal processes such as inventory management, order tracking, and financial accounting.

6. **Taler Merchant Sends the Money to the Exchange**:
   After processing the payment and interacting with the ERP, the Taler Merchant sends the transaction amount back to the Taler Exchange. This step ensures that the transaction moves forward from the Taler system to the banking system.

7. **Taler Exchange Sends Money to the Bank**:
   The Taler Exchange forwards the payment to the bank, facilitating the transfer of funds from the GNU Taler ecosystem to the traditional banking infrastructure. This ensures that the merchant receives their payment in a conventional fiat currency.

8. **Bank Transfers Money to the Merchant's Account**:
   Once the payment is processed by the bank, the funds are transferred to the merchant's bank

account. This completes the financial transaction, ensuring that the merchant is compensated for the sale made through the POS system.

9. **ERP Receives Payment Information and Reconciles the Order**:
   Finally, the ERP system receives a notification from the Taler Merchant regarding the successful payment and reconciles the order. This step ensures that all aspects of the transaction are reflected in the ERP system, including inventory updates, financial records, and order fulfillment status.

3.1.2 Infrastructure and Packaging

Starting with the initial infrastructure of the ERP currently in use. We assume that it likely includes a component responsible for external system connectivity. This component is typically connected to the internal system via an internal interface, as depicted in Figure 2. If this internal interface is well-defined as in case of SAP we might want to utilize it, as usually they provide a better performance, otherwise we have to utilize the external interface of the ERP, if none are present, the possibility of such integration is highly questionable. Additionally, the ERP infrastructure might include components that facilitate communication with banks and databases. However, since the connections to these components, as well as their existence, may differ from one system to another, we will treat them as optional elements that can vary based on specific ERP configurations.
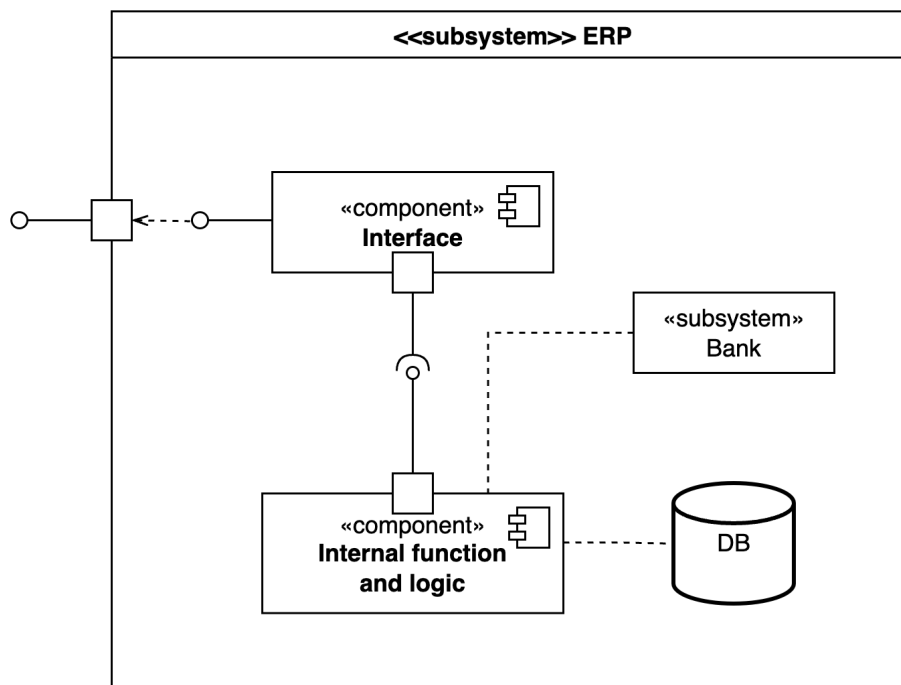
Figure 2: Component Diagram: Initial Simplified Infrastructure of the ERP

Also, we know how the architecture of Taler looks, as shown in Figure 3. The system is composed of two main parts.
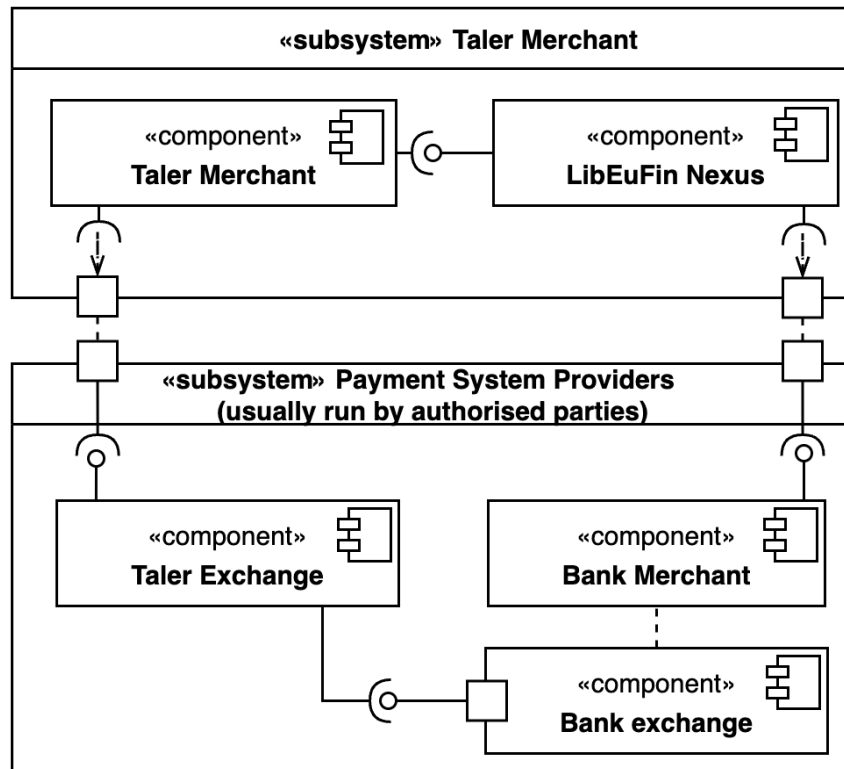
Figure 3: Component Diagram: Initial Taler Infrastructure

These two parts are:

1. **Payment System Provider**:
   This part runs externally and could be managed by authorities, financial institutions, or organizations licensed to issue electronic money. The configuration and controlling parties of this component depend heavily on the regulations in each specific country or its regions.

   - The key module here is the **Taler Exchange**, which handles the main transaction processes in the system.
   - The exchange communicates with its **Bank exchange** to manage interbank transactions with **Bank of merchant**.

2. **Taler Merchant**:
   This part is being run and managed by each specific business.

   - The **Taler Merchant** is the core of this component, handling all merchant-related operations.

- Optionally, the **Taler Merchant** can be configured to include a **LibEuFin** component, which enables communication with the merchant's bank through the EBICS protocol.

This modular architecture ensures flexibility, allowing businesses and financial institutions to configure Taler's components according to their specific operational and regulatory requirements.

As a result of the creation of the new module, we believe that several new components will appear within the ERP subsystem. Specifically, we anticipate on the addition of a **"Marshall"** and an **"Integration Module"** components, as shown in Figure 4 the new communications and components are marked in blue.

The **Marshall** component acts as the primary communication agent between the Taler subsystem and the ERP subsystem. It is responsible for managing and storing requests exchanged between the two platforms. Additionally, it plays a critical role in:

- Balancing the load on the ERP subsystem, which may require extra time to process transactions.
- Managing external calls and ensuring resilience during downtime in the Taler subsystem.
- Accessing the **Database (DB)** to store necessary information for queuing and efficiently processing requests.

The **Integration Module** serves as the main logic processor, handling various integration scenarios described in the following subsections. Positioned internally, this module is assumed to:

- Have direct access to the **Internal Function and Logic** component.
- Access the **Database (DB)** to store temporary states of requests managed by the Marshall component.
- Handle the requests which have been assigned by the **Marshall**, and ensure that all transactions align with ERP workflows, including inventory updates, order reconciliation, and financial reporting.
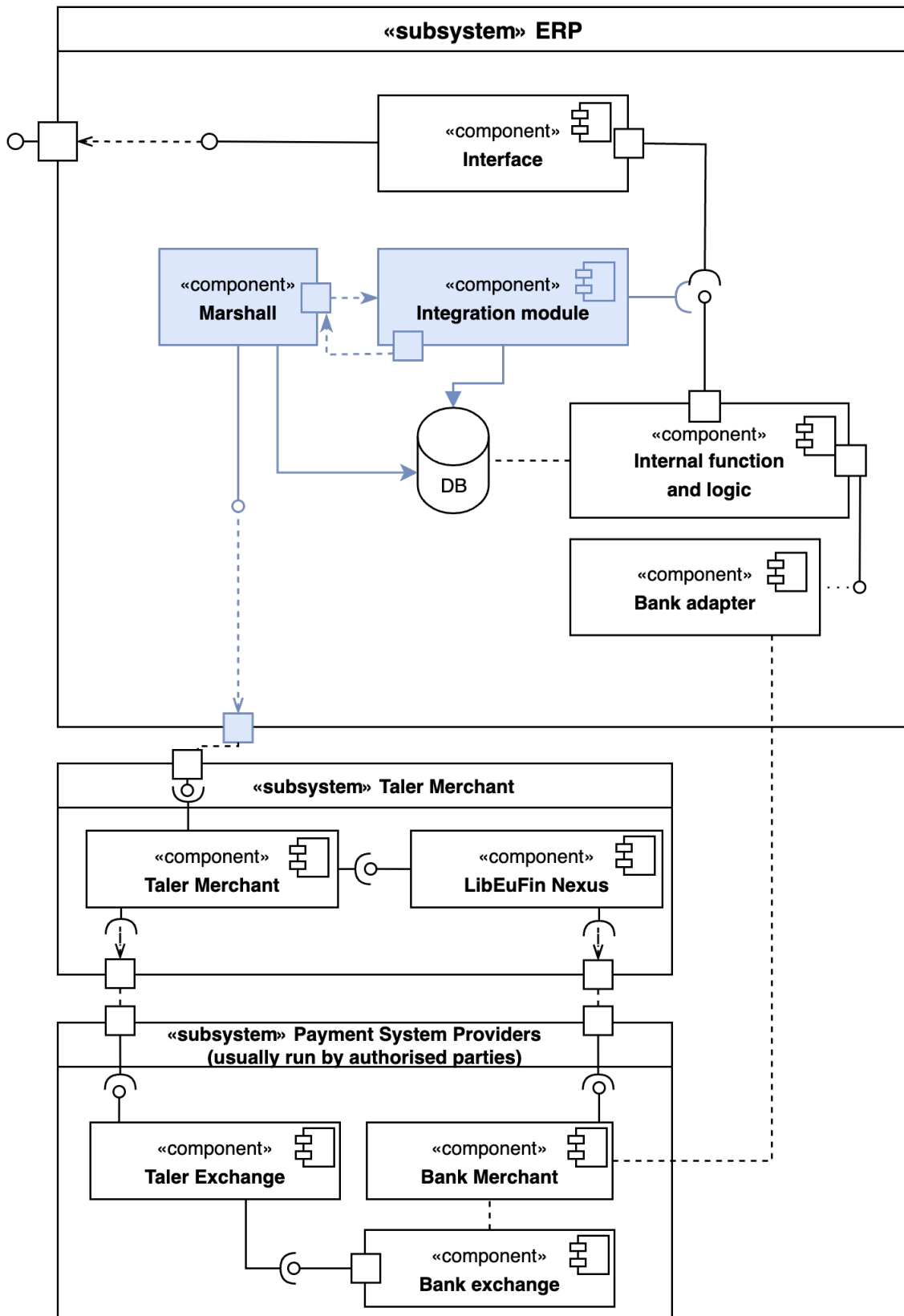
Figure 4: Component Diagram: Proposed Infrastructure of the Integration

To provide additional context, a package diagram is presented in Figure 5. In the ERP package, ideally, only one new package — the **Integration package** — is introduced to interact with the existing system. Integration package will facilitate communication between the old packaging and the Taler system.

The Taler system, in turn, introduces its own package **Taler Merchant** package, which handles merchant-specific logic and single-page administration application (**SPAA**).

The **Taler Merchant** will interact with the integration package in the ERP system, which contains the communication module named **Marshall** and the **Integration module** with necessary logic for integrating with the existing ERP subsystems and workflows.

This modular approach ensures that the integration remains scalable and maintainable, minimizing disruption to existing systems while enabling the addition of GNU Taler functionalities.
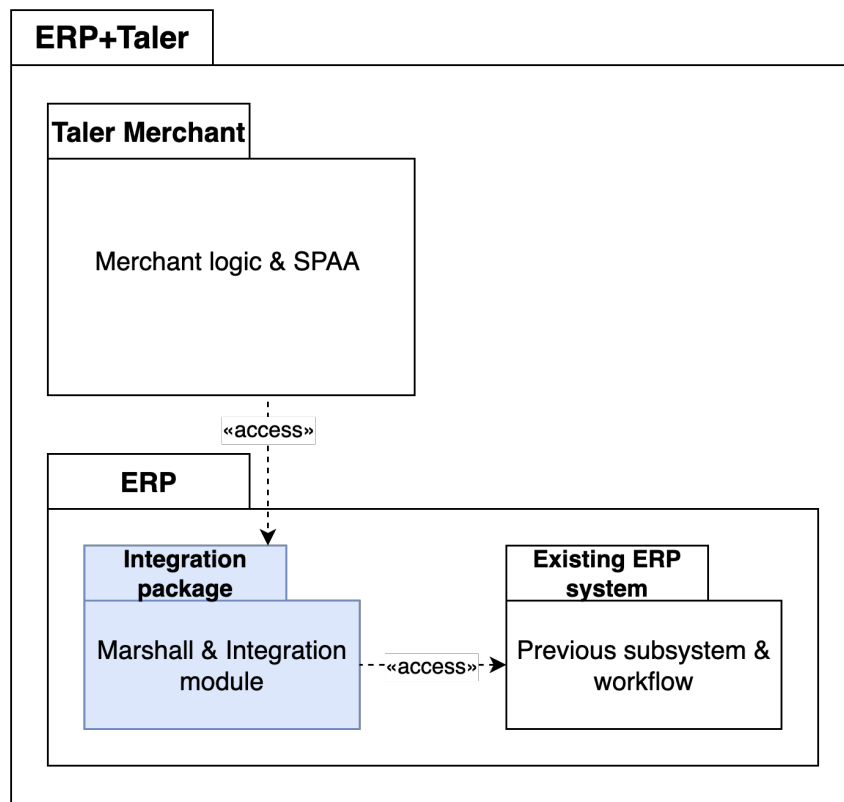
Figure 5: Package diagram: ERP and Taler Integration

3.1.3 Centric Integration Design

The integration of GNU Taler with ERP systems introduces the need for new communication pathways, making it essential to establish how bidirectional synchronization will occur. Referring to Figure 4,

a key question arises: how should data flow between the systems, and which system should act as the primary authority for maintaining consistency? To address these questions, the concept of **centric integration** is introduced, emphasizing two distinct approaches: **Taler-Centric Integration** and **ERP-Centric Integration**.

These approaches define how the integration should be structured, where the focus lies, and how synchronization responsibilities are distributed.

1. **Taler-Centric Integration** This approach is suited for businesses that are either starting from scratch or already use a Taler system and wish to easily migrate the data to ERP with more tools. In this model:

   - The Taler Merchant Backend acts as the primary source of truth, managing all business data, including orders, inventory, and consumer interactions.
   - The ERP system is integrated mainly for financial reporting, tax compliance, and other processes dependent on data from GNU Taler.
   - This configuration is ideal for businesses that prefer to handle the majority of operational processes on GNU Taler while relying on the ERP system for financial and reporting tasks.

2. **ERP-Centric Integration** This approach is ideal for businesses that already have an ERP system (such as SAP or Dolibarr) in place and want to integrate GNU Taler as a payment method. In this model:

   - The ERP system remains the primary source of truth for business data, such as orders, inventory, and consumer information.
   - The ERP system interacts with the Taler Merchant Backend for payment processing and related functionalities.
   - This setup ensures that the core business processes are maintained within the ERP system, while GNU Taler adds new payment method, enhancing the overall user experience.

By categorizing the processes into these two groups, developers can more easily define the appropriate strategy and logic for integration. This approach provides businesses with diverse possibilities, especially when developers implement all available options, enabling flexibility in choosing the integration model that best aligns with their operational needs — whether they prefer their ERP system as the backbone of operations or want to leverage the capabilities of the Taler Merchant Backend for managing transactions and business data.

It is also important to note that some business processes remain consistent across both approaches. So far the only such process is the **payment reconciliation process** will follow the same workflow, ensuring a standardized approach to this operation.

3.2 Taler-Centric Integration

3.2.1 High-level Data Flow

In this approach, the GNU Taler Merchant Backend is used as the source of truth for most business data. This includes:

1. **Order Information**: Managing the creation and modification of orders.
2. **Inventory Updates**: Handling changes to inventory data and reflecting stock availability.
3. **Order and Payment Statuses**: Tracking updates to orders and payment statuses from the Taler payment system.

However, reconciliation of payments is initiated by the ERP system. The results of this reconciliation are then communicated back to the GNU Taler Merchant Backend through updates to the order status.
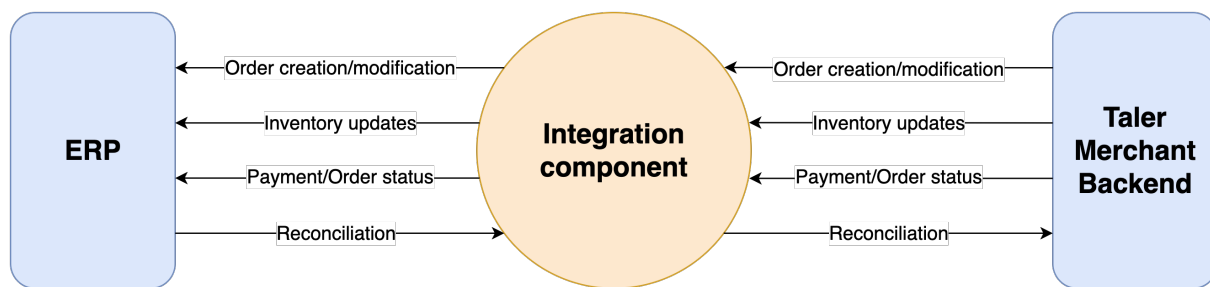
Figure 6: Data Flow Diagram: High-level Communication of Taler as Source of Truth and ERP

3.2.2 Inventory Management Process

We expect that no order can be created without having the necessary goods in inventory, even with the fact that Taler allows such behaviour. Therefore, the first step is to transfer the inventory data into the ERP system. To facilitate this, the **inventory management process** was created. The BPMN diagram illustrating this process is shown in Figure 7.

This process is initiated by a timer, which must be configured by the system administrator. It is designed to retrieve and save the two main components of the inventory: **categories** and **products**. As a result, the process consists of two primary sub-processes:

1. **Update Categories**
2. **Update Products**

This process is fully automated and designed to run without requiring any manual input.
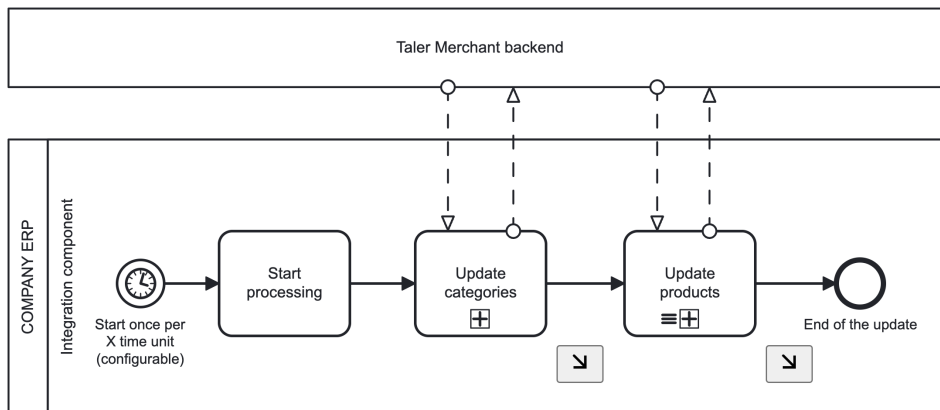
Figure 7: BPMN: Inventory Management Process from Taler

As shown in the diagram, the process includes the sub-process **Update Categories**, which is illustrated in Figure 8. This sub-process retrieves the categories from the Taler Merchant Backend and updates them in the ERP system.
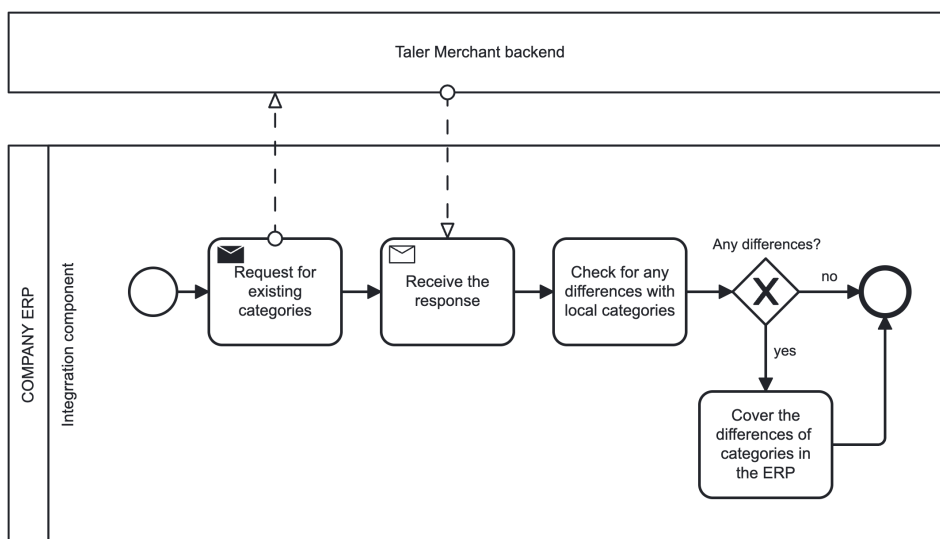
Figure 8: BPMN Sub-process: Update Categories from Taler

Another sub-process shown in Figure 7 is **Update Products** which is shown in Figure 9, which is responsible for preparing the inventory data required for creating orders. **Accuracy** in the update/delete path stands for checking that the information stored in the ERP is the same as received from the Taler, it is important to check in cases, where modifications on Taler side were made in situations such as non-standard sales/refund processes (e.g. products are missing, stolen products, out of term of validity etc.), in such cases updates of these products needs to be additionally handled by personal, to insure correct status of product movements.
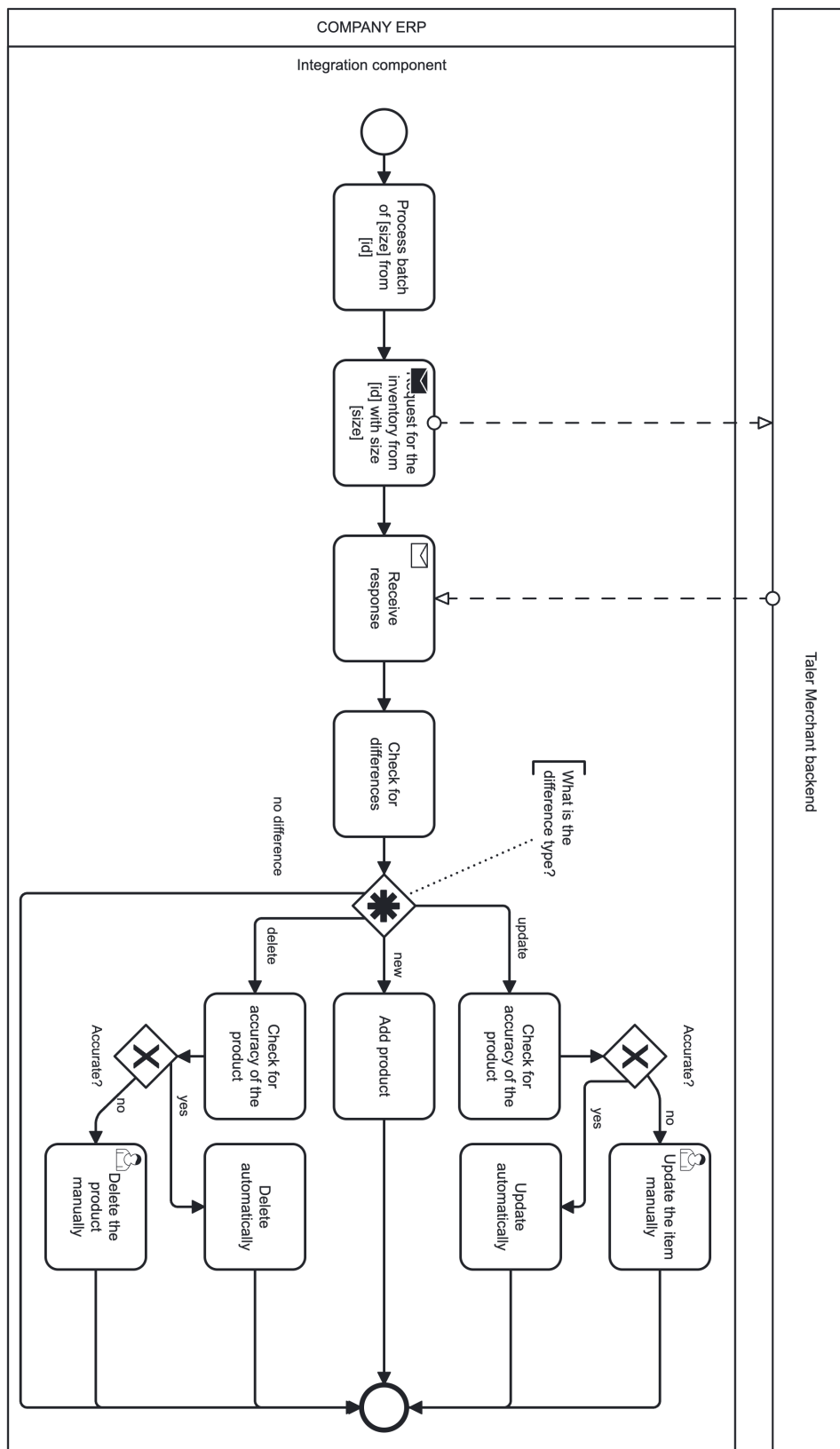
Figure 9: BPMN Sub-process: Update Products from Taler

The **sequence diagram for the inventory management process**, illustrated in Figure 10, outlines the interaction flow between the Taler Merchant Backend and the ERP system. This process ensures synchronization of categories and products in the inventory across both systems. Key steps:

1. **Category Updates:**

    - The ERP system sends a request to the Taler Merchant Backend to retrieve updated category information.
    - The Taler Merchant Backend responds with the latest category data.
    - The ERP system verifies if there are any new categories or updates.
    - If updates are identified, the ERP system synchronizes the new categories; otherwise, no changes are made.

2. **Item Updates:**

    - After category updates, the ERP system requests item details from the Taler Merchant Backend.
    - The Taler Merchant Backend sends item data to the ERP system.
    - The ERP system validates the accuracy of the received data and updates the inventory accordingly.

3. **Final Validation:**

    - Both categories and products are verified for completeness and accuracy.
    - Any discrepancies trigger alerts or logs for manual intervention.

This process ensures that the ERP system and Taler Merchant Backend maintain a consistent view of inventory, enabling smooth business operations.

Figure 10: Sequence Diagram: Inventory Management Process from Taler

3.2.3 Sales Process with Transfer After Order is Created

This process involves transferring the order details immediately after the order is created, even before payment is received. The BPMN diagram in Figure 11 illustrates the proposed integration.

This process is requiring additional round of communication comparing to transferring after payment is received, which makes it more complex to implement. Developers opting for this approach will need to handle potential compensation events if the payment is not completed after the order has been created. Therefore, for a simpler initial implementation, it is recommended to consider the **Sales Process with Transfer After Order is Paid**, which is described in the following section 3.2.4.

Key steps in this process include:

1. **Order Creation and Transfer**:
   The order details are transferred to the ERP system as soon as the order is created. During this time, the system waits for payment to be confirmed. If the payment is not received within the specified timeframe, the process is terminated, and compensation logic is triggered.

2. **Compensation Events**:
   If payment fails or is not completed, the system initiates compensation events to handle the incomplete order. These events could include canceling the order or notifying the relevant stakeholders for manual intervention.

This process provides more immediate visibility into the order details within the ERP system but comes with additional challenges in terms of handling incomplete transactions.
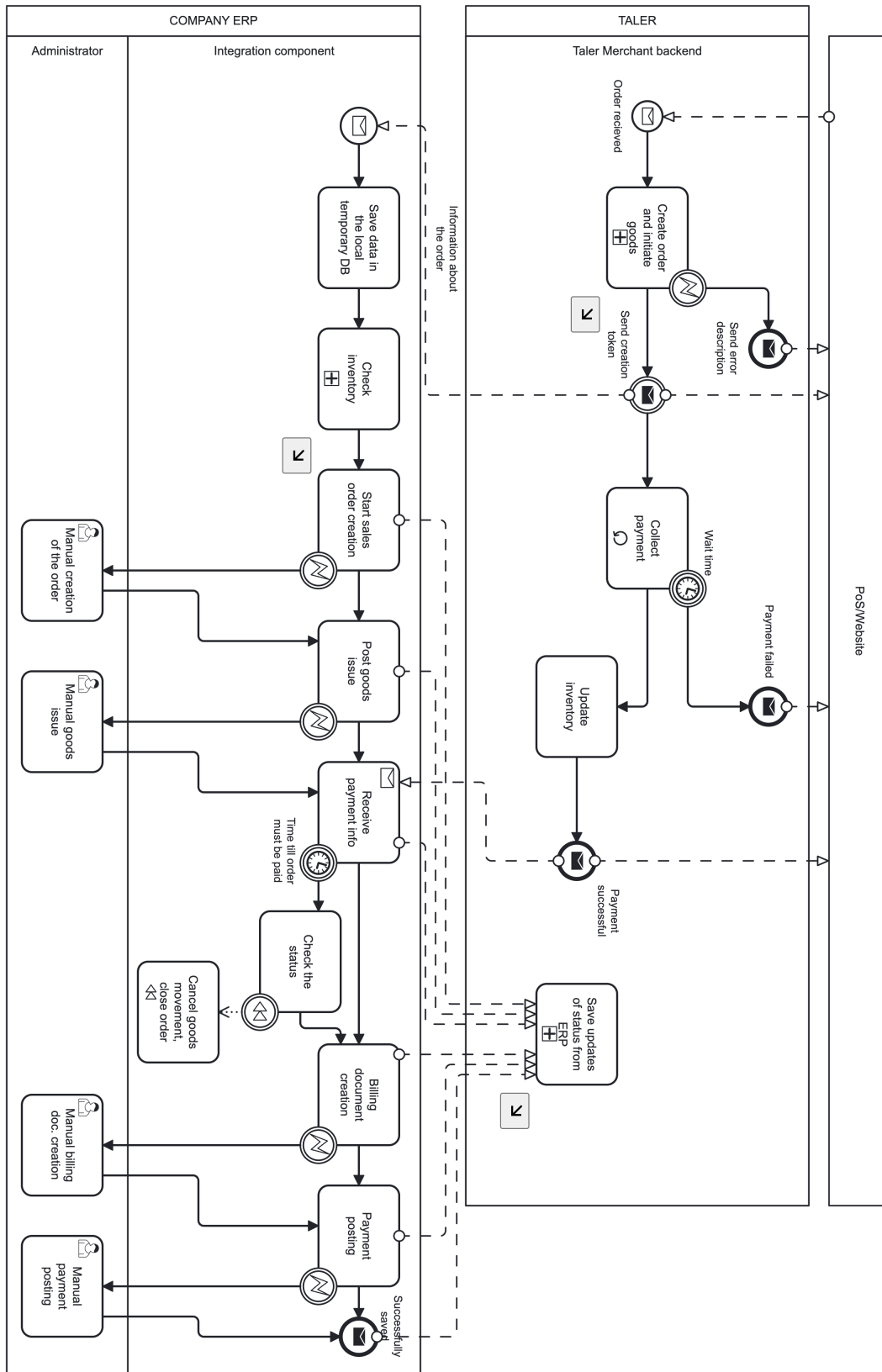
Figure 11: BPMN: Sales Process from Taler on Order Being Created

The **sequence diagram for the sales process from Taler on order being created** is presented first in Figure 12 and Figure 13. This process outlines the interaction flow when the order is created in the system before payment is confirmed. Key highlights of this sequence include:

1. **Order Initialization**:

   - The consumer initiates an order through the Point-of-Sale (PoS) or website.
   - The Taler Merchant Backend receives the order and processes the initial steps.

2. **Inventory Check**:

   - The system validates that all goods in the order are available in the inventory.
   - If some products are missing, a placeholder product is created for these products, or an update is triggered for inventory addition.

3. **Order Creation**:

   - The validated order is officially created in the ERP system.
   - **Goods Issue**: The system ensures the products are marked for delivery or consumption.
   - **Billing Document Creation**: The financial document detailing the transaction is prepared.
   - **Payment Posting**: Although the order has been created, payment reconciliation remains pending.

4. **ERP System Update**:

   - The ERP system logs the order details and updates internal inventory and accounting records.

This process is ideal for businesses that prioritize early visibility into the ERP system, even before payment is finalized. However, additional compensation logic may be required in case payment is not completed within the stipulated time.
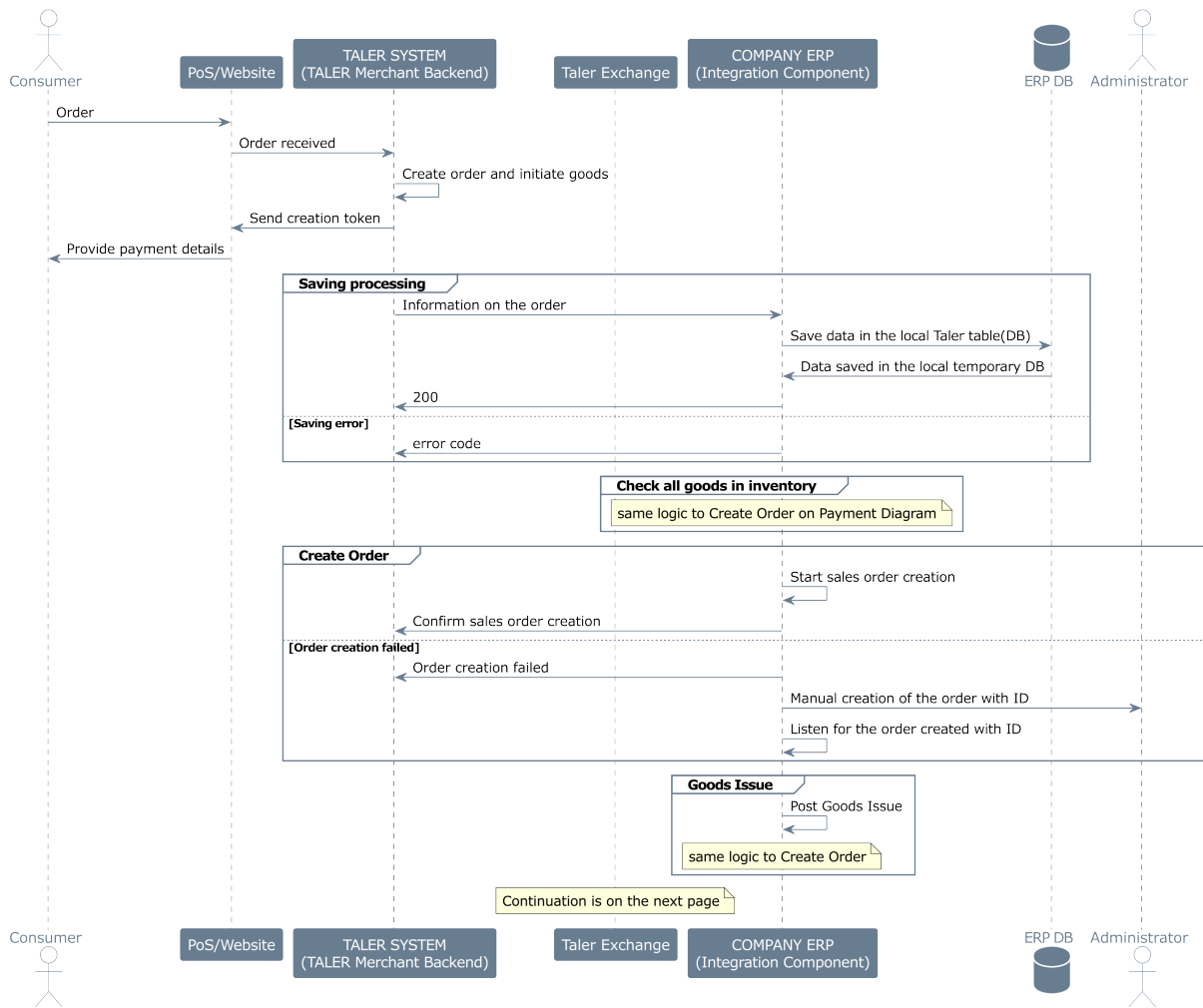
Figure 12: Sequence Diagram: Sales Process from Taler on Order Being Created Part 1

Figure 13: Sequence Diagram: Sales Process from Taler on Order Being Created Part 2

3.2.4 Sales Process with Transfer After Order is Paid

This process is simpler and involves transferring the order details only after payment has been confirmed. In comparison to the previous process this one does not require additional confirmation or cancellation from the GNU Taler Merchant backend system on the matter of the payment status, skipping some compensation and handling logic, making it easier to implement, especially for the first iteration of the integration.

The BPMN diagram in Figure 14 illustrates the integration of the **GNU Taler Merchant Backend** with the company's **ERP system** and highlights the steps involved:

1. **Order Creation and Payment Collection**:
   Once an order is received, the Taler Merchant Backend creates the order and begins goods handling. During this step, the sub-process **Create Order and Initiate Goods Issue** is executed, as shown in Figure 15. The system then waits for payment confirmation with long polling, and waiting up to the payment deadline (this options is not locked and is configurable for each order). Once payment is received, the process proceeds with order approval.

2. **Confirmation and Approval**:
   After payment is confirmed, the backend confirms the order's processing. It then sends info about transaction to the ERP.

3. **Order Handling in the ERP System**:
   The ERP system performs the following backend operations:

   - **Data Saving**: Saves the order in a temporary database.
   - **Inventory Check**: Verifies stock availability.
   - **Sales Order Creation**: Initiates the sales order (may require manual input).
   - **Goods Issue**: Posts goods (may require manual input).
   - **Billing Document Creation**: Generates a billing document (may require manual input).
   - **Payment Posting**: Posts the payment in the ERP system (may require manual input).

Once these steps are completed, the order is successfully saved, and the notification is sent to the Taler Merchant Backend.

Sub-Processes:

- **Create Order and Initiate Goods Issue**:
  This sub-process verifies the correctness of the order and checks for goods availability. It is illustrated in Figure 15.

  - **Check Order**: Ensures the order is valid for the merchant.
  - **Check Inventory**: Confirms goods are available in stock.
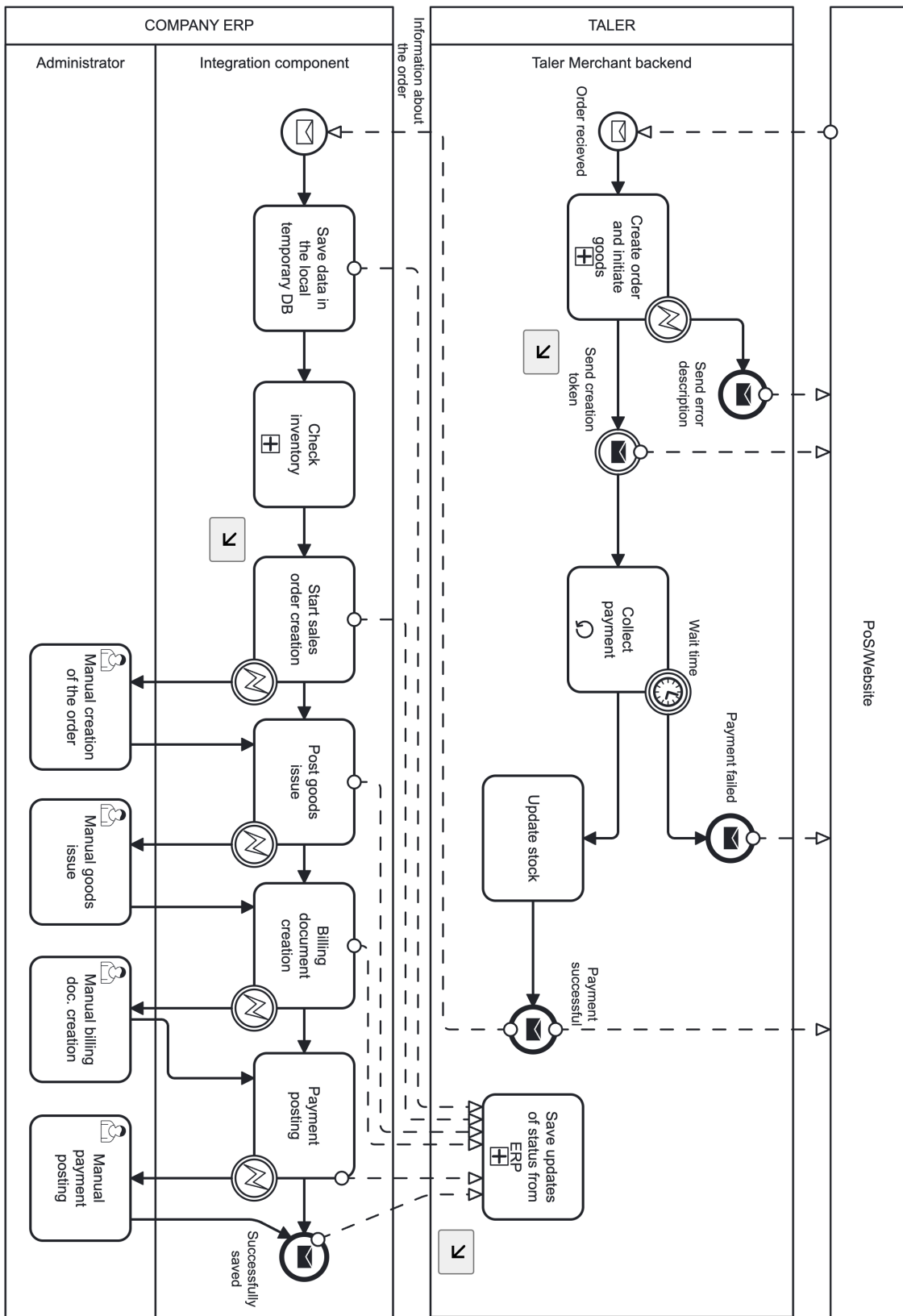
Figure 14: BPMN: Sales Process from Taler on Order Being Paid
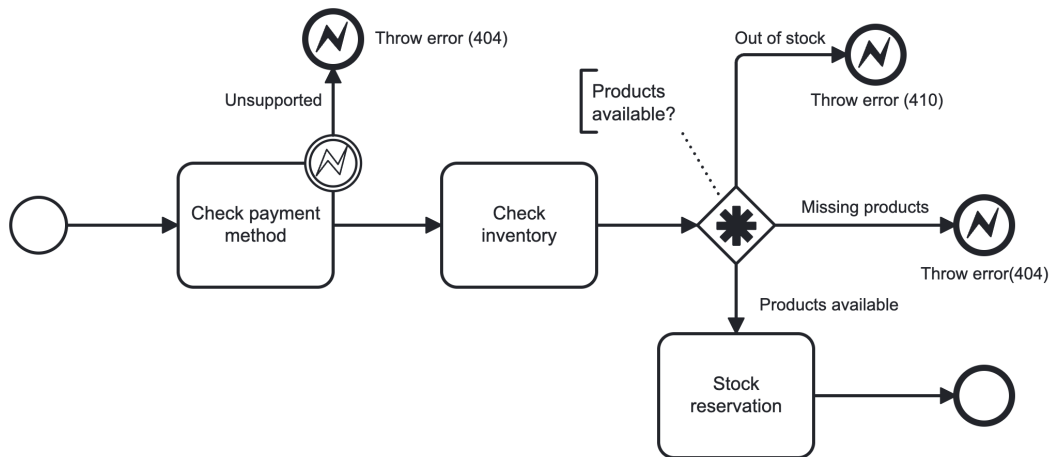
Figure 15: BPMN Sub-process: Create Order and Initiate Goods Issue

- **Inventory Preparation for Order**:
  This sub-process, shown in Figure 16, ensures that the inventory is ready for the order. Steps include:

  1. Checking for products being in stock.
  2. If unavailable, attempting to retrieve products from the Taler Merchant Backend.
  3. If still unavailable, initiating a placeholder item for order creation.
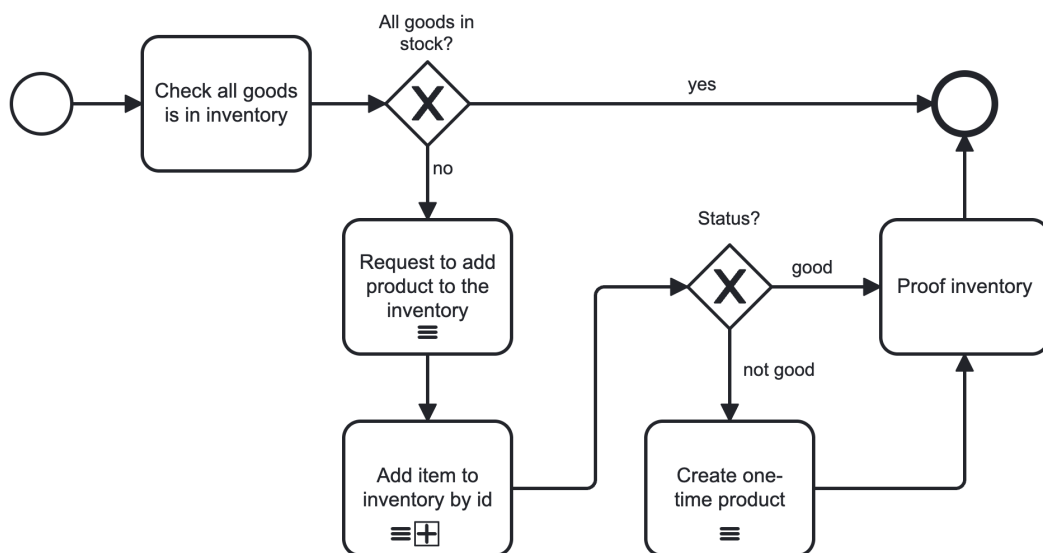
Figure 16: BPMN Sub-process: Check Inventory for Order Creation from Taler

This approach is well-suited for developers looking for a straightforward integration method with minimal complexity.

The **sequence diagram for the sales process from Taler on order being paid** is shown in Figure 17, Figure 18 and Figure 19. This process is more asynchronous, where order details are only synchronized after payment has been successfully processed. Key highlights include:

1. **Payment Confirmation**:

   - The consumer completes the payment through the Taler Wallet.
   - The Taler Exchange confirms the successful payment and notifies the Merchant Backend.

2. **ERP Integration**:

   - The Taler Merchant Backend initiates communication with the ERP system to create and finalize the order.

3. **Order Creation**:

   - The ERP system processes the order creation, ensuring inventory alignment and financial consistency.

   - **Goods Issue**: Products are marked for delivery or consumption within the ERP system.

   - **Billing Document Creation**: The ERP system generates a billing document for financial records.

   - **Payment Posting**: The payment is logged in the ERP system, completing the reconciliation.

4. **Completion**:

   - The ERP system finalizes the order, updates inventory, and reconciles the payment status, ensuring accurate records across all systems.

This process is simpler to implement and eliminates the need for compensation logic, as payment is already confirmed before order synchronization. For readability reasons, this diagram was divided into 2 parts. Consult next 2 pages to get the full picture.
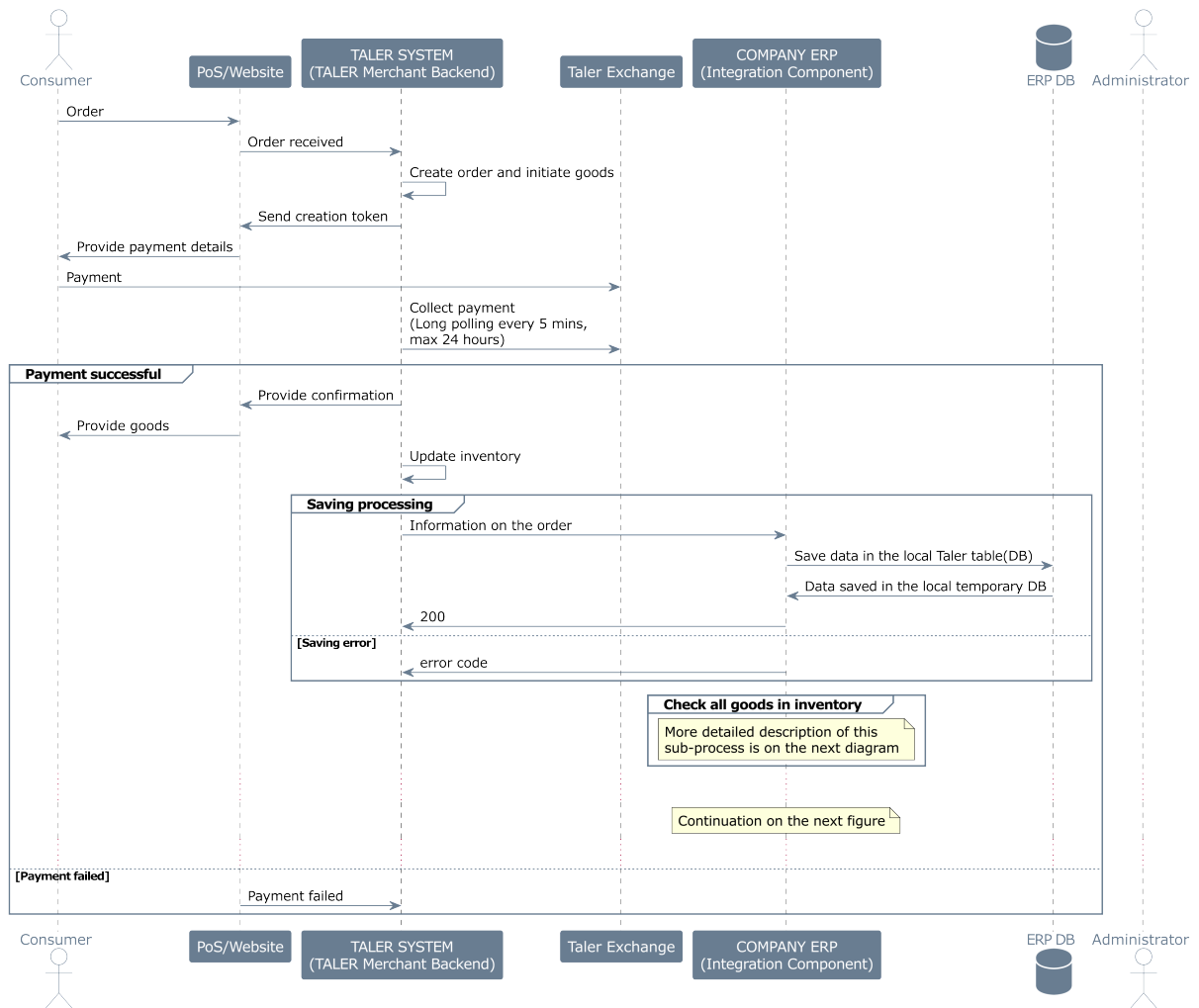
Figure 17: Sequence Diagram: Sales Process from Taler on Order Being Paid Part 1
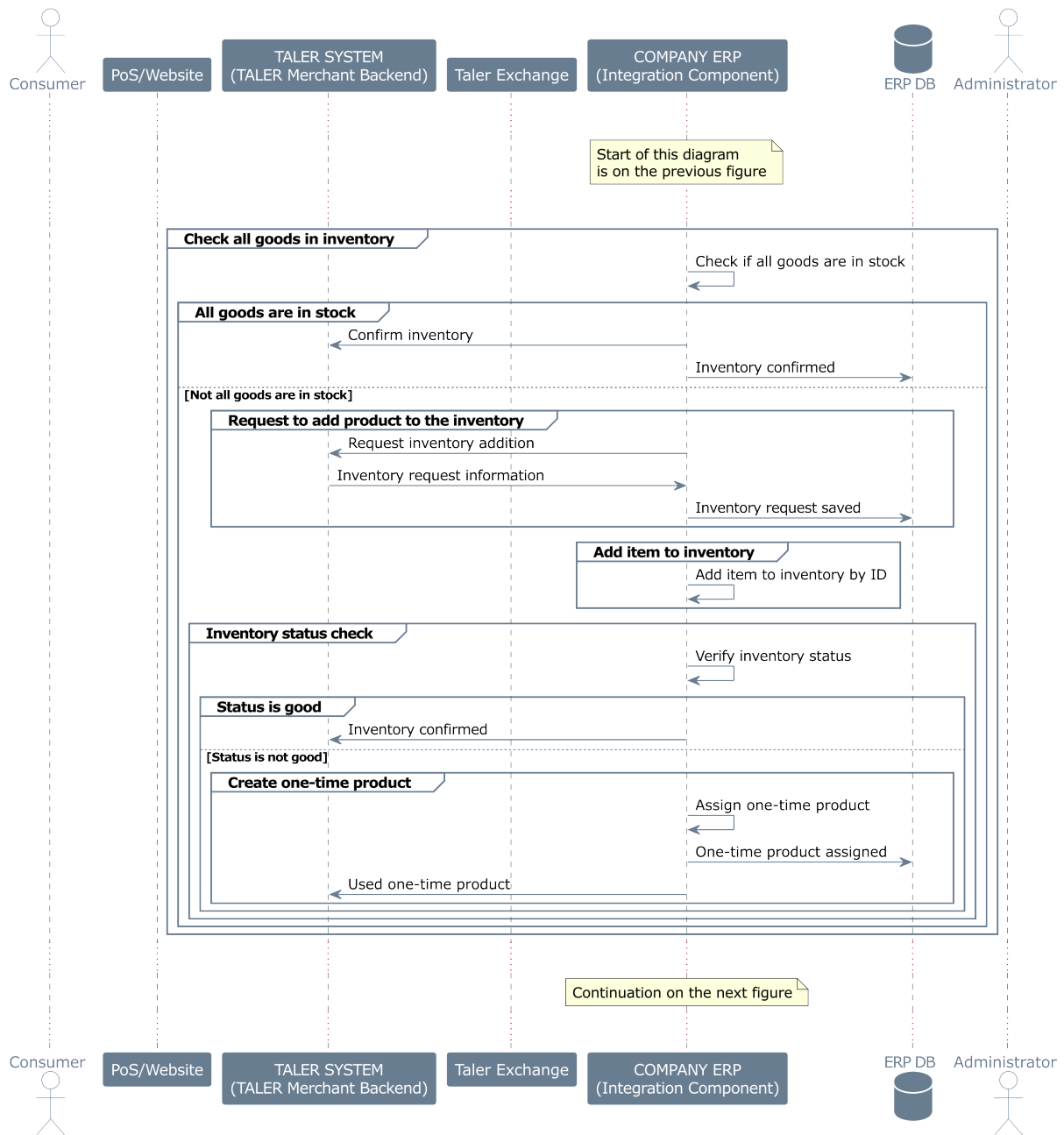
Figure 18: Sequence Diagram: Sales Process from Taler on Order Being Paid Part 1.5
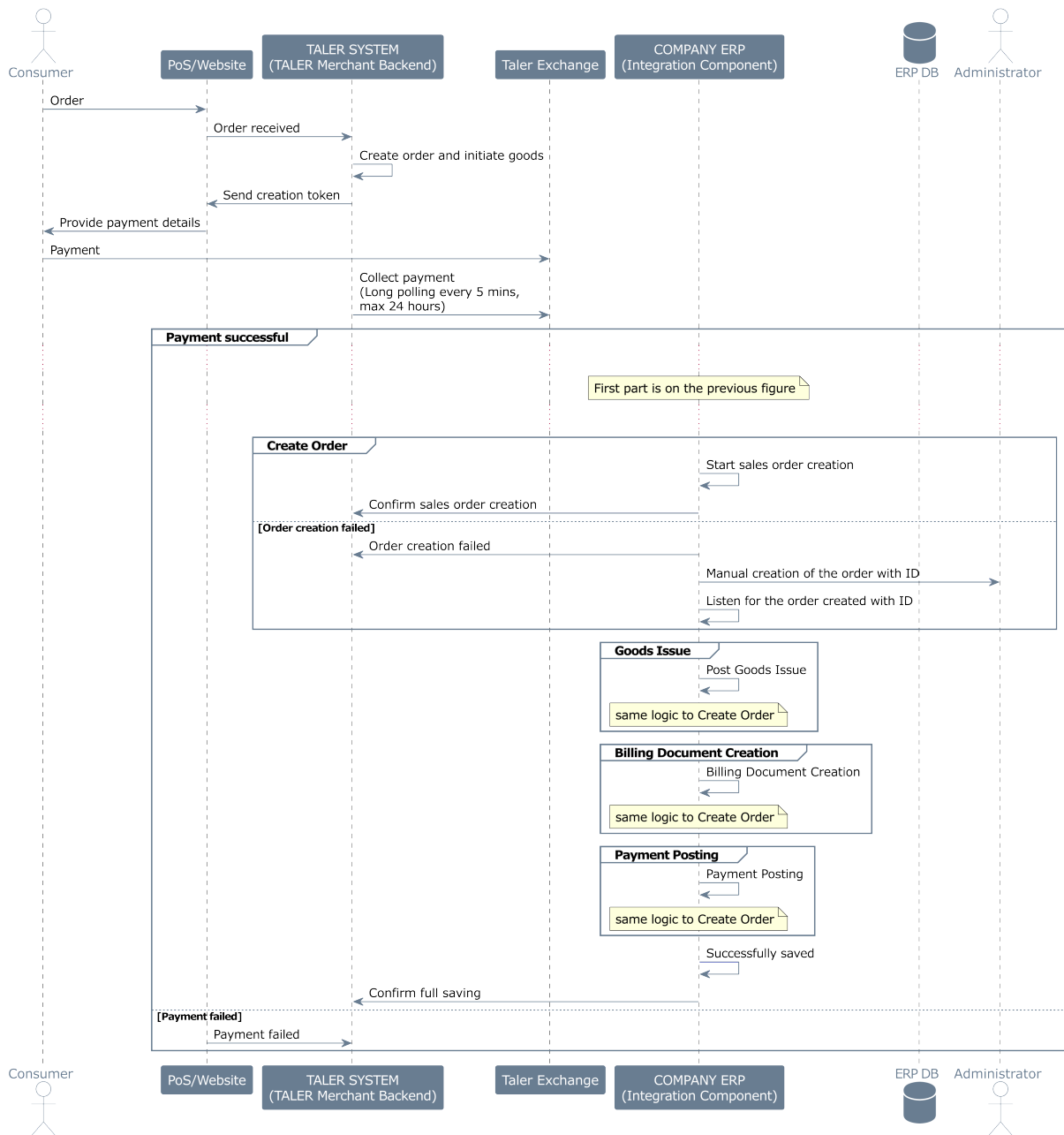
Figure 19: Sequence Diagram: Sales Process from Taler on Order Being Paid Part 2

3.2.5 Refund Process

The refund process is a critical aspect of integrating GNU Taler with the ERP system. The BPMN diagram illustrating this process is presented in Figure 20. The process begins with a refund request from administrator through the user interface, which is afterward managed by the Taler Merchant Backend. Once the Taler Merchant Backend confirms the refund, the process is transferred to the ERP system for further handling.

The steps involved in this process are as follows:

1. **Save Data in the Local Database**:

   The refund request is recorded in a temporary local database to ensure traceability and consistency.

2. **Locate the Order in the ERP System**:

   The system verifies that the order associated with the refund exists within the ERP system.

3. **Initiate Return Order Creation**:

   A return order is created to manage the refund workflow.

4. **Create Billing Document for the Refund**:

   A billing document is generated to document the financial aspects of the refund.

5. **Post the Payment Document to the Refund Order**:

   The payment document is finalized and posted to the refund order, completing the process.

While these steps may vary depending on the specific ERP system being used, they generally encompass the essential actions required for handling refunds effectively.
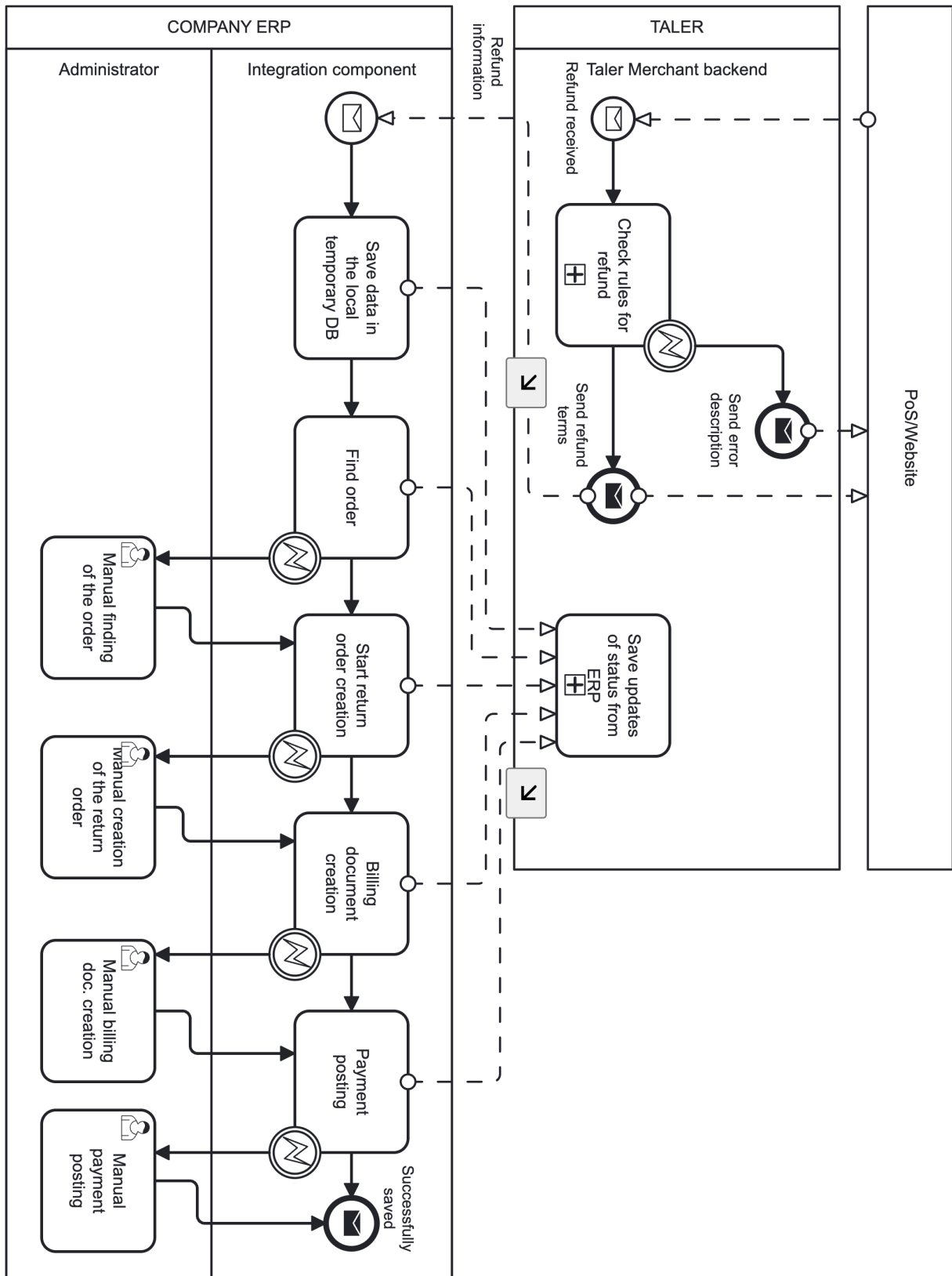
Figure 20: BPMN: Refund Process from Taler

The **sequence diagram for the refund process from Taler**, shown in Figure 21, outlines the key interactions between the Taler system and the ERP system during a refund. This process ensures accurate synchronization and traceability across platforms. Key steps:

1. **Refund Validation:**

   - The user initiates a refund request via the Taler system (e.g., PoS or website).
   - The Taler Merchant Backend verifies if the refund is allowed and confirms its eligibility.

2. **Data Storage:**

   - The refund request is saved in the Taler Merchant Backend's database for traceability and audit purposes.

3. **Refund Processing:**

   - The Taler Merchant Backend communicates with the ERP system to initiate the refund process by creating a return order.

4. **Return Billing Document:**

   - The ERP system generates a return billing document to record the refund transaction.

5. **Return Payment Posting:**

   - The ERP system posts the payment for the refund, completing the financial reconciliation.

6. **Completion:**

   - Both the Taler system and the ERP system update their records, and the consumer is notified of the successful refund.

This streamlined process ensures accurate and efficient refunds across the Taler and ERP systems.
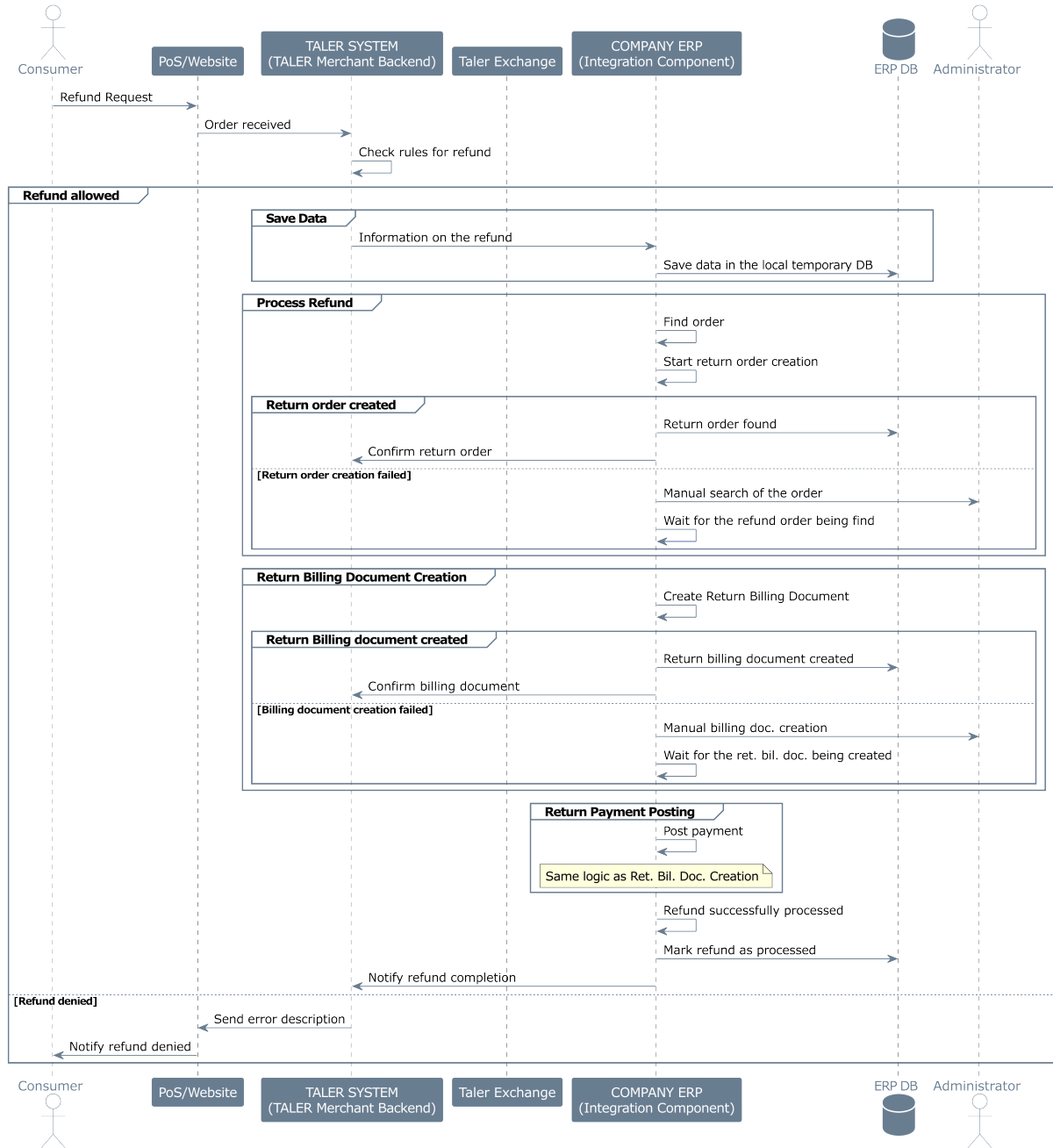
Figure 21: Sequence Diagram: Refund Process from Taler

3.3 ERP-Centric Integration

3.3.1 High-level Data Flow

In this approach, the ERP system is the source of truth for the majority of the business data, except for **transactions data** (which includes updates for the **order/payment status**) received from the GNU Taler Merchant Backend via API responses and the webhook service.

The ERP could provide **inventory information** to Taler to display products on receipts issued to consumers through the Taler payment system.

To process payments within the Taler system, the ERP must handle the **order creation** in the GNU Taler Merchant Backend. Additionally, all **modifications** to orders and the initiation of refund processes are expected to also originate from the ERP.

Similar to the Taler-Centric Integration, reconciliation of bank transfers requires the ERP to first provide the bank transfer data to GNU Taler merchant backend. Subsequently, transaction data is managed on ERP, with requests coming from the GNU Taler merchant backend's webhook system.
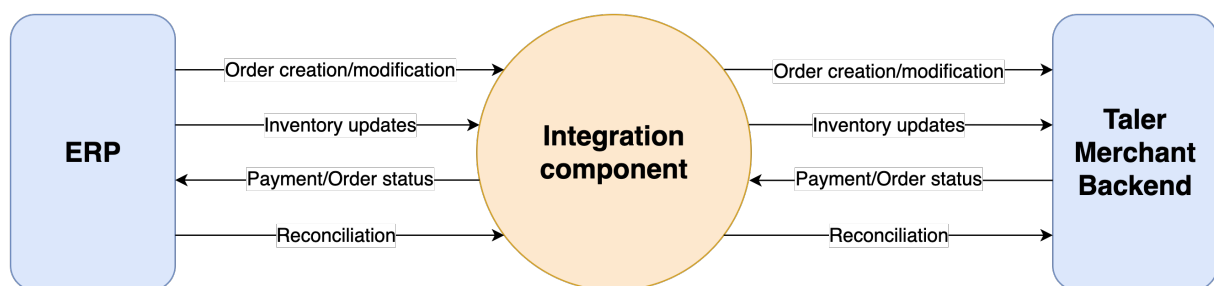
Figure 22: Data Flow Diagram: High-level Communication of Taler and ERP as Source of Truth

3.3.2 Inventory Management Process

The inventory management process is simpler compared to inventory-related processes in Taler. This is because its only job is to provide the inventory data needed for order receipts that consumers see in their wallets. So, there's no need to worry about constantly updating stock information.

Also, it might be a good idea to add a specific tag or category to mark products as "Taler related." That way, only the products marked for Taler get added to the Taler Merchant Backend inventory.

The **sequence diagram for the inventory management process the ERP system**, as illustrated in Figure 24, outlines the steps involved in updating inventory data from the ERP system to the Taler Merchant Backend. This process ensures synchronization of inventory information, enabling consistent and accurate data flow between systems. The key steps in this process include:
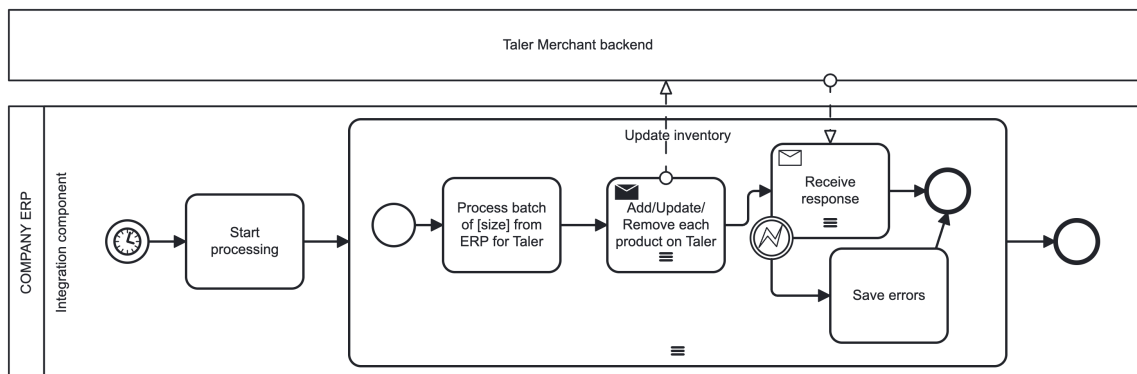
Figure 23: BPMN: Inventory Management Process from ERP System

1. **Batch Processing Initialization**:

   - The ERP system (Integration component) begins the inventory update process and prepares a batch of products to be synchronized with the Taler Merchant Backend.

2. **Batch Processing**:

   - The batch is sent from the ERP system to the Taler Merchant Backend, specifying the size of the batch for processing.

3. **Iterative Item Update**:

   - For each item in the batch:
     - The ERP system sends the item to the Taler Merchant Backend for addition or update.
     - The Taler Merchant Backend responds with the result of the update.

4. **Error Handling**:

   - If the response from the Taler Merchant Backend contains errors:
     - The ERP system saves the errors for review or further action.
   - If the response is successful:
     - The ERP system marks the item as successfully processed.

5. **Completion**:

   - Once all products in the batch are processed, the ERP system concludes the inventory update process and logs the outcome.

This process ensures that the inventory data in the ERP system remains consistent with the Taler Merchant Backend, allowing for accurate order fulfillment and inventory tracking. The iterative error handling mechanism allows for seamless recovery and resolution of issues during synchronization.
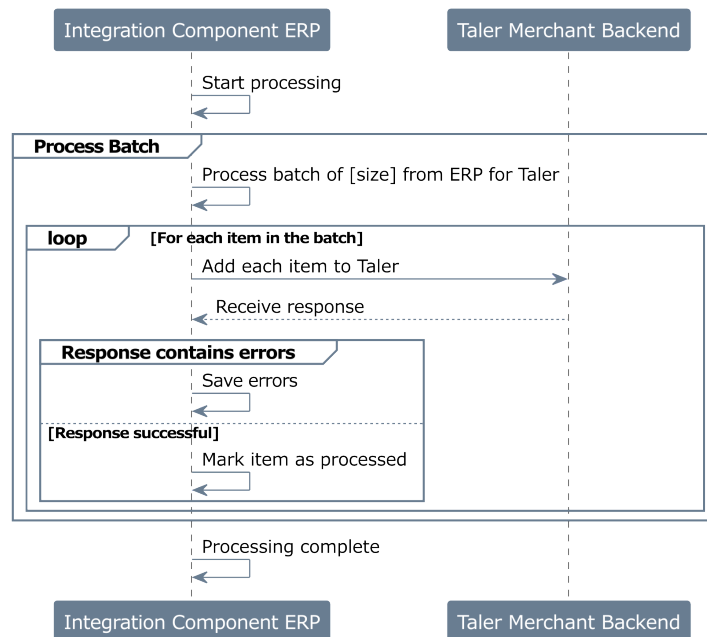
Figure 24: Sequence Diagram: Inventory Management Process from ERP System

3.3.3 Sales Process

The sales process is simpler compared to the sales processes where the Taler Merchant Backend acts as the source of truth. This approach may hold more significance for companies using their ERP system as the primary operational backbone while adding GNU Taler as a payment solution.

The BPMN diagram in Figure 25 illustrates the steps involved in processing an order within the ERP system. In this setup:

- The ERP system remains the primary manager of order creation, inventory updates, and financial workflows.
- GNU Taler is integrated solely for handling payment processing and synchronization.

This configuration allows businesses to maintain consistency in their existing workflows while seamlessly adding GNU Taler for enhanced payment capabilities.

Figure 25: BPMN: Sales Process from ERP System

The **sequence diagram for the sales process from the ERP system**, illustrated in Figure 26, outlines the detailed workflow of order and payment management when initiated from the ERP system. This process ensures seamless coordination between the ERP system and the Taler Merchant Backend for handling sales and payments. Key steps:

1. **Sales Process in ERP:**

   - The consumer initiates a sales order in the ERP system via the administrator or directly through an integrated platform.
   - The ERP system (Integration component) processes the sales order, verifying product availability and consumer details.
   - Once validated, the ERP system forwards the sales order information to the Taler Merchant Backend.

2. **Payment Processing in Taler Merchant Backend:**

   - The Taler Merchant Backend receives the sales order and begins the payment process.
   - The system generates a payment link or request for the consumer via the Taler Wallet.
   - The consumer completes the payment using the Taler Wallet, and the Taler Merchant Backend confirms the payment success.

3. **Payment Information Handling:**

   - The Taler Merchant Backend communicates payment confirmation back to the ERP system.
   - The ERP system logs the payment information, updates the financial records, and marks the sales order as completed.

4. **Completion:**

   - The ERP system finalizes the sales process, ensuring that inventory and accounting records are synchronized.
   - Administrators or users are notified of the successful order and payment completion.
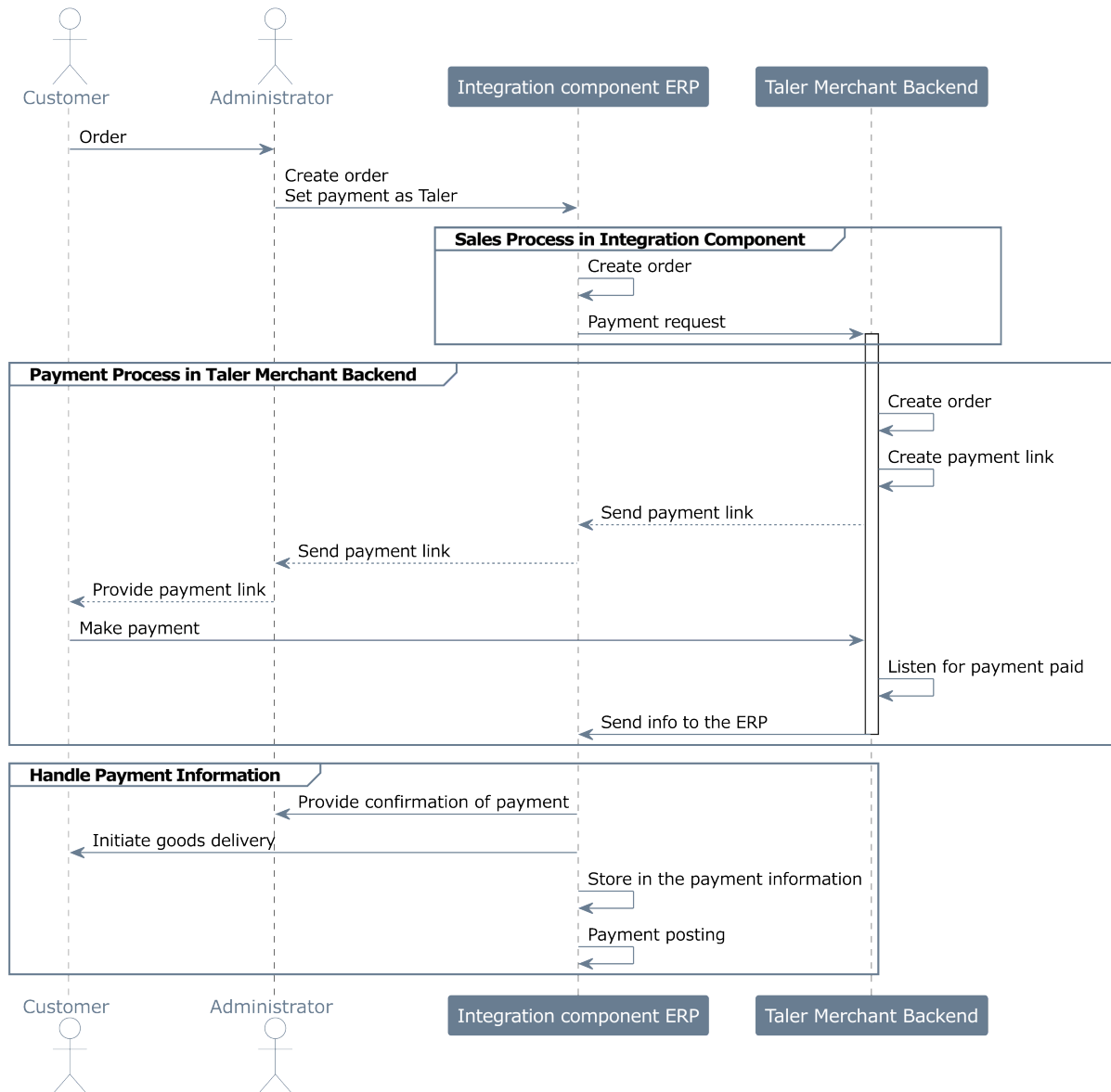
Figure 26: Sequence Diagram: Sales Process from ERP System

3.3.4 Refund Process

Similar to the refund process initiated from the Taler Merchant Backend, the ERP-initiated refund process involves the following steps:

1. **Saving Data in the Local Database**:
   The refund request is logged in a temporary local database for traceability and further processing.

2. **Locating the Order in the ERP System**:
   The system verifies the existence of the order in the ERP to ensure the refund request is valid.

3. **Initiating Return Order Creation**:
   A return order is created within the ERP system to handle the reverse transaction workflow.

4. **Creating a Billing Document for the Refund**:
   The system generates a billing document to record the financial details of the refund.

5. **Posting the Payment Document to the Refund Order**:
   The refund payment is posted to the return order, completing the process.

Unlike the process initiated from the Taler Merchant Backend, the ERP-initiated refund process allows for more flexible error handling. In cases where the Taler Merchant Backend fails to respond or returns an error, the system can trigger alternative workflows or manual intervention to ensure the refund is processed correctly.
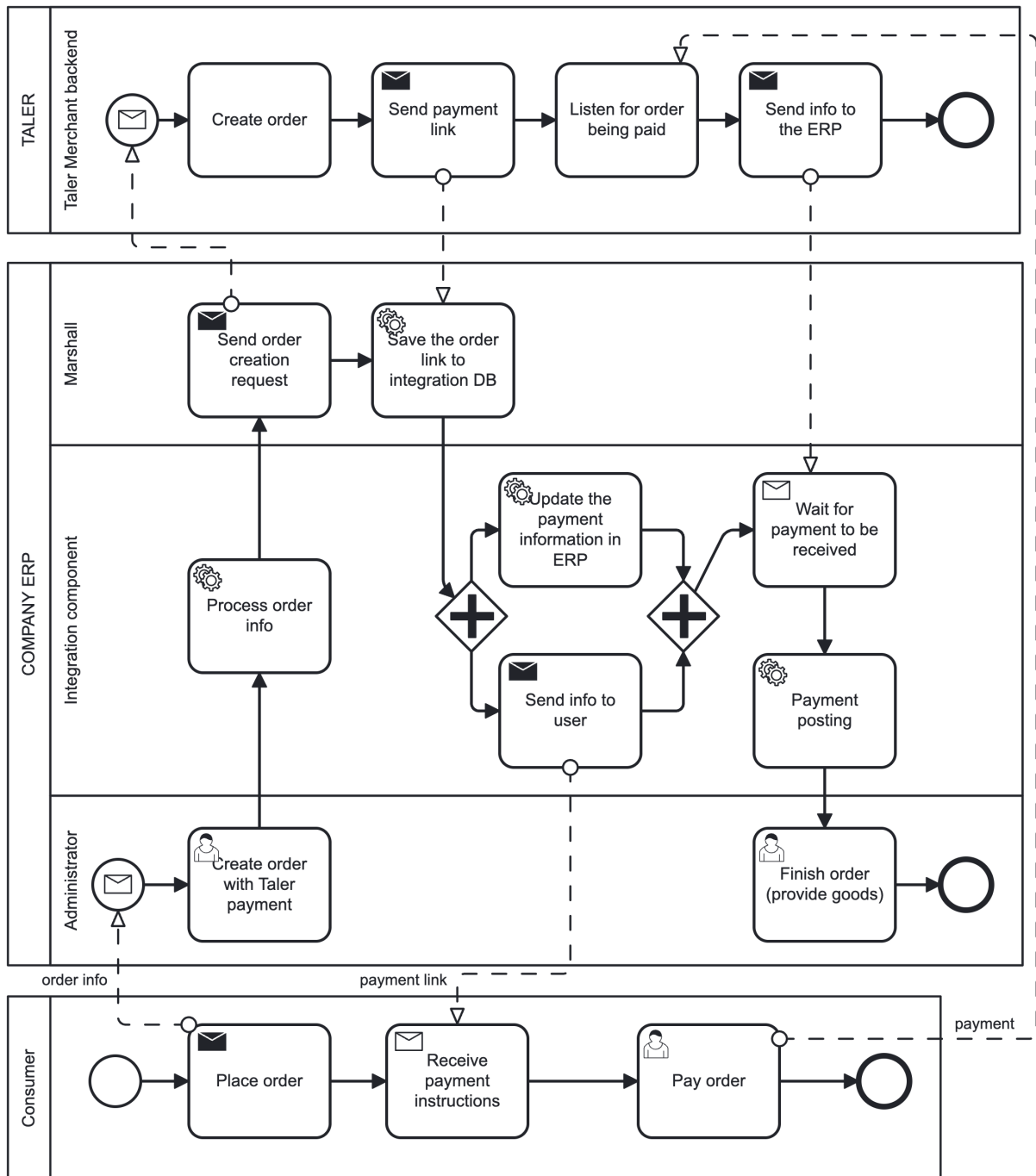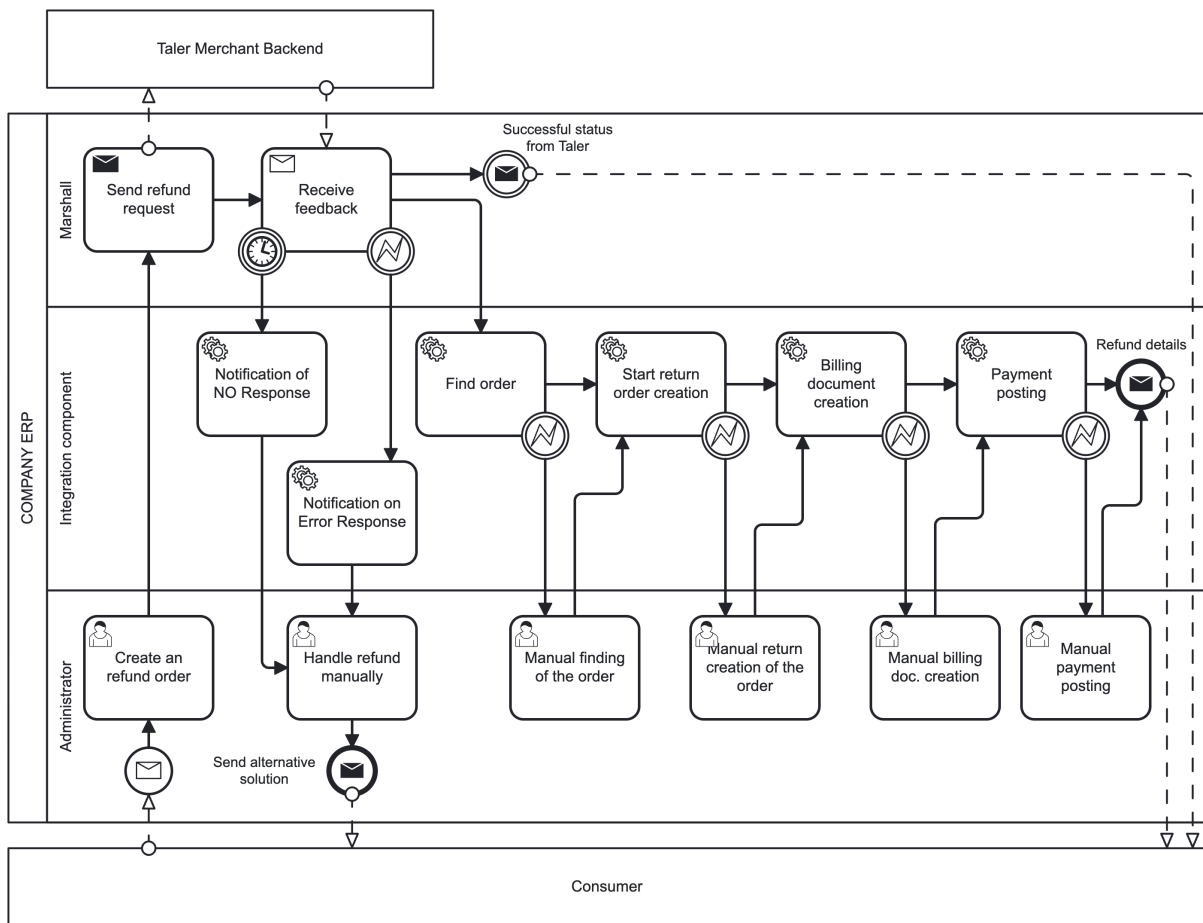
Figure 27: BPMN: Refund Process from ERP System

The **sequence diagram for the refund process from the ERP system**, as illustrated in Figure 28, outlines the key interactions between the ERP system, the Taler Merchant Backend, and the administrator during a refund process. This workflow ensures that refund requests are properly managed, tracked, and reconciled across both systems. Key steps in the process include:

1. **Consumer Request and Administrator Action**:

   - The consumer initiates a refund request via the ERP system.
   - The administrator verifies the request and triggers the refund process within the ERP system.

2. **Communication with Taler**:

   - The ERP system sends a refund request to the Taler Merchant Backend and awaits a response.
   - The Taler Merchant Backend processes the request and provides a successful response back to the ERP system, confirming eligibility for the refund.

3. **Finding and Verifying the Order**:

   - The ERP system searches for the original order associated with the refund request.
   - Upon successfully locating the order, the ERP system proceeds to the next steps.

4. **Return Order Creation**:

   - A return order is created within the ERP system to manage the refund workflow.
   - The ERP system confirms the successful creation of the return order and logs it for further processing.

5. **Billing Document Creation**:

   - The ERP system generates a billing document to document the financial aspects of the refund.
   - Confirmation of successful billing document creation is sent to the administrator and logged in the system.

6. **Completion and Notifications**:

   - The refund process is finalized within the ERP system, and all records are updated accordingly.
   - Both the consumer and the administrator are notified of the successful refund.
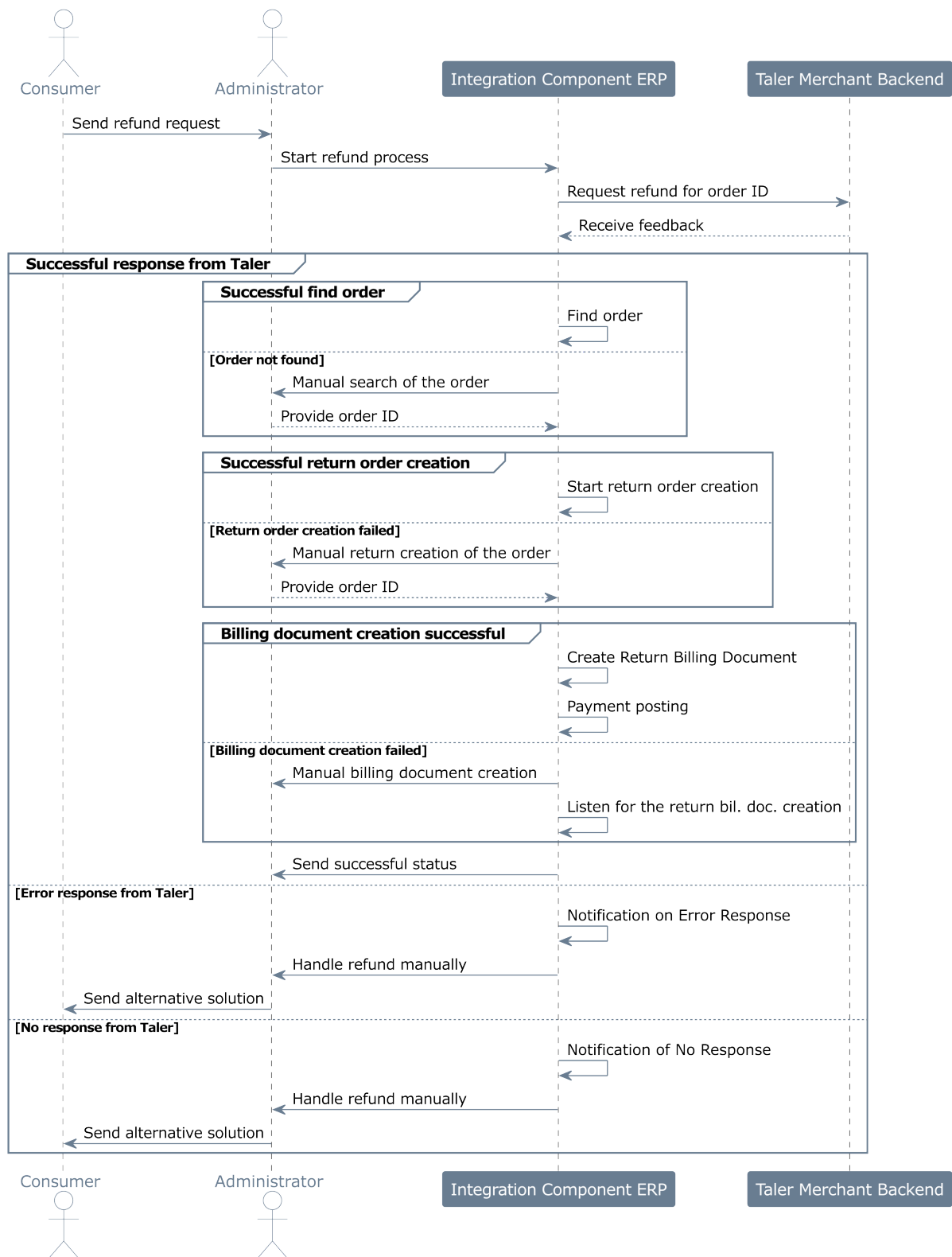
Figure 28: Sequence Diagram: Refund Order from ERP System

3.3.5 Payment Reconciliation Process

The payment reconciliation process ensures synchronization of financial transactions between the Taler system and the ERP system. As shown in Figure 29, the process involves:

1. **Check for Unprocessed Transfers**: The ERP system identifies pending transactions.
2. **Check for Taler Transfers**: Filters transfers related to Taler.
3. **Reconciliation Decision**:

   - **Yes**: Transfers are sent for reconciliation.
   - **No**: Process ends for non-Taler transfers.

4. **Send Transfers for Reconciliation**: Transfers are submitted to the Taler Merchant Backend.
5. **Update Order Payment Status**: If successful, payment status is updated in the ERP system.
6. **Manual Intervention**: Errors or no response prompt manual updates by administrators.
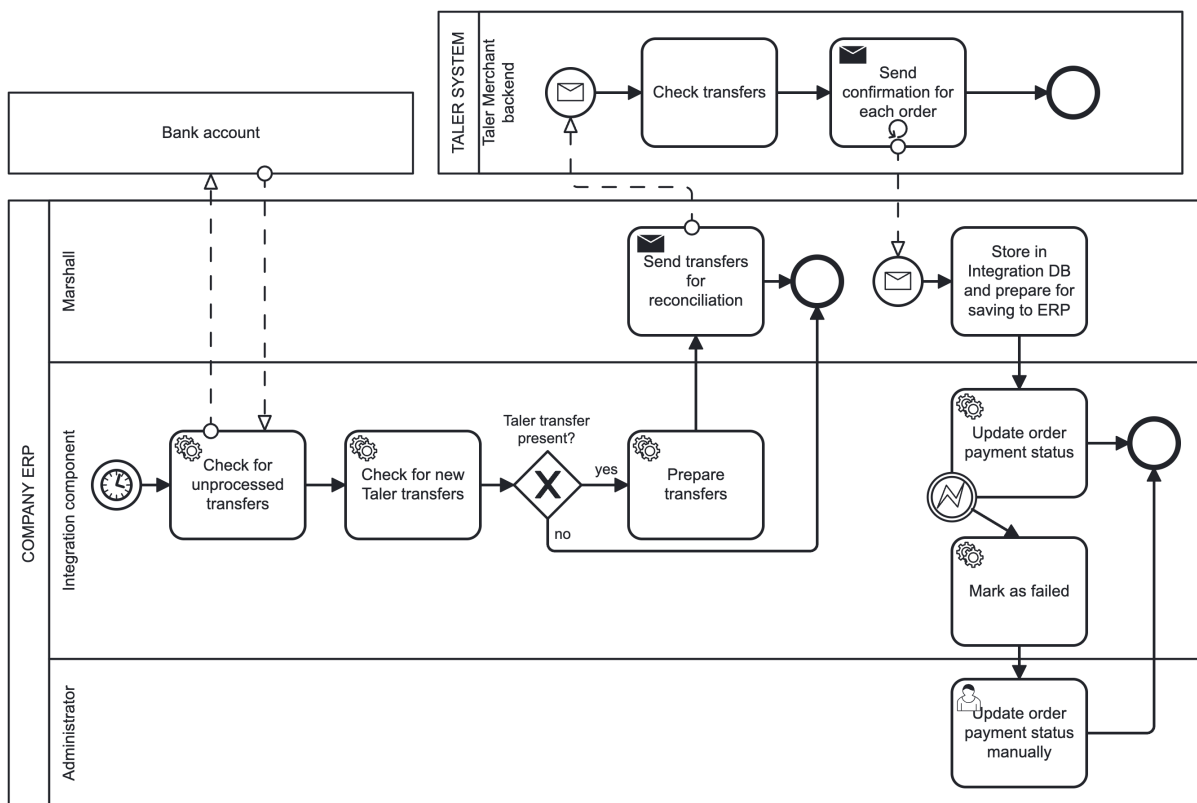
Figure 29: BPMN: Bank Reconciliation Process

3.4 User Interface

As part of this report, we propose a conceptual design for integrating GNU Taler functionalities into the ERP system. The user interface should focus on providing an intuitive experience with essential visual feedback for managing Taler-related orders.
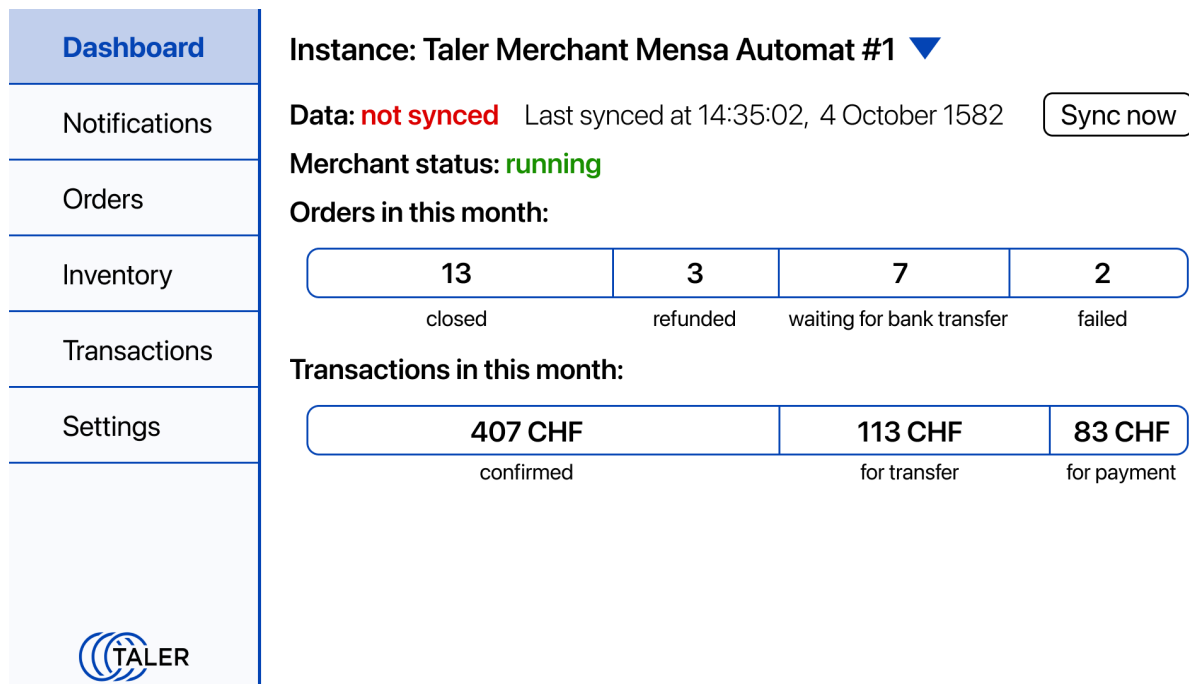
Figure 30: UI: Main Dashboard (Not Synced)

The main screen could feature an overview dashboard with key metrics and status indicators specific to Taler-related transactions, such as the number of closed, refunded, or failed orders, as well as transaction summaries. The design concept, depicted in Figure 30, showcases a straightforward layout with categorized sections for orders, inventory, transactions, and settings, enabling users to quickly access and manage Taler-related operations.

User roles and permissions should guide access to various views and functionalities within the system. For instance, administrators might have full access to configurations and transaction details, while regular users might only see limited views relevant to their tasks. This role-based access control ensures that sensitive data and operations remain secure and that users only interact with features aligned with their responsibilities.

For a detailed exploration of various interface designs and user interactions, see the UI samples provided in Appendix B. These samples demonstrate potential layouts for dashboards, order views, transaction summaries, and inventory management screens, offering a comprehensive starting point for implementation.

4 Discussion

4.1 Limitations

The study highlighted several potential challenges that could arise during the integration of GNU Taler with ERP systems. Mismatches in data formats between GNU Taler and ERP platforms, such as SAP or Dolibarr, could complicate data synchronization processes. API limitations, particularly in ERP systems, may restrict certain functionalities, such as handling edge cases like partial refunds or batch inventory updates. The theoretical approach also identified delays in real-time data exchange as a potential hurdle, which could impact operational efficiency in high-volume environments.

4.2 Security Considerations

Authentication and Access Control

GNU Taler APIs require **authentication tokens** for access, ensuring only authorized users can perform operations. Tokens must be securely stored and scoped to specific permissions. Role-Based Access Control (RBAC) within ERP systems should limit sensitive operations like refunds to trusted personnel.

Encryption and Data Protection

All communications between GNU Taler and ERP systems must use **TLS/SSL** encryption to secure data in transit. Sensitive data, such as order and payment information, must be encrypted at rest using database or field-level encryption.

Synchronization Mechanisms

Periodic reconciliation jobs ensure consistency between GNU Taler and ERP systems, while **webhooks** enable real-time updates on payments and inventory. Failed notifications should be retried with exponential backoff. Conflict resolution strategies and timestamps help identify the source of truth during discrepancies.

Implementation Considerations for ERP Systems

Using secure **API gateways** enforces rate limits and monitors API usage. Detailed access logs and monitoring tools help detect suspicious activities, while audit trails track changes to sensitive data for compliance and accountability. These practices ensure secure and reliable ERP integration.

4.3 Integration Strategy

The integration of GNU Taler with ERP systems involves a systematic and phased approach to ensure minimal disruption to existing workflows while leveraging the unique capabilities of both platforms.

By combining technical precision with strategic foresight, the following steps are recommended:

Connecting Taler's Backend API with the ERP System

The first step in the integration process is establishing a seamless connection between Taler's backend API and the ERP system's financial and operational modules. This involves:

- Conducting a detailed analysis of the ERP's existing architecture to identify integration points that align with Taler's API endpoints.
- Developing middleware to act as a translator between the ERP and Taler, ensuring compatibility and optimal performance.
- Testing API authentication mechanisms, such as token-based access, to guarantee secure and reliable data exchange.

By focusing on compatibility and security in this phase, businesses can establish a strong foundation for further integration.

Synchronizing Order and Payment Data

A critical aspect of the integration is synchronizing order and payment data between GNU Taler and the ERP system. This process includes:

- Mapping data flows between the systems to ensure that every transaction is accurately captured in both platforms.
- Utilizing Taler's webhook system to trigger real-time updates for payment statuses, refunds, and order changes, ensuring ERP data remains up-to-date.
- Creating reconciliation workflows to resolve discrepancies between systems during synchronization.

This synchronization not only improves operational efficiency but also reduces the risk of data inconsistencies that could lead to errors in financial reporting.

Implementing Phased Rollout for Core Functionalities

To ensure a smooth transition, the integration process should adopt a phased rollout strategy across key functional areas. Each phase should begin with testing in controlled environments to identify potential issues before full deployment. Suggested phases include:

a. Inventory Management

- Begin by synchronizing categories and items between Taler and the ERP system.
- Implement batch processing for large inventory updates to minimize performance bottlenecks.
- Use test environments to validate inventory accuracy before moving to production.

b. Order Management

- Integrate order creation workflows with the ERP to allow seamless processing from Taler to the ERP system.
- Test edge cases, such as incomplete orders or simultaneous updates, to ensure stability.
- Gradually expand order types (e.g., transfer on order being created, orders from different internal systems of ERP) as confidence in the integration grows.

c. Refund Handling

- Design refund workflows to reconcile customer refunds initiated in Taler with ERP records.
- Introduce robust error handling mechanisms to account for delays or failed refund transactions.
- Monitor user feedback during testing to refine the process before scaling.

This phased approach allows businesses to address challenges incrementally, reducing the risk of widespread disruption.

Testing Environments and Continuous Improvement

Establishing a robust testing environment is essential for validating the integration at each stage. Key considerations include:

- Using sandbox environments for both GNU Taler and the ERP system to simulate real-world scenarios without impacting live data.
- Running stress tests to evaluate the performance of the integration under high transaction volumes.
- Implementing automated testing frameworks to streamline validation processes and identify issues proactively.

Additionally, feedback loops should be established to gather insights from users and stakeholders, enabling continuous improvement and adaptation.

Handling Edge Cases and Failures

No integration is complete without mechanisms to address edge cases and failures. This includes:

- Designing fallback procedures for failed transactions, such as retry mechanisms or manual interventions.
- Implementing logging and monitoring tools to track synchronization delays and discrepancies in real time.
- Creating error resolution workflows to address issues with minimal operational impact.

By proactively preparing for edge cases, businesses can ensure that the integration remains resilient and reliable.

5 Conclusion

5.1 Key Findings

The study demonstrates that integrating GNU Taler with ERP systems like SAP and Dolibarr has the potential to significantly enhance operational efficiency. The proposed framework highlights how automation can reduce manual workflows, improve data accuracy, and streamline payment and order reconciliation processes.

5.2 Practical Implications

The theoretical integration offers valuable insights for businesses looking to modernize their operations. SMEs, in particular, can benefit from the privacy-focused payment features of GNU Taler combined with the advanced resource management capabilities of ERP systems like SAP and Dolibarr. By enabling real-time data synchronization and improved transaction transparency, the integration framework could help businesses reduce costs, enhance customer experiences, and adapt to evolving digital payment trends.

5.3 Future Research

Building on the findings of this study, the authors plan to pursue a practical implementation of the integration between GNU Taler and SAP S/4HANA in the coming months. This hands-on approach will validate the proposed framework, address real-world challenges, and further refine the methodologies for future integrations.

References

[1]   R. M. Stallman, *Free software, free society: Selected essays of richard m. stallman*. Boston, MA, USA: GNU Press, Free Software Foundation, 2002. Available: https://www.gnu.org/philosophy/fsfs/rms-essays.pdf

[2]   "GNU taler: Privacy-preserving payment system." GNU Project, 2024. Available: https://taler.net/

[3]   F. Dold, "The GNU taler system: Practical and provably secure electronic payments," Thesis, Université de Rennes 1, Inria, 2019.

[4]   "Dolibarr open source ERP and CRM." Dolibarr Foundation, 2024. Available: https://www.dolibarr.org/

[5]   D. Chaum, C. Grothoff, and T. Moser, "How to issue a central bank digital currency," *Swiss National Bank Working Papers*, no. 3, pp. 1–34, 2021.

[6]   "What is copyleft?" GNU Project, 2025. Available: https://www.gnu.org/licenses/copyleft.en.html

[7]   *Taler merchant backend manual*. Taler Systems SA, 2024. Accessed: Oct. 22, 2024. [Online]. Available: https://docs.taler.net/taler-merchant-manual.html

[8]   "GNU taler core API - merchant documentation." GNU Taler Project, 2024. Available: https://docs.taler.net/core/api-merchant.html

[9]   "GNU taler merchant API - YAML specification." GNU Taler Project, 2024. Available: https://git.taler.net/erp-integration-spec.git/tree/taler_api/Merchant-API.yaml

[10]   "GNU taler merchant API - postman collection." GNU Taler Project, 2024. Available: https://git.taler.net/erp-integration-spec.git/tree/taler_api/GNU%20Taler%20Merchant%20API.postman_collection.json

[11]   J. Frye and M. Newman, *Financial accounting in SAP ERP: Business user guide*. Rheinwerk Publishing, 2017.

[12]   "SAP history: 1972 to 1980." SAP SE, 2025. Available: https://www.sap.com/about/company/history/1972-1980.html

[13]   R. Anderson, *Using SAP s/4HANA: An introduction for business users*. Rheinwerk Publishing, 2019.

[14]   K. Kingsley, *Introducing SAP bank communication management in SAP s/4HANA*. Rheinwerk Publishing, 2020.

[15]   F. Färber, N. May, W. Lehner, P. Große, I. Müller, and J. Dees, "The SAP HANA database: An architecture overview," *IEEE Data Engineering Bulletin*, vol. 35, no. 1, pp. 28–33, 2012.

[16]   J. Boeder and B. Groene, *The architecture of SAP ERP: Understand how successful software works*. Books on Demand, 2014.

[17]   P. Mandal and A. Gunasekaran, "Issues in implementing ERP: A case study," *European Journal of Operational Research*, vol. 146, no. 2, pp. 274–283, 2003.

[18]   L. K. Lau, *Managing business with SAP: Planning, implementation, and evaluation*. IGI Global, 2005.

[19]   F. M. Elbahri, O. I. Al-Sanjary, and M. A. M. Ali, "Difference comparison of SAP, oracle, and microsoft solutions based on cloud ERP systems: A review," in *2019 IEEE 15th international colloquium on signal processing & its applications (CSPA)*, 2019, pp. 247–251. doi: 10.1109/C-SPA.2019.8695976.

[20]   T. Mladenova, "Open-source ERP systems: An overview," in *2020 international conference automatics and informatics (ICAI)*, 2020, pp. 203–207. doi: 10.1109/ICAI50593.2020.9311331.

[21]   A. Magnusson, "A framework for selecting an ERP open source system: A case study." Lund University, LU-CS-EX, 2016. Available: https://lup.lub.lu.se/student-papers/record/8900113/file/8900116.pdf

[22]   L. Aversano, "Issue reports analysis in enterprise open source systems," in *Proceedings of the 21st international conference on enterprise information systems (ICEIS)*, SCITEPRESS, 2019, pp. 234–241. Available: https://www.scitepress.org/Papers/2019/77578/77578.pdf

[23]   B. Beghman, "Integration of business intelligence system and management information systems: An open-source approach." Core.ac.uk, 2011. Available: https://core.ac.uk/download/pdf/148828825.pdf

[24]   E. Kadasah and O. Alrwais, "Evaluation of training modules in open source ERP," *International Journal of Information Technology*, 2022, Available: https://www.researchgate.net/publication/361326041

[25]   "Taler merchant integration guide." Taler Operations AG, 2024. Available: https://stage.taler-ops.ch/en/merchants.html

Appendices

Appendix A: Webhook documentation for Taler

Webhook Events   The GNU Taler Merchant Backend supports various webhook events that can trigger HTTP(S) requests to external servers based on specific actions such as payments, refunds, and inventory updates.

Path Table

Event Type	Trigger Condition	Provided Data Fields
order_pay	Payment for an order is received	`contract_terms`, `order_id`
order_refund	Refund is issued for an order	`timestamp`, `order_id`, `contract_terms`, `refund_amount`, `reason`
order_settled	An order is fully settled	`order_id`
category_added	A new product category is added	`webhook_type`, `category_serial`, `category_name`, `merchant_serial`
category_updated	A product category is updated	`webhook_type`, `category_serial`, `old_category_name`, `category_name`, `category_name_i18n`, `old_category_name_i18n`
category_deleted	A product category is deleted	`webhook_type`, `category_serial`, `category_name`
inventory_added	A new inventory product is added	`webhook_type`, `product_serial`, `product_id`, `description`, `unit`, `taxes`, `price`, `total_stock`, `total_sold`, `total_lost`, `address`, `next_restock`, `minimum_age`

Event Type	Trigger Condition	Provided Data Fields
inventory_updated	A product in inventory is updated	`webhook_type`, `product_serial`, `old_description`, `description`, `old_price`, `price`, `total_stock`, etc.
inventory_deleted	A product is deleted from inventory	`webhook_type`, `product_serial`, `product_id`, `description`, `price`, `total_stock`, `total_sold`, `total_lost`

Webhook Data Fields

Order Pay Events

- **Description**: Triggered when payment for an order is successfully completed.
- **Data Fields**:

  - `contract_terms`: JSON object of the contract terms.
  - `order_id`: Unique identifier of the paid order.

Order Refund Events

- **Description**: Triggered when a refund is processed for an order.
- **Data Fields**:

  - `timestamp`: Time of refund (in nanoseconds since 1970).
  - `order_id`: ID of the refunded order.
  - `contract_terms`: JSON of the contract terms of the refunded order.
  - `refund_amount`: Amount refunded.
  - `reason`: Reason for the refund (provided by merchant staff).

Order Settled Events

- **Description**: Triggered when an order has been fully settled, meaning all payments are wired to the merchant.
- **Data Fields**:

  - `order_id`: Unique identifier of the settled order.

Category Events

- **Category Added**:

    - `webhook_type`: `category_added`
    - `category_serial`: Unique ID of the category.
    - `category_name`: Name of the category.
    - `merchant_serial`: ID of the associated merchant.

- **Category Updated**:

    - Includes old and new category names, localized fields (`category_name_i18n`).

- **Category Deleted**:

    - `webhook_type`: `category_deleted`
    - `category_serial`: ID of the deleted category.

Inventory Events

- **Inventory Added**:

    - Includes product details such as `description`, `unit`, `taxes`, `price`, `total_stock`, `address`, and `minimum_age`.

- **Inventory Updated**:

    - Includes updated and previous values of product details such as `description`, `price`, and `total_stock`.

- **Inventory Deleted**:

    - `webhook_type`: `inventory_deleted`
    - `product_serial`: Unique product ID of the deleted item.

Webhook Configuration

1. Webhooks are set up using the following endpoints:

    - **Create a Webhook**: `POST` `/instances/{INSTANCE}/`**`private`**`/webhooks`
    - **Inspect Webhooks**: `GET` `/instances/{INSTANCE}/`**`private`**`/webhooks`
    - **Update a Webhook**: `PATCH` `/instances/{INSTANCE}/`**`private`**`/webhooks/{WEBHOOK_ID}`

- **Delete a Webhook**: `DELETE /instances/{INSTANCE}/`**`private`**`/webhooks/{`
  `WEBHOOK_ID}`

2. **Webhook Payload**: Webhook payloads are defined using **Mustache templates**. Depending on the triggering event, the templates receive event-specific data fields for expansion.

3. **Retries**: Webhooks will automatically retry (with increasing delays) if the target server returns a temporary error status (e.g., `HTTP 5xx`).

Appendix B: UI Samples

Dashboard
Notifications
Orders
Inventory
Transactions
Settings
TALER

Instance: Taler Merchant Mensa Automat #1 ▼

Data: synced

Merchant status: running

Orders in this month:

13	3	7	2
closed	refunded	waiting for bank transfer	failed

Transactions in this month:

407 CHF	113 CHF	83 CHF
confirmed	for transfer	for payment
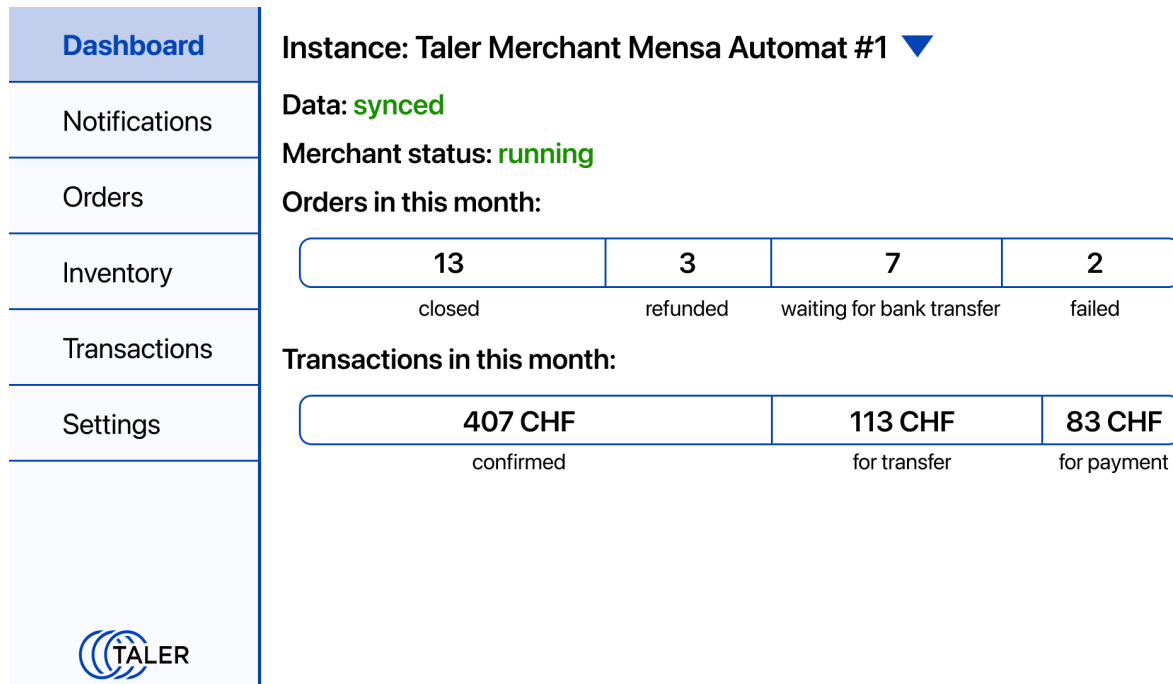
Figure 31: UI: Main Dashboard

Dashboard	Filters:						
Notifications							
Orders	**Id** ▼▲	Number	Product price	Products			
Inventory	**product1**	5434	$ 15 670	5 articles list	✎	PDF	•••
Transactions	**phone1**	5215	$ 3 231	3 articles list		PDF	•••
Settings	crazy1	4564	$ 5 674	7 articles list		PDF	•••
	product2	3765	$ 789	5 articles list	✎	PDF	•••
	product3	2145	$ 15 670	24 articles list	✎	PDF	•••
	phone2	4573	$ 3 231	3 articles list		PDF	•••
	phone3	6734	$ 5 674	7 articles list		PDF	•••
TALER	**phone4**	1298	$ 789	5 articles list		PDF	•••

Figure 32: UI: Inventory View

Dashboard	Filters:							
Notifications								
Orders	**Date** ▼▲	Type	Field	Status	Description			
Inventory	**23.06.2018**	message	order	No action	Lorem ipsum dolor sit amet, consectetur adipiscing elit.		PDF	•••
Transactions	**18.05.2018**	message	order	No action	Lorem ipsum dolor sit amet, consectetur adipiscing elit.		PDF	•••
Settings	**3.05.2018**	message	category	No action	Lorem ipsum dolor sit amet, consectetur adipiscing elit.		PDF	•••
	24.04.2018	message	transaction	No action	Lorem ipsum dolor sit amet, consectetur adipiscing elit.		PDF	•••
	23.06.2018	message	inventory	No action	Lorem ipsum dolor sit amet, consectetur adipiscing elit.		PDF	•••
	18.05.2018	error	order	Resolved	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	✎	PDF	•••
	3.05.2018	warning	inventory	Review	Lorem ipsum dolor sit amet, consectetur adipiscing elit.	✎	PDF	•••
TALER	**24.04.2018**	message	category	No action	Lorem ipsum dolor sit amet, consectetur adipiscing elit.		PDF	•••

Figure 33: UI: Notification View

	Date ▾ ▴	Number	Amount	Products	Payment	Delivery	Status			
Filters:										
	23.06.2018	5434	$ 15 670	5 articles list	Paid	23.06.2018 the data driver	Not sent	✎	PDF	•••
	18.05.2018	5215	$ 3 231	3 articles list	Pending payment	18.05.2018 the data driver	Confirmed		PDF	•••
	3.05.2018	4564	$ 5 674	7 articles list	Delay 13 days	3.05.2018 the data driver	Pending payment		PDF	•••
	24.04.2018	3765	$ 789	5 articles list	Paid	24.04.2018 the data driver	Paid	✎	PDF	•••
	23.06.2018	2145	$ 15 670	24 articles list	Paid	23.06.2018 the data driver	Loaded	✎	PDF	•••
	18.05.2018	4573	$ 3 231	3 articles list	Pending payment	18.05.2018 the data driver	Delivered		PDF	•••
	3.05.2018	6734	$ 5 674	7 articles list	Paid	3.05.2018 the data driver	Partially loaded		PDF	•••
	24.04.2018	1298	$ 789	5 articles list	Paid	24.04.2018 the data driver	Partially delivered		PDF	•••

Sidebar: Dashboard, Notifications, **Orders**, Inventory, Transactions, Settings — TALER

Figure 34: UI: Orders View

	Date ▾ ▴	Number	Amount	Products	Payment	Delivery	Status			
Filters:										
	23.06.2018	5434	$ 15 670	5 articles list	Paid	23.06.2018 the data driver	Not sent	✎	PDF	•••
	18.05.2018	5215	$ 3 231	3 articles list	Pending payment	18.05.2018 the data driver	Confirmed		PDF	•••
	3.05.2018	4564	$ 5 674	7 articles list	Delay 13 days	3.05.2018 the data driver	Pending payment		PDF	•••
	24.04.2018	3765	$ 789	5 articles list	Paid	24.04.2018 the data driver	Paid	✎	PDF	•••
	23.06.2018	2145	$ 15 670	24 articles list	Paid	23.06.2018 the data driver	Loaded	✎	PDF	•••
	18.05.2018	4573	$ 3 231	3 articles list	Pending payment	18.05.2018 the data driver	Delivered		PDF	•••
	3.05.2018	6734	$ 5 674	7 articles list	Paid	3.05.2018 the data driver	Partially loaded		PDF	•••
	24.04.2018	1298	$ 789	5 articles list	Paid	24.04.2018 the data driver	Partially delivered		PDF	•••

Sidebar: Dashboard, Notifications, Orders, Inventory, **Transactions**, Settings — TALER

Figure 35: UI: Transaction View

Dashboard

Notifications

Orders

Inventory

Transactions

Settings

TALER

Taler Merchant Backend URL:

https://ultimate.merchant.business.com

Access token:

megasecuretoken1234578

Main business system:

⦿ ERP   ◯ Taler

Figure 36: UI: Settings View

Dashboard

Notifications

Orders

Inventory

Transactions

Settings

TALER

Taler Merchant Backend URL:

https://ultimate.merchant.business.com

Access token:

megasecuretoken1234578

Choose main business system:

◯ ERP   ◯ Taler

Add Taler Merchant

Figure 37: UI: Settings Set-up View