

Automated Cargo Manifest Integration with Customs System Using Agile

Rizal Ahmad Maulana ^{1*}, Ajib Susanto ^{2*}

* Department of Informatics Engineering, Universitas Dian Nuswantoro
111202314974@mhs.dinus.ac.id¹, ajib.susanto@dsn.dinus.ac.id²

Article Info

Article history:

Received 2026-08-31

Revised 2026-10-09

Accepted 2026-10-18

Keyword:

*Cargo Manifest,
CEISA 4.0,
Host-to-Host Integration,
NVOCC,
Odoo ERP*

ABSTRACT

Non-Vessel Operating Common Carrier (NVOCC) operations require accurate and timely cargo manifest submission to customs systems, yet the absence of direct integration forces operators into a fragmented workflow of internal data preparation, manual file export, and separate portal upload, increasing operational burden and processing time. This study develops an NVOCC module on Odoo Enterprise Resource Planning (ERP) that integrates manifest data with the Indonesian customs system CEISA 4.0 via a Host-to-Host (H2H) interface. Unlike conventional flat-file API bridges, the module decomposes manifest data relationally across dedicated database models, incorporates a data cleansing engine to resolve common spreadsheet import anomalies, and applies role-based access control with TLS-encrypted communication to secure credential handling. Development followed an Agile approach across three sprints, during which functional and fault-injection testing identified and resolved two defects, an incomplete authentication configuration and inaccurate identity/container field mapping, before the module achieved a fully successful transmission. An XML fallback mechanism is proactively generated alongside every JSON payload, supporting submission continuity independent of H2H outcomes. Empirical evaluation across five operators using a single Master Bill of Lading encapsulating 255 House BL records demonstrated an 87.5% reduction in operator click interactions (from eight clicks to one) and a 93.3% reduction in average processing time (from 18.00 to 1.20 seconds), representing an approximately 15-fold increase in transmission speed. The study contributes an applied integration architecture combining ERP-based centralization, secure customs API integration, verifiable Agile-driven defect resolution, and a business-continuity fallback for NVOCC operations.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. INTRODUCTION

Digital transformation in the maritime logistics industry drives the need for integrated data exchange between enterprise operational systems and government services. In the Indonesian customs environment, the Customs-Excise Information System and Automation (CEISA) 4.0 provides an Open API that allows external systems to connect with customs services, including manifest data exchange. This mechanism provides an integration pathway for enterprise systems to exchange data with the Directorate General of Customs and Excise in a more structured manner [1].

A Non-Vessel Operating Common Carrier (NVOCC) is an operational entity that manages cargo data and transport

documents without operating its own vessels. In the manifest reporting process, the managed data includes master manifest information, House Bill of Lading, shipper, consignee, container, transport carrier, commodities, cargo weight, and various customs reference codes. The multitude of interrelated attributes necessitates consistent data throughout every stage of the submission process. The implementation of CEISA in the logistics enterprise environment indicates that the customs system has become an integral part of document management and logistics activities [2].

In practice, issues arise when a company's internal systems are not directly integrated with customs infrastructure. Manifest data residing in internal systems must be extracted, converted into a format compliant with CEISA specifications,

and subsequently uploaded via the Customs web portal as a formatted file (XML or JSON). This creates a fragmented workflow, requiring operators to repeatedly switch between internal systems and external submission channels for each manifest submission cycle [2]. In environments with high manifest transaction volumes, this workflow fragmentation increases the operator's workload, slows down document processing times, and creates opportunities for data discrepancies between systems.

This issue also relates to the need to maintain consistency in manifest attributes across every stage of the shipment process. Data points such as company identity, document numbers, HS codes, cargo weight, and container information must remain consistent between the internal system and the data transmitted to customs channels. Without a direct integration pathway, repetitive manual data conversion and transfer increase the risk of format mismatches and attribute errors, which can lead to document rejection by the CEISA system [2]. Therefore, application-level integration is required so that manifest data already present in the internal system can be processed and transmitted via a single automated workflow, eliminating the need for manual conversion and platform switching.

One approach that can be used to support the integration of these processes is Enterprise Resource Planning (ERP). ERP provides an integrated environment for managing data and business processes, enabling information to be utilized across interconnected functions. Studies on ERP implementation within supply chains indicate that ERP can support process integration and coordination throughout the supply chain [3]. At the implementation level, Odoo has also been used to support supply chain process management within corporate environments [4]. In addition to internal data centralization, a communication mechanism is required to connect the company's systems with external systems. RESTful web services represent an integration approach that enables data exchange between applications via API-based services [5]. On the platform side, Odoo also provides an external API that allows external applications to interact with the data and functions available within the Odoo system [6].

Previous research indicates that the implementation of CEISA has been examined within service and logistics environments [2], whereas research on ERP and Odoo has focused more on the integration of business processes and supply chains within companies [3], [4]. Research on RESTful web services also addresses inter-application integration mechanisms [5]. However, such studies have not specifically examined the integration of ERP-based NVOCC manifest management, Host-to-Host (H2H) communication with CEISA 4.0, and recovery mechanisms for instances where the primary channel fails. To clarify the position of this study relative to prior research, a comparison of research scopes serving as the basis for identifying the research gap is presented in Table I.

TABLE I
COMPARISON OF RESEARCH SCOPE

Study	ERP/Odoo	CEISA	H2H/API	Fallback	NVOCC Manifest
Pratiwi et al. [4]	Yes	No	No	No	No
Rayansa et al. [2]	No	Yes	Not focused	No	Logistics
Herfandi et al. [5]	No	No	Yes	No	No
Pratama et al. [7]	No	No	Yes	No	No
This study	Yes	Yes	Yes	Yes	Yes

Based on Table I, previous studies have focused on different aspects. Research by Pratiwi et al. [4] focused on the implementation of Odoo for supply chain management, whereas Rayansa et al. [2] addressed the implementation of CEISA within a logistics environment. Herfandi et al. [5] examined RESTful web services as a data integration mechanism, while Pratama et al. [7] implemented REST APIs in different information systems. Based on the selected studies reviewed in this study, no study was identified that combines ERP-based NVOCC manifest data management, H2H integration with CEISA 4.0, and an XML fallback mechanism within a single operational workflow.

The architectural contribution of this module extends beyond that of a simple API bridge, and is reflected in three distinguishing technical elements. First, rather than mapping data in a flat file manner directly from Excel into JSON/XML, as is common in simple commercial integration solutions, this module decomposes data relationally across three database models (*nvocc.master* for vessel header data, *nvocc.house* for consignment details, and *nvocc.goods* for commodity-level details), so that manifest data is stored in a structured form with potential reuse by other modules such as invoicing or warehouse management without duplicate data entry. Second, the module incorporates a data cleansing engine at the import stage that addresses common Excel-related anomalies, such as corrupted date formats and hidden whitespace characters, before the data is persisted to the database. Third, the module applies a smart-defaults strategy that automatically injects standard values for empty secondary attributes during payload generation (for example, a "NO MARKING" value for an empty Marking field), a defensive approach rarely implemented in conventional bridge applications.

Addressing this research gap, this study develops an Odoo ERP-based NVOCC module to serve as a central hub for managing manifest data prior to its transmission to CEISA 4.0. The module is designed to handle data importation, attribute validation, data mapping, JSON payload generation, authentication, and transmission via a Host-to-Host (H2H) channel. Official documentation from the Directorate General of Customs and Excise indicates that CEISA 4.0 provides an Open API as an integration pathway between external

systems and customs services [1]. By leveraging this mechanism, manifest data residing in the internal system can be processed through a unified, integrated workflow before being transmitted to the customs service.

In addition to H2H as the primary path, this study provides an XML-based fallback mechanism. When transmission via the H2H channel fails under the tested conditions, the system generates an XML file based on validated manifest data. This file can be used to resume the transmission process via an alternative channel available according to service conditions. This mechanism serves as an application-level business continuity approach, as the alternative path still requires a transmission channel and operator intervention.

The development of the NVOCC module employs the Agile approach. Agile was selected because the development process is iterative, requiring adjustments based on testing results and changing requirements during implementation. Systematic studies on Agile indicate that while the iterative approach is widely used in software development, its effectiveness remains contingent upon process management and feedback throughout the development lifecycle [8]. Other research also highlights that Agile adoption carries risks if aspects such as processes, communication, and stakeholder engagement are not effectively managed [9]. In this study, development is divided into three sprints: data modeling and data binding, H2H connectivity and authentication, and system resilience and testing.

This research contributes by integrating NVOCC manifest data management, payload validation and mapping, H2H communication with CEISA 4.0, and an XML fallback mechanism into a single Odoo ERP-based workflow. With this design, Odoo serves not only as an internal data management system but also as an integration point linking manifest processing with customs services. The system was subsequently evaluated through functional testing and fault injection to verify data import, validation, authentication, H2H transmission, and the generation of fallback XML files.

This research aims to: (1) design and implement an Odoo ERP-based NVOCC module capable of centralizing manifest data management, (2) implement authenticated Host-to-Host (H2H) communication to transmit manifest data to CEISA 4.0, (3) implement and test an XML-based fallback mechanism for instances where the H2H process fails, and (4) evaluate the system workflow using functional testing, fault injection, and quantitative measurement of the resulting processes.

II. METHODS

This research is an applied software engineering study aimed at designing and building an integrated NVOCC Module within the Odoo Enterprise Resource Planning (ERP) ecosystem [10]. System requirements data were gathered through direct observation of cargo manifest reporting workflows, functional interviews with operations management, and a literature review of the OpenAPI CEISA 4.0 technical specifications [11]. This technical data served as

the primary foundation for formulating the system architecture to address issues of data redundancy and reporting inefficiencies in logistics companies [2]. The research was conducted within the operational environment of an IT consultancy specializing in Odoo ERP implementation, focusing on the development of the NVOCC module. To produce a comprehensive, structured, and measurable solution, the methodological process was divided into four main stages: data collection, system design, development methodology, and software testing. A visual summary of these four research framework stages is illustrated in Figure 1.

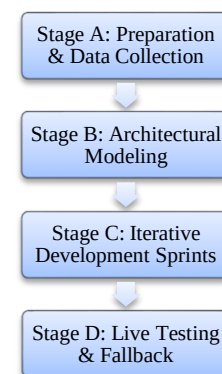


Figure 1. Research Framework

A. Data Collection Techniques.

The data collection procedure in this study was conducted through three main techniques to ensure the accuracy of system design. First, direct observation was conducted on the daily cargo manifest reporting workflow at the partner company, encompassing the data preparation and normalization stages in the internal system, the conversion process into a standardized file format, up to uploading to the CEISA 4.0 customs channel, to empirically identify bottlenecks caused by workflow fragmentation between systems [2]. Second, semi-structured interviews were conducted with two key informants, namely the Operational Director of the IT consulting company as the technical lead responsible for system development, and the Odoo Functional Division as system informants, to identify the specific functional requirements of the module to be built. Third, literature and documentation studies were conducted by reviewing the technical specifications of the CEISA 4.0 OpenAPI published on the official portal of the Directorate General of Customs and Excise, to formulate the standard parameters and cargo data structures that must be transmitted in every manifest delivery [12].

B. System Design

The system design phase focuses on modeling the software architecture as a solution to data integration issues. The module database design encompasses three main models within the Odoo framework:

- *nvocc.master* as the parent model containing manifest header data, including vessel information, voyage number, port of loading and destination, and NVOCC company data.
- *nvocc.house* as the detail model containing House Bill of Lading data per shipper and consignee transaction, including container number, HS code, cargo weight, and packaging type.
- *nvocc.reference* as a configuration model storing reference codes such as Customs office codes, country codes, container types, and agent codes.
- *nvocc.connection* as a model for storing connection configurations to CEISA 4.0, consisting of the Customs Username, Password, and API Key.

This design encompasses the configuration of database table relationships within Odoo ERP to enable the centralized consolidation of data entities specifically vessels, ports, shippers, and cargo consignees [4]. Additionally, the structure of output data payloads in both JSON and XML formats was modeled [13]. At this stage, the mapping of the XML output structure was precisely configured to distinguish between Passport and NPWP (Taxpayer Identification Number) identity formats. The system logic was specifically designed to avoid standard label attributes such as "TaxID," which were identified as frequently triggering the receiving system to unilaterally force the interpretation of the data as an NPWP.

With respect to attribute completeness requirements, the official CEISA payload specification in fact tolerates empty values for a number of secondary fields, such as *npwpPengirim*, *npwpNotify*, and certain supplementary data arrays. Nevertheless, this module applies internal validation that is more strict than the tolerance permitted by the official API, by rigidly blocking core logistics data elements, namely Customs Office Code, Company NPWP, Master BL Date, Means of Transport data, Ports, Container, Weight/Dimensions, and HS Code, when left unfilled. This engineering decision was made to minimize the risk of rejection by the CEISA server due to format errors (HTTP 400 Bad Request). For Port Code validation, CEISA requires the standard five-digit UN/LOCODE format (e.g., SGSIN, IDTES), compliance with this standard is maintained through two layers of protection: input restriction at the database level through a Many2one relational field that links port selection directly to a standardized reference table (preventing free-text entry by the operator), and forced string truncation at the payload-generation level to ensure the port code's character length conforms to the maximum limit permitted by the Customs system. Overall, the architectural design for the system integration spanning from the Odoo module to the CEISA 4.0 portal is illustrated in Figure 2.

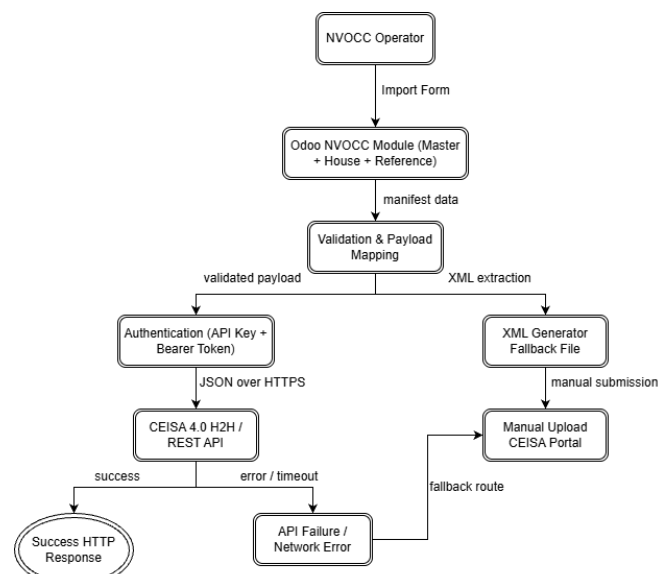


Figure 2. Integration Architecture NVOCC Module with CEISA 4.0

C. System Development Method.

The development of this NVOCC Module adopts the Agile Software Development method. The selection of the Agile method is highly relevant and crucial because the customs ecosystem is dynamic, where regulations and interface specifications from the Directorate General of Customs and Excise (DJBC) are subject to frequent updates [14]. This method offers high flexibility, enabling the system to continuously adapt to changes without requiring a restart of the development cycle [10]. By focusing on the iteration of high-value-added features, the completion of the core module integration becomes significantly more efficient [15]. Software engineering procedures using the Agile method in this study were executed iteratively through three main Sprint stages to ensure the system could adapt to CEISA specifications:

- **Sprint 1 (Structure Modeling & Data Binding):** Focuses on designing the Odoo relational database architecture to consolidate logistics entities (vessels, ports, cargo [4], and developing JSON and XML payload generation algorithms with specific Passport/NPWP format mapping to prevent attribute label coercion.
- **Sprint 2 (H2H Connectivity & Security):** Focuses on programming Host-to-Host (H2H) transmission using RESTful API architecture [5], integrating CEISA endpoints, and configuring authorization using Customs Bearer Tokens and API Keys.
- **Sprint 3 (System Resilience & Testing):** Focuses on developing a Fallback Mechanism in the event of external server failure, followed by functionality testing using Blackbox Testing and Fault Injection (failure engineering) methods.

Each Sprint concludes with a review phase to evaluate system performance and a phased deployment phase to the

production or testing environment to validate feature functionality before the next iteration begins [16]. The software engineering procedure is executed iteratively through three main Sprint stages. Details of the development iteration plan are outlined in Table II.

TABLE II
DEVELOPMENT ITERATION PLAN

Sprint	Objective	Activities	Output	Validation
1	Data modeling & binding	Odoo models, bulk import, payload mapping	Master/House/Reference	Data validation
2	H2H & security	Token retrieval, API request, response handling	JSON transmission	API response test
3	Resilience & test	Fault injection, XML generator, functional test	Fallback XML	Failure recovery test

The Agile adaptation practiced in this study is evidenced by two instances of learning and adjustment that occurred during development. First, in the early stage of Sprint 2, the initial assumption that API authorization required only username and password credentials proved insufficient after transmission testing returned an HTTP 401 Unauthorized response, this finding prompted an adjustment to the authorization architecture through the addition of a Customs API Key to the authentication scheme. Second, the discovery of an operator habit of inadvertently duplicating data rows within the Excel Manifest Data Template file prompted the addition of a real-time duplicate-checking logic during the import stage, outside the original Sprint plan. At the end of each Sprint, functional testing and acceptance approval were conducted by the Operational Director, acting as technical lead, together with the Odoo Functional Division team at the IT consulting company where this research was conducted, who determined the module's readiness to proceed to the next iteration.

D. Software Testing

Testing in this study focuses on validating data transmission capabilities and system resilience. All testing stages were executed using actual operational cargo manifest data obtained directly from the partner company. Utilizing a real-world dataset of 255 House Bill of Lading records ensures the testing environment accurately reflects the complexities and regulatory standards of NVOCC operations. Test attributes include master bill of lading numbers, company identities, IMO numbers, HS codes, weights, shippers, consignees, etc. The testing scenarios were divided into two main stages: functional testing (Blackbox Testing) to test Host-to-Host (H2H) transmission directly to the CEISA 4.0 gateway, and network failure engineering testing (Fault

Injection) to validate the XML file-based recovery mechanism.

To strengthen the evaluation, this study utilizes four quantitative metrics: operator interaction steps, platform switches, transmission success rate, and Fallback Generation Success Rate. The reduction in operator interaction steps is calculated using Equation (1).

$$\text{Step Reduction (\%)} = ((\text{Step_manual} - \text{Step_H2H}) / \text{Step_manual}) \times 100\% \quad (1)$$

The overall system workflow and testing scenarios are illustrated in Figure 3.

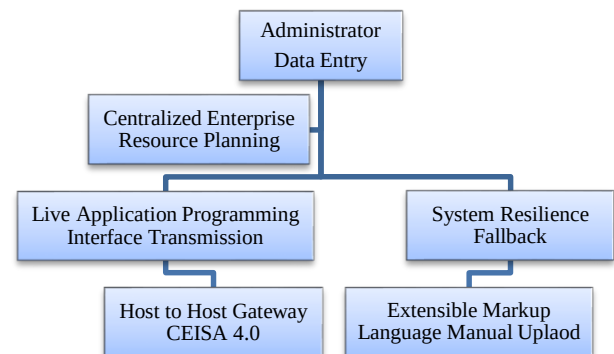


Figure 3. System Workflow and Testing

III. RESULT AND DISCUSSION

In accordance with the designed Agile framework, the results of the research and development of the integrated NVOCC Module within the Odoo ERP ecosystem were evaluated through three Sprint iterations. Each stage was specifically tested and analyzed to validate the resolution of real-world operational problems, encompassing the elimination of redundant input processes, dismantling administrative bottlenecks, to improving system resilience against API network failure risks.

A. Data Consolidation and Attribute Mapping.

The initial stage of NVOCC reporting operations involves processing cargo data in massive volumes. To eliminate manual re-entry processes prone to mechanical errors, this module utilizes a spreadsheet-based bulk-upload facility. The system enforces strict date format standardization (DD/MM/YYYY) to prevent errors or anomalies when the Odoo system extracts data into the central database. The spreadsheet data structure for the centralized mass upload process can be seen in Figure 4.

with OpenAPI specifications and transmits it directly to the CEISA server. The interface for executing manifest document transmission via this H2H API is shown in Figure 6.

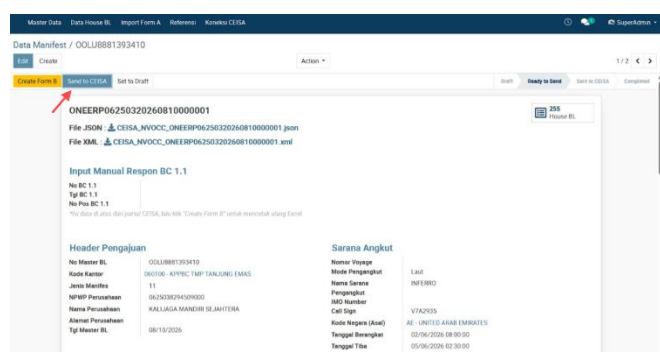


Figure 6. Document Transmission Execution via H2H API

The success of this two-way transmission is clearly recorded in the Odoo system logs. The log capture demonstrates that the CEISA endpoint responded to the transmitted payload with an HTTP Status Code 200 OK and a "Success" confirmation message, which automatically updated the document status within the ERP to "Sent to CEISA." Evidence of this status, in the form of the successful API response log, is presented in Figure 7.

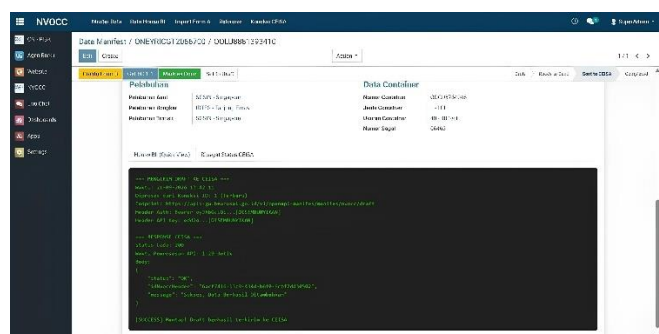


Figure 7. CEISA 4.0 H2H Success Console Log

Handling of responses from the CEISA 4.0 system in this module operates at two layers. At the authorization layer, the system processes the response from the authentication endpoint by extracting the access token together with its validity duration, once the token has expired, the system automatically triggers a request for a new token before proceeding with data transmission. At the transmission layer, the sending function reads the HTTP status code and determines a distinct course of action for each condition: a successful response (200/201) accompanied by a confirmation message from CEISA locks the document to "Sent to CEISA" status, a rejection response (400) displays the error details in the CEISA Status History log so the operator can trace the problematic attribute, an authorization failure response (401) notifies the operator of an issue with the Token or API Key, while a network failure or timeout is

halted without disrupting the stability of the Odoo application, accompanied by guidance to use the XML fallback file. If transmission is unsuccessful under any condition, the document status remains at "Ready to Send," and the failure message is recorded in the CEISA Status History log for operator follow-up.

Beyond the immediate HTTP response, transmission success in this study is corroborated through two additional layers of validation. First, at the application layer, the Odoo document status is transitioned from "Ready to Send" to "Sent to CEISA" only after the response payload explicitly reports an "OK" and "Sukses" (success) status, rather than being triggered by the HTTP status code alone, the transmission log shown in Figure 7 reflects this application-level confirmation. Second, CEISA 4.0 provides a dedicated OpenAPI status history endpoint (GET /v1/openapi-manifes/manifest/nvocc/status/{nomorAju}) that returns the submission history for a given manifest reference number, enabling the transmitted manifest to be independently cross-verified against the customs authority's own records rather than relying solely on the initial API response. The scope of the present evaluation extends up to the "Sent to CEISA" status, verification of the subsequent reconciliation stage, in which the customs-issued BC 1.1 approval number is retrieved and the document transitions to "Completed," was not yet evaluated at the time of this study and is noted as a limitation in the Conclusion.

To mitigate the risk of duplicate submissions, the module applies safeguards at two levels. At the application layer, the "Send to CEISA" action is single use: once a manifest has been successfully transmitted, the corresponding action control is removed from the document interface, preventing an operator from inadvertently re-triggering transmission for the same record. At the customs system layer, the H2H integration deliberately submits manifest data to CEISA 4.0 in a draft state rather than invoking the platform's alternative endpoint for immediate final submission. This design decision, adopted after consultation with the implementing organization's technical team, favors error-recovery capability over the throughput advantage of a direct-to-final submission mode: because a draft-state submission remains editable and, if necessary, removable on the CEISA portal, an inadvertent duplicate transmission can be identified and resolved prior to formal customs review, rather than producing two conflicting final submissions.

D. XML Schema and Fallback Mechanism.

Delegating the entire logistics operational workflow exclusively to a single API channel entails a high level of vulnerability, particularly when the external server experiences technical issues or downtime. To ensure system resilience, this module is designed with an automated XML schema generator that serves as a fallback mechanism. Figure 8 illustrates this XML document extraction process, which functions as an operational backup mechanism.

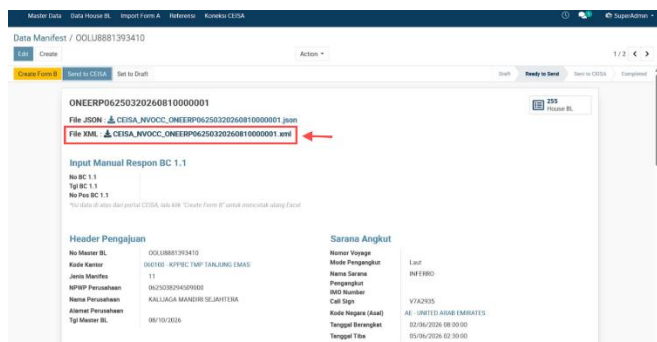


Figure 8. XML File Download as Fallback

For every submission with a "Ready to Send" status, the system automatically generates an .xml file (as shown in Figure 8) that is immediately available for download. The tag structure in this XML file has been engineered to strictly comply with CEISA 4.0 bulk upload specifications. If H2H transmission fails due to network anomalies, operational staff can manually upload this XML file to the CEISA web portal. This mechanism preserves the availability of validated manifest data for alternative submission when the H2H channel is unavailable.

From a transaction-management perspective, the module does not require manual re-entry when an H2H attempt fails or times out. The corresponding XML file is generated proactively at the moment a manifest is validated and transitions to "Ready to Send," prior to any H2H transmission attempt, rather than being generated reactively only after a failure is detected. Consequently, both submission pathways, H2H and XML, are derived from the same validated and locked dataset at the point of validation. When an H2H attempt fails, the previously generated XML file is immediately available for the operator to download and submit manually through the CEISA portal's XML upload channel, without requiring the manifest to be re-validated or reprocessed. This pre-generation strategy eliminates the delay that an on-demand fallback-generation step would otherwise introduce and ensures that a failed transmission attempt does not leave the manifest data in an inconsistent or unrecoverable intermediate state.

The selection of the XML format as a fallback mechanism is based on the official channel architecture provided by the Directorate General of Customs and Excise (DJBC). On the CEISA 4.0 web portal, manual manifest document uploads by users only support structured XML file formats through the Import XML menu, while the JSON format is exclusively dedicated to the Host-to-Host (H2H) transmission pathway via RESTful API. Therefore, providing an XML file is the only legally and operationally valid recovery route if the H2H connection is disrupted. The validation of the XML schema generated by the Odoo module was conducted through a reverse-engineering process against the official, approved CEISA XML file templates. The compliance level of the generated XML files was directly tested by uploading them to the CEISA portal and executing the system's "Completeness

Check" (POST /manifes/nvocc/cek-lengkap) feature, where the customs system successfully validated all document elements without any tag anomalies.

Although containing the same business data entities from the Odoo database, the data structures of the H2H JSON payload and the fallback XML file have fundamentally different schema architectures. The JSON payload is designed following the CEISA 4.0 OpenAPI specifications, which are flat hierarchical object-oriented, with the main nodes being the submission header, the *dataKelompokPos* array, *masterBl*, and container details in *dataBl*s, *blPetikemasTerangkut* and commodities in *dataBl*s, *blHs*. In contrast, the XML file follows a nested tag-based cascading declaration schema (<Declaration>) required by the CEISA 4.0 manual upload channel, where the transport means are defined in <BorderTransportMeans>, cargo details are mapped into repeating <Consignment><ConsignmentItem> elements, the master document number in <PreviousDocuments>, and container data is encapsulated in the <TransportEquipment><ContainerItem> tags. The Odoo module simultaneously manages the conversion of internal data into these two distinct schema structures during document validation.

The system does not currently apply differentiated automatic handling between temporary and permanent failures, all types of H2H transmission failure are treated uniformly. At the logging level, the transmission function is capable of classifying failure types, response codes 400 and 401 indicate a permanent failure (rejected data or invalid credentials), while a connection exception indicates a temporary failure (network disruption), however this classification is recorded solely in the CEISA Status History log without influencing the subsequent action taken by the system. Because the XML fallback file is generated proactively alongside the JSON payload at the moment the document is validated, rather than reactively after a failure is detected, the operator retains discretion in both temporary and permanent failure cases: retrying the H2H transmission when the log indicates a network disruption, or immediately downloading and uploading the already-available XML file when the data is indicated to have been structurally rejected by CEISA.

E. Functional Testing and Failure Scenarios.

The resilience and reliability of the NVOCC module were evaluated using the Fault Injection method based on empirical scenarios directly on the CEISA 4.0 transmission endpoint (POST /manifes/nvocc/draft). The first failure scenario tested network authorization reliability (Authentication Fault), where an initial transmission that only included the username and password credentials resulted in an HTTP 401 Unauthorized error code. The system successfully recovered after the authorization architecture was updated to include the Customs API Key alongside the user credentials. The second scenario tested resilience against Payload Parsing Faults.

During the initial development phase, failures were detected in the sender and receiver ID type attributes, which were unilaterally mapped to the default 16-digit NPWP, as well as the missing container type tag in the XML document.

The system responded to this failure by revising the conditional logic for data mapping within the Odoo module, ensuring passport identities and container data were precisely mapped according to the Manifest Template data. The third scenario tested master data completeness anomalies (Data Completeness Fault), where empty address fields in Odoo's Companies settings resulted in null values for the company address attribute, which were still passed into both the JSON and XML payloads without being intercepted by the system during the validation phase. Since the NVOCC module does not enforce mandatory checks on company master data completeness before transitioning the document to the "Ready to Send" status, this issue was subsequently resolved by completing the configuration on the client side rather than through the module's automated validation mechanism. Finally, a Network Disruption simulation was conducted by modifying the endpoint URL to ensure the system did not deadlock but immediately redirected the operator to utilize the fallback XML file. The details of the testing scenarios and the system recovery mechanisms are summarized in Table III.

TABLE III
FUNCTIONAL AND FAULT INJECTION TESTING RESULT

Scenario	Input/Condition	Response	Fallback Mechanism	Status
Authentication Fault	API transmission without including a valid CEISA API Key	HTTP 401 Unauthorized	The Odoo system logs the error, adding the API Key parameter to the <i>nvocc.connection</i> model restores authorization until a Bearer Token is issued.	Success
Payload Parsing Fault (H2H)	Passport identity format transmitted without conditional parsing	Schema rejection / data defaulted to NPWP	JSON mapping algorithm update separates Passport and NPWP attributes, precisely mapping the 255 House BL data.	Success
XML Schema Fault	Initial XML file extraction missing container type attributes	Blank container type attribute on the	Re-engineering of the <TransportEquipment> tag mapping in Odoo's XML generator,	Success

		CEISA portal	successful validation via the Completeness Check feature.	
Data Completeness Fault	Company address in Odoo Companies Settings is empty	Payload contains "Address not filled" value	System does not block the validation process, empty data is still passed to the payload. Issue resolved by completing the company master data in Odoo settings, not by the module's automated validation mechanism.	Success
Endpoint Disruption	Modification of the gateway endpoint URL	Connection Timeout	H2H transmission is automatically halted, the document remains in a safe status, and the fallback XML file is immediately available for the operator to download.	Success
Fallback Validation	Manual upload of the generated XML file to the CEISA web portal	CEISA 4.0 portal validation	XML document accepted without schema errors, manifest status is Draft on the Customs portal.	Success

F. Evaluation of Operational Efficiency Improvement.

The implementation of this software architecture effectively resolves inefficiencies in NVOCC reporting. Table IV summarizes the actual performance metric comparison between the conventional workflow and the H2H integrated system.

To strengthen the objective evaluation, measurements were conducted by comparing the number of operator interaction steps in each workflow for one manifest delivery cycle using a dataset of 255 House Bill of Lading records. This dataset consists of actual production transaction data from the partner company's operations, rather than simulated data, representing the manifest for the vessel INFERRO (Master BL Number: OOLU8881393410) with a shipping route from the Port of Singapore (SGSIN) to the Port of

Tanjung Emas, Semarang (IDTES), recorded during the operational period of January-February 2026.

TABLE IV
WORKFLOW EFFICIENCY COMPARISON

Evaluation Metric	Manual Process (Pre-H2H)	NVOCC H2H Module
Input Workload	Requires a multi-step submission workflow: internal data preparation, XML file export from Odoo, and manual upload to the CEISA web portal	Single centralized input within ERP, automatic transmission synchronization via API.
Processing Time	Averages 18.00 seconds per submission cycle due to manual platform switching.	Averages 1.20 seconds per transmission directly from the ERP.
Accuracy & Validation	Highly susceptible to human error on HS codes and identities.	Automatically validated with improved identity attribute logic.
Operational Reliability	Requires manual file handling and upload upon network disruption.	Fallback XML is proactively pre-generated at validation.

All reference data used in this testing, including port codes, packaging types, and identity types, was sourced directly from the official CEISA Customs OpenAPI reference database and loaded into the NVOCC module's Reference menu. In the conventional document submission phase based on XML export, the operator must perform 8 click interaction steps (including downloading the XML file, logging into the CEISA portal, navigating the main menu, navigating the sub-menu, searching for the file on the local computer, and uploading). With the H2H architecture, this entire sequence of interactions is reduced to a single click ("Send to CEISA") on the Odoo interface. This reduction in interaction steps is expressed by Equation (1):

$$\text{Step Reduction (\%)} = ((8 - 1) / 8) \times 100\% = 87.5\% (1)$$

To address the need for a consistent evaluation of efficiency improvement and ensure that the results are not solely influenced by a specific user's experience, processing time measurements were conducted directly in the partner company's Odoo production environment involving five different operators. The measurements were performed using the same dataset of 255 House Bill of Lading records. The comparison of actual processing time between the manual pathway (XML) and the integrated pathway (H2H) across the five operators is presented in Table V.

TABLE V
PROCESSING TIME COMPARISON

Operator	XML Time (seconds)	H2H Time (seconds)
Operator 1	16,00	1,08
Operator 2	19,00	1,37
Operator 3	19,00	1,20
Operator 4	16,00	0,91
Operator 5	20,00	1,42
Average	18,00	1,20

Based on Table V, during the submission phase, the H2H pathway yielded an average time of 1.20 seconds (range 0.91–1.42 seconds). Meanwhile, the manual XML pathway required an average of 18.00 seconds (ranging from 16 to 20 seconds). These results confirm that, consistently across operators with varying levels of system familiarity, the H2H pathway reduces processing time by 93.3%, operating on average 15 times faster than the manual pathway. Furthermore, the accumulation of these quantitative measurement results is summarized in Table VI.

TABLE VI
QUANTITATIVE MEASUREMENT RESULTS

Metric	XML Workflow	H2H Workflow	Improvement	Unit
Operator Interaction Steps	8	1	87.5% Reduction	clicks
Average Processing Time	18.00	1.20	93.3% Reduction	seconds
Platform Switches per Cycle	1	0	Eliminated	switches
Fallback Generation Success Rate	Manual upload path	Pre-generated at validation	Available before H2H attempt	-

The evaluation in Table VI demonstrates that the H2H implementation eliminates the need for operators to manually switch to the CEISA portal, drastically reducing the physical interaction burden from 8 clicks (which sequentially included: downloading the XML file, logging into the CEISA portal, opening the Manifest Online menu, opening the NVOCC submenu, clicking Restore Data, selecting the XML format, searching for the file in the local directory, and clicking the upload button) to a single click in the H2H pathway. Regarding transmission reliability, after the system successfully resolved identity mapping and authorization anomalies during the testing phase (Table III), the H2H channel was proven capable of processing the bulk payload in its entirety. The system successfully transmitted one bulk submission containing one Master BL and its 255 House BL records with precisely mapped attributes. Additionally, the XML fallback mechanism, verified under a simulated

endpoint disruption, successfully provided a valid backup file proactively without requiring manifest data re-validation, confirming that operational continuity is preserved even when the H2H channel is unavailable.

To obtain a more precise overview, processing-time measurements were conducted across five different operators using the same dataset of 255 House Bill of Lading records. During the submission phase, the H2H pathway yielded an average time of 1.20 seconds (standard deviation of 0.21 seconds, range 0.91-1.42 seconds). Meanwhile, the manual XML pathway required an average of 18.00 seconds (ranging from 16 to 20 seconds across all trials). These results indicate a processing time reduction of 93.3%, meaning the H2H channel operates on average 15 times faster than the manual pathway. Correspondingly, the number of physical operator interactions during the submission phase decreased drastically from 8 clicks in the XML pathway (which sequentially included: downloading the XML file, logging into the CEISA portal, opening the Manifest Online menu, opening the NVOCC submenu, clicking Restore Data, selecting the XML format, searching for the file in the local directory, and clicking the upload button) to a single click in the H2H pathway, representing an 87.5% reduction in interactions. All processing-time measurements in this study were conducted by five different operators using the same production dataset to ensure the consistency of the efficiency improvement regardless of the operator's familiarity with the system.

G. Scalability and Reliability Considerations.

To analyze the behavior of the integration architecture under high-volume and concurrent operational conditions, several scalability and reliability characteristics of the module were examined based on its underlying implementation. In terms of concurrency, Odoo's multi-worker architecture enables multiple operators to initiate H2H transmissions for different Master BLs simultaneously, because each transmission accesses independent database records, parallel submissions are processed through separate worker threads without inducing table locks or deadlocks at the application level. Regarding the transmission architecture, the current implementation processes each H2H request synchronously using a blocking HTTP call (`requests.post()`) with a 30-second timeout, leaving the operator interface in a waiting (loading) state until a response is received, the module has not yet implemented asynchronous background queueing (such as cron jobs or queue mechanisms) to decouple the transmission process from the operator's interactive session.

Concerning rate limiting, CEISA 4.0 imposes request thresholds on its API, the module currently detects the resulting HTTP 429 (Too Many Requests) responses and records them in the CEISA Status History log, but it has not yet implemented an automatic delay-and-retry mechanism, thus requiring operators to manually re-trigger the transmission once the limitation period expires. These

characteristics indicate that while the module's concurrency handling supports concurrent use at the application level, the synchronous transmission model and the absence of an automated recovery mechanism for rate limits present architectural constraints that must be addressed in future developments to support higher-volume operational scenarios. Regarding the potential for prolonged service disruptions at the CEISA gateway, this has been accommodated at the architectural level through the XML fallback mechanism. However, this study has not evaluated the module's behavior under prolonged operational disruptions on the Odoo server itself, which is noted as an additional limitation in the Conclusion section.

IV. CONCLUSION

This research successfully designed and implemented an Odoo ERP-based NVOCC Module integrated Host-to-Host with the CEISA 4.0 infrastructure. The four stated objectives were addressed through the implemented architecture and evaluation. First, the module effectively centralized manifest data management into a single internal database, eliminating repetitive tasks. Second, authenticated H2H communication was successfully implemented, evidenced by an HTTP 200 OK response from the CEISA gateway. Third, the proactive XML-based Fallback mechanism was successfully executed, ensuring documents remain ready for manual upload when the API channel is disconnected.

Fourth, the evaluation proves that the H2H architecture reduces operator interaction steps by 87.5% and eliminates manual platform switching. Further testing validated the system's capability to process a single bulk payload (1 Master BL encompassing 255 House BLs) in its entirety after authorization configuration and attribute mapping anomalies were resolved. The XML fallback mechanism was independently proven to proactively generate a compliant backup file under a simulated network disconnection, supporting the continuity of manifest submission operations even when the primary communication line is severed. Nevertheless, this study acknowledges operational limitations stemming from a high dependency on the dynamics of the CEISA 4.0 OpenAPI. Unilateral regulatory modifications by the customs authority such as the recent mandate shifting container type formats from alphabetical characters to numeric codes necessitate continuous system adaptation. The module mitigates this challenge by centrally and dynamically assembling the JSON payload within a single function (`action_generate_json`), enabling isolated data structure updates. Furthermore, the current operational scope is exclusively restricted to initial submissions targeting the draft endpoint (`/nvocc/draft`). Post-submission document revision scenarios (*manifest redress*) are not currently accommodated, aside from reverting the document to a *Draft* state for manual re-importation. Consequently, the implementation of asynchronous transmission queues (*cron jobs*), an automatic delay-and-retry mechanism to handle API rate limiting, and

the integration of API redress functionalities constitute critical development areas recommended for future research.

REFERENCES

- [1] Direktorat Jenderal Bea dan Cukai, "CEISA 4.0 Open API," Kementerian Keuangan Republik Indonesia. Accessed: Aug. 17, 2026. [Online]. Available: <https://openapi.beacukai.go.id/>
- [2] W. R. Rayansa, E. B. Setiawan, M. Ruminda, P. Ricardianto, A. Wardhana, and E. N. Adi, "The Implementation of Customs-Excise Information System and Automation Application in Logistic Companies at Soekarno-Hatta Airport, Indonesia," *International Journal of Scientific Engineering and Science*, vol. 6, no. 10, pp. 28–35, 2022.
- [3] G. T. Pontoh, A. Indrijawati, F. Selvi, L. Ningsih, and D. R. Putri, "A Systematic Literature Review of ERP and RFID Implementation in Supply Chain Management," *WSB Journal of Business and Finance*, vol. 58, no. 1, pp. 80–96, 2024, doi: 10.2478/wsbjbf-2024-0009.
- [4] D. A. G. Pratiwi, G. A. A. Putri, and I. P. A. E. Pratama, "Implementasi Supply Chain Management Menggunakan Software Odoo (Studi Kasus Perusahaan Furniture)," *JITTER- Jurnal Ilmiah Teknologi dan Komputer*, vol. 3, no. 2, 2022.
- [5] Herfandi, M. Julkarnain, and M. Hanif, "Desain dan Implementasi RESTful Web Services untuk Integrasi Data dan Aplikasi," *JINTEKS (Jurnal Informatika Teknologi dan Sains)*, vol. 4, no. 1, pp. 36–41, 2022.
- [6] Odoo, "External API," Odoo Documentation. Accessed: Aug. 17, 2026. [Online]. Available: https://www.odoo.com/documentation/18.0/developer/reference/external_api.html
- [7] M. W. Pratama, T. Fatimah, and A. A. A. Ushud, "Penerapan Metode REST API Untuk Presensi Siswa Magang Pada PT. Menara Indonesia," *Bit (Fakultas Teknologi Informasi Universitas Budi Luhur)*, vol. 19, no. 2, pp. 118–123, 2022.
- [8] S. P. Rath, N. K. Jain, G. Tomer, and A. K. Singh, "A systematic literature review of agile software development projects," *Inf. Softw. Technol.*, vol. 182, p. 107727, 2025, doi: 10.1016/j.infsof.2025.107727.
- [9] J.-N. Meckenstock, "Shedding light on the dark side - A systematic literature review of the issues in agile software development methodology use," *Journal of Systems and Software*, vol. 211, p. 111966, 2024, doi: 10.1016/j.jss.2024.111966.
- [10] S. B. Atim, "Permodelan Sistem Informasi Penjualan Barang Berbasis Website Menggunakan Metode Agile," *Journal of Artificial Intelligence and Technology Information (JAITI)*, vol. 2, no. 1, pp. 14–25, 2024.
- [11] I. F. Nurmasita, K. Itsnaeni, and N. Savitri, "Analisis Literatur Sistem Informasi Kepabeanan (CEISA) Dalam Efisiensi Pencatatan Laporan Keuangan Pada Instansi Bea Cukai Indonesia," *Jurnal Riset Akuntansi*, 2025.
- [12] N. N. Syahidah and D. Saputera, "Implementation of the Use of CEISA 4.0 By PT Putra Dwimitra Lestarindo As A Customs Clearance Service Companies (PPJK) In Improving The Ease Of Export And Import Activities," *COSTING: Journal of Economic, Business and Accounting*, vol. 8, no. 4, 2025.
- [13] S. Hidayat, H. C. Kirana, H. Wicaksono, and K. Nugroho, "Perancangan Enterprise Architecture Berbasis Togaf ADM Untuk Integrasi Sistem Menggunakan ODOO ERP pada PT Bromindo Mekar Mitra," *SENTRI: Jurnal Riset Ilmiah*, vol. 5, no. 1, pp. 136–143, 2026.
- [14] N. S. Oktaviani and H. Baturohmah, "Penggunaan Pada Ceisa 4.0 Untuk Meningkatkan Pelayanan Penerimaan Dokumen Pabean Di Kantor Bea dan Cukai kota Bogor," *Jurnal Pengabdian Kepada Masyarakat Abdi Putra*, vol. 5, no. 3, pp. 295–299, 2025.
- [15] J. C. Chandra, M. Anthony, Ricky, and D. Novita, "Systematic Literature Review Analisis Efektivitas Metode Agile Dalam Pengembangan Sistem Informasi," *INDEXIA: Informatic and Computational Intelligent Journal*, vol. 6, no. 2, pp. 123–131, 2025.
- [16] S. Supriyatna, E. Fahrudin, and R. Hartono, "Penerapan Metode Agile Dalam Pembuatan Aplikasi Berbasis Web Untuk Pengembangan Usaha Berkelanjutan (PUB) Pada Badan Usaha Jasa Konstruksi," *Spectrum: Multidisciplinary Journal*, vol. 2, no. 3, pp. 184–198, 2025.