UDC 004.412:519.237.5

# INFORMATION SYSTEM FOR COLLECTING ACCOUNTING DATA FROM ZENOTI

Pukhalevych A., Ph.D. (Computer Science), Assoc. Prof.[1]; Stjernelykke M.[3]; Frost T.[3]
[1] Admiral Makarov National University of Shipbuilding; [2] Origix ApS; [3] ownr ApS
[1] Ukraine, Mykolaiv; [2] Denmark, Ejby; [3] Denmark, Glostrup
[1] andrii.pukhalevych@nuos.edu.ua

***Abstract.*** *This paper presents the design approach of an information system for collecting accounting data from Zenoti platform through API integration. The proposed information system enables systematically retrieving operational and financial data through Zenoti's REST API endpoints and storing it in a local database for subsequent transfer to accounting systems such as e-conomic. The study identifies required data entities and API endpoints.*
***Keywords:*** *Zenoti; API integration; accounting data; information system; data synchronization; REST API; webhook integration.*

***Introduction.*** Contemporary business environments increasingly rely on specialized information systems to manage operational processes and financial data. Among these platforms, Zenoti has emerged as a leading solution, serving thousands of businesses globally with its cloud-based architecture and extensive feature set [1].

Zenoti's architecture leverages Amazon Web Services infrastructure to provide scalable, reliable service delivery across multiple geographic regions. The platform processes millions of appointments annually and manages complex operational workflows including service delivery, product sales, employee scheduling, and customer loyalty programs [1]. However, despite its comprehensive functionality within operational domains, Zenoti exhibits limited native integration capabilities with external accounting systems commonly deployed in financial management workflows.

This integration gap creates significant challenges for organizations requiring seamless data flow between operational and financial systems. Manual data transfer processes introduce error potential, consume staff resources, and delay financial reporting. Furthermore, the complexity of Zenoti's data model, which encompasses multiple entity types with hierarchical relationships, makes manual accounting reconciliation particularly challenging. These factors establish clear necessity for automated information systems capable of extracting, transforming, and preparing Zenoti data for integration with accounting platforms such as e-conomic, QuickBooks, or similar systems.

***The research aim*** is to identify and document the complete set of Zenoti API endpoints and data collection procedures required to establish comprehensive accounting data synchronization.

***The main part.*** The base URL for all Zenoti REST API operations is `https://api.zenoti.com/v1` [2]. Authentication is performed using API keys provided through Zenoti's developer portal, with each request requiring proper authorization headers [2, 3].

Zenoti's API architecture implements a center-based organizational model where all data operations are scoped to specific business locations (centers). Consequently, the initial requirement for any data collection workflow is obtaining the complete list of centers under the organization's account [3]:

```
GET /centers
```

The response contains center identifiers, names, addresses, and operational status information.

Master data entities represent relatively static information that changes infrequently and should be synchronized on a daily basis. The following endpoints must be invoked sequentially for each center in the organization.

Services [3]

`GET /centers/{center_id}/services?expand=additional_info&expand=catalog_info&expand=variants_info&expand=add_ons_info`

This paginated endpoint retrieves service catalog data including pricing structures, tax configurations, service variants, and add-on options. Service pricing and tax treatment may vary by center, necessitating center-specific retrieval. The expand parameters ensure complete service metadata is included in responses.

Employee Data [3]:

`GET /centers/{center_id}/employees`

Employee records contain provider information essential for commission calculations, service attribution, and labor cost allocation in accounting systems. This paginated endpoint returns active and inactive employees with their roles, schedules, and commission structures.

Product Catalog [3]:

`GET /centers/{center_id}/products`

Product data includes retail inventory items sold through the platform. This paginated endpoint returns SKUs, descriptions and pricing for all products.

Package Information [3]:

`GET /centers/{center_id}/packages?expand=centers&expand=benefits`

Packages represent prepaid service or product bundles requiring specialized accounting treatment. This paginated endpoint returns package definitions, pricing, center availability, and included service benefits necessary for deferred revenue calculations.

Guest Data [3]:

`GET /guests/?center_id={center_id}&expand=address_info&expand=preferences`

This paginated endpoint returns customers master data. Guest records contain customer information, contact details, preferences, and loyalty program status. For incremental updates after initial synchronization, append `&last_updated=YYYY-MM-DD` parameter to retrieve only records modified since previous run.

Appointment Data [3]. The appointments endpoint requires date range specification with a maximum interval of 10 days:

`GET /appointments?center_id={center_id}&start_date=YYYY-MM-DDT00:00:00&end_date=YYYY-MM-DDT23:59:59`

For initial historical data import, the system must iterate through consecutive 10-day intervals to retrieve complete appointment history. For ongoing synchronization, daily execution with previous day's date range ensures current data availability.

Invoice data represents completed financial transactions and constitutes the most critical information for accounting integration. Zenoti does not provide a direct endpoint for listing invoices; instead, two complementary approaches enable invoice data collection.

The *Invoice.Closed* webhook event notifies the information system immediately when invoices are finalized. This real-time approach ensures minimal latency between transaction completion and accounting system availability. Upon receiving the webhook notification, the system must request the specific invoice details using the invoice identifier provided in the webhook payload.

Sales Report Endpoint:

```
POST /reports/sales/accrual_basis/flat_file
```

This paginated endpoint retrieves sales data on an accrual basis suitable for accounting purposes. The request body must specify:

```
{"start_date": "YYYY-MM-DD 00:00:00", "end_date": "YYYY-MM-DD 23:59:59"}
```

For webhook-triggered updates, include *guest_ids* filter to retrieve specific transactions:

```
{
  "start_date": "YYYY-MM-DD 00:00:00",
  "end_date": "YYYY-MM-DD 23:59:59",
  "guest_ids": ["guest_id"]
}
```

The sales report response contains comprehensive financial information including earned and redeemed loyalty points on each invoice item, package utilization (quantity and amount), taxes, discounts, and payment allocations.

Following invoice closure, detailed transaction data must be retrieved:

```
GET /invoices/{invoice_id}?expand=InvoiceItems&expand=Transactions
```

This endpoint returns line-item details including services performed, products sold, employee attribution, pricing breakdowns, applied discounts, and payment transactions. The expand parameters ensure complete invoice structure including all nested objects.

Loyalty points affect revenue recognition and require separate tracking:

```
GET /guests/{guest_id}/points/1?invoice_id={invoice_id}
GET /guests/{guest_id}/points/0?invoice_id={invoice_id}
```

The first endpoint retrieves earned points for the invoice, while the second retrieves redeemed points. This information supports loyalty liability accounting and revenue deferral calculations.

The accounting summary report provides daily closure totals essential for reconciliation:

```
GET /reports/accounting_summary
```

This endpoint returns aggregated daily totals across payment methods, refunds, tips, taxes, and other financial categories. These totals enable validation that detailed transaction imports match Zenoti's internal accounting calculations.

All retrieved data must be persisted in a local database implementing a normalized schema that mirrors Zenoti's entity relationships while supporting efficient querying for accounting system export.

***Conclusions.*** Comprehensive accounting data collection from Zenoti requires execution of a complex sequence of API operations spanning master data retrieval and synchronization, and real-time webhook integration. The information system must orchestrate daily master data updates across multiple entity types, manage paginated responses for large datasets, implement incremental update strategies for guest data, handle date-windowed appointment queries, and integrate webhook notifications for real-time invoice collection.

## REFERENCES


[1] Amazon Web Services. (2025). *Zenoti Uses Generative AI to Enhance Customer Experience through Its HyperConnect Solution.* https://aws.amazon.com/solutions/case-studies/india-zenoti-case-study/

[2] Zenoti. (n.d.). *Zenoti API General Setup and Documentation*. https://docs.zenoti.com/

[3] Zenoti. (n.d.). *Zenoti API Reference.* https://docs.zenoti.com/reference/