# International Journal of Advanced Research in Education and TechnologY (IJARETY)

Volume 12, Issue 2, March-April 2025

Impact Factor: 8.152

www.ijarety.in editor.ijarety@gmail.com

International Journal of Advanced Research in Education and TechnologY(IJARETY)

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

# Scalable Workday Benefits Integration Frameworks for Enterprise HR Technology

**Venkata Kalyan Polamarasetty**

Independent Researcher, USA

**ABSTRACT:** Enterprise organizations increasingly rely on cloud-based human resources (HR) platforms to manage complex employee processes across multiple countries, business units, workforce populations, and regulatory environments. Benefits administration is one of the most integration-intensive areas of HR technology because it requires continuous coordination between HR systems, benefits providers, payroll platforms, insurance carriers, financial applications, identity services, and external regulatory or reporting systems. As workforce size and benefits complexity increase, traditional point-to-point integrations can become difficult to maintain, expensive to scale, and vulnerable to data inconsistencies.

This article presents a generalized framework for designing scalable benefits integration architectures around Workday and other enterprise HR technology ecosystems. The framework focuses on modular integration patterns, standardized data exchange, API-based connectivity, event-driven processing, batch interfaces, transformation and validation services, error handling, monitoring, security, and operational governance. Rather than treating benefits integration as a collection of independent interfaces, the proposed approach views the integration landscape as a reusable platform capable of supporting multiple benefit providers, workforce populations, jurisdictions, and downstream applications.

The article examines how integration architecture can address common challenges such as employee eligibility changes, enrollment events, dependent information, life-event processing, carrier file generation, payroll synchronization, reconciliation, duplicate transactions, failed transmissions, and changes in provider requirements. It also discusses strategies for separating business rules from interface-specific logic so that new providers and benefits programs can be incorporated without extensive redesign.

A scalable architecture is further evaluated from the perspectives of performance, reliability, security, maintainability, observability, and extensibility. The proposed framework combines centralized integration governance with loosely coupled processing components, enabling organizations to support both real-time and scheduled benefits transactions. Appropriate validation, auditability, encryption, access controls, and exception-management mechanisms are incorporated throughout the integration lifecycle.

The resulting framework provides a generalized architectural model for organizations seeking to modernize enterprise benefits integrations while reducing interface complexity and improving operational resilience. The approach can be applied not only to Workday-based environments but also to broader HR technology ecosystems where employee, benefits, payroll, financial, and third-party systems must exchange accurate and timely information at enterprise scale.

**KEYWORDS:** Workday, Benefits Integration, Enterprise HR Technology, HRIS, API Integration, Benefits Administration, Integration Framework, Event-Driven Architecture, Data Validation, Enterprise Integration, Payroll Integration, Carrier Integration, Cloud HR, Integration Monitoring

## I. INTRODUCTION

Enterprise human resources (HR) operations have evolved from centralized administrative systems into interconnected digital ecosystems that support employee lifecycle management, payroll, workforce planning, talent management, time tracking, and benefits administration. Cloud-based Human Capital Management (HCM) platforms such as Workday have become important components of these ecosystems because they provide a common platform for maintaining employee and organizational information. However, the value of a modern HCM platform depends not only on its internal capabilities but also on its ability to exchange reliable information with external and enterprise applications.

Benefits administration represents a particularly complex integration domain. Employee benefits may involve medical and dental insurance providers, retirement and savings platforms, life and disability insurance carriers, flexible

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

spending accounts, wellness providers, payroll systems, financial applications, and third-party administrators. Each participating system may have different data structures, communication protocols, processing schedules, validation requirements, and security controls. Consequently, benefits integration must coordinate a large number of transactions while preserving data accuracy and ensuring that changes are delivered to the appropriate downstream systems.

A typical benefits lifecycle can generate numerous integration events. An employee may become eligible for benefits after joining an organization, change coverage following a qualifying life event, add or remove dependents, modify an elected plan during an enrollment period, or terminate employment. Each change can require updates to one or more external providers. When these transactions are processed through tightly coupled or point-to-point interfaces, increasing the number of benefit plans and providers can significantly increase architectural complexity.

Scalability therefore becomes an important consideration in benefits integration design. A framework that works effectively for a small workforce may become inefficient when applied to a global organization with thousands or hundreds of thousands of employees. Higher transaction volumes can increase processing time, while changes in provider specifications can require modifications across multiple interfaces. Furthermore, failures become more difficult to identify when integrations lack centralized monitoring, correlation identifiers, standardized error handling, and reconciliation mechanisms.

Another challenge is the diversity of integration methods used within enterprise HR environments. Some benefits providers support modern REST or SOAP APIs, while others continue to depend on scheduled files, secure file transfers, or standardized carrier formats. An enterprise integration architecture must therefore support multiple communication patterns rather than assuming that every system can participate through a single technology. Batch processing may remain appropriate for high-volume carrier transmissions, whereas event-driven or API-based integration may be more suitable for transactions requiring near-real-time processing.

Data consistency is equally important. Employee information originating in the HCM platform may need to be transformed before being consumed by a benefits provider. Names, addresses, employment status, eligibility dates, plan identifiers, coverage levels, dependent relationships, and effective dates may all require transformation or validation. Incorrect mappings can result in rejected transactions, delayed enrollments, incorrect deductions, or discrepancies between HR and carrier records. A scalable framework must therefore treat data validation and transformation as explicit architectural capabilities rather than as incidental logic embedded within individual interfaces.

Security introduces an additional layer of complexity because benefits information can contain sensitive employee and dependent data. Integration services must protect information during transmission and processing while implementing appropriate authentication, authorization, encryption, access controls, logging, and retention mechanisms. Security requirements must be incorporated into the architecture from the beginning rather than added after interfaces have already been implemented.

The increasing adoption of cloud services and API-driven architectures also creates an opportunity to modernize benefits integration. Instead of building isolated interfaces for every provider, organizations can introduce reusable integration services, standardized canonical data models, centralized transformation components, and common monitoring mechanisms. Such an approach allows provider-specific requirements to remain within controlled adapter layers while maintaining consistent business and employee data models across the enterprise.

This article presents a generalized framework for scalable Workday benefits integrations that addresses these challenges through modular architecture, standardized data exchange, reusable transformation and validation capabilities, multiple integration patterns, centralized monitoring, security controls, and operational governance. The objective is not to prescribe a single implementation for every organization, but to establish architectural principles that can be adapted to different workforce sizes, benefits portfolios, provider ecosystems, and regulatory environments.

The remainder of the article examines the major components of this framework, including the enterprise benefits integration landscape, architectural requirements, integration patterns, data transformation and validation, scalability strategies, security, monitoring and reconciliation, implementation considerations, and future opportunities for intelligent automation. Through this approach, benefits integration can evolve from a collection of fragile interfaces into a scalable enterprise integration capability that supports reliable and maintainable HR operations.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

## II. ENTERPRISE BENEFITS INTEGRATION LANDSCAPE

Modern benefits administration operates within a distributed ecosystem rather than a single HR application. While Workday can act as the central system for worker, organizational, eligibility, and benefits information, multiple external systems participate in the complete benefits lifecycle. These systems exchange information in different formats and at different frequencies, creating an integration landscape that must be carefully designed to maintain consistency and reliability.

### 2.1 Core Systems in the Benefits Ecosystem

A typical enterprise benefits ecosystem consists of several interconnected layers. The HCM platform serves as a primary source for employee and benefits-related information. External benefits carriers consume eligibility and enrollment information and return transaction acknowledgments or exception information. Payroll systems use benefits elections to calculate employee deductions and employer contributions. Financial systems may consume aggregated benefits costs for accounting and reporting purposes.

Other systems can include identity and access management platforms, employee self-service applications, data warehouses, analytics platforms, regulatory reporting systems, and third-party benefits administrators. The number and type of participating systems vary by organization, but the architectural challenge remains similar: information must move consistently across systems with different technical and business requirements.

### 2.2 Benefits Data Flow

Benefits integration generally involves several stages of data movement. An employee or administrator first creates or changes benefits information within the HR platform. The integration layer identifies relevant changes, validates the information, transforms it into the required target representation, and delivers it to the appropriate downstream provider.

The receiving system then processes the transaction and may return an acknowledgment, success response, rejection, or error. This response must be correlated with the original transaction so that the organization can determine whether the employee's information was successfully synchronized.

A simplified lifecycle can therefore be represented as:
**Source HR Data → Change Detection → Validation → Transformation → Routing → Provider Transmission → Acknowledgment → Reconciliation**

This sequence demonstrates why benefits integration should be considered an end-to-end processing capability rather than simply a mechanism for transferring files.

### 2.3 Point-to-Point Integration Challenges

Historically, organizations frequently implemented benefits interfaces as individual point-to-point connections. Although this approach can be relatively simple when only a few systems are involved, complexity increases rapidly as the number of providers grows.

If each system requires custom transformation, authentication, error handling, monitoring, and communication logic, similar capabilities may be duplicated across many interfaces. A change in a common employee attribute can consequently require modifications to multiple integrations.

Point-to-point architectures can also make troubleshooting difficult. An operations team may need to inspect different logs, file-processing systems, middleware components, and provider responses to determine why a single employee transaction failed. The absence of common correlation identifiers and centralized monitoring further increases the operational burden.

### 2.4 Need for a Modular Integration Framework

A scalable framework separates common integration capabilities from provider-specific requirements. Common capabilities may include authentication, employee-data validation, transformation, routing, logging, retry processing, encryption, notification, and reconciliation. Provider-specific adapters can then focus primarily on the data format and communication requirements of individual carriers.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

This separation provides an important architectural advantage. When a new benefits provider is introduced, the organization can reuse the existing integration infrastructure instead of developing an entirely independent interface. Similarly, improvements to common capabilities such as monitoring or security can be implemented centrally and applied across multiple integrations.

### 2.5 Real-Time and Batch Integration

Benefits ecosystems typically require a combination of real-time, near-real-time, and batch processing.

**Real-time integration** is useful when immediate synchronization is important and the target provider exposes suitable APIs. Examples can include selected employee or eligibility updates.

**Near-real-time integration** can process events asynchronously while allowing some delay between the source transaction and downstream delivery. This model can improve scalability by decoupling source applications from external provider availability.

**Batch integration** remains useful for high-volume carrier processing. Scheduled files can group large numbers of transactions and transmit them at predefined intervals. Batch processing can also be appropriate when external providers require specific file formats or scheduled transmission windows.

A scalable architecture should therefore support multiple processing patterns instead of attempting to force every benefits transaction into a single integration model.

### 2.6 Integration as an Enterprise Capability

The increasing complexity of benefits ecosystems suggests that integration should be managed as an enterprise capability with standardized architectural principles. Rather than designing each interface independently, organizations can establish reusable patterns for data exchange, validation, transformation, security, monitoring, exception management, and reconciliation.

Such standardization provides a foundation for extending benefits integrations as the organization grows. New providers, plans, geographic regions, and workforce populations can be incorporated with less disruption when the underlying framework already provides common services.

**Table 1. Typical Components of an Enterprise Benefits Integration Ecosystem**

| Component | Primary Responsibility | Typical Integration Requirement |
|---|---|---|
| HCM Platform | Maintains worker and benefits information | Source data and business events |
| Integration Layer | Coordinates data movement | Routing, transformation, orchestration |
| Benefits Carrier | Processes coverage and eligibility | API or carrier file |
| Payroll Platform | Calculates deductions and contributions | Benefits-to-payroll synchronization |
| Financial System | Supports accounting and reporting | Aggregated financial data |
| Data/Analytics Platform | Provides reporting and analysis | Batch or API data ingestion |
| Identity Services | Supports authentication and access | Secure identity exchange |
| Monitoring Platform | Tracks integration health | Logs, metrics, alerts |
| Reconciliation Services | Compares source and target states | Transaction and exception matching |

The enterprise benefits integration landscape therefore establishes the foundation for the architecture discussed in the following sections. A scalable framework must accommodate heterogeneous systems while providing consistent mechanisms for processing, validating, securing, monitoring, and reconciling benefits information across the entire ecosystem.

## III. ARCHITECTURAL REQUIREMENTS FOR SCALABLE BENEFITS INTEGRATION

A scalable benefits integration framework must satisfy more than basic connectivity requirements. Enterprise HR environments generate frequent employee and benefits changes, operate across heterogeneous applications, and require

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

strong controls for data accuracy, security, availability, and auditability. The architecture must therefore address both functional integration requirements and non-functional characteristics such as scalability, resilience, maintainability, and observability.

### 3.1 Functional Requirements

The primary functional requirement is the reliable movement of benefits-related information between Workday, benefits providers, payroll platforms, and other enterprise applications. The framework should support employee eligibility, enrollment elections, dependent information, coverage changes, employment status changes, and termination events.

The integration layer should be capable of identifying relevant changes and determining which downstream systems require updates. This prevents unnecessary transmission of unchanged information and reduces processing overhead.

A generalized processing sequence is:
**Identify Change → Determine Eligibility → Validate Data → Transform Data → Route Transaction → Deliver → Capture Response → Reconcile**

Each stage should have a well-defined responsibility so that failures can be isolated and processed without affecting unrelated transactions.

### 3.2 Scalability

Scalability is one of the most important requirements for enterprise benefits integration. Transaction volumes can increase significantly during annual enrollment periods, organizational restructuring, mergers, acquisitions, or large workforce changes.

A scalable architecture should support horizontal processing, asynchronous workloads, queue-based buffering, and configurable batch sizes where appropriate. Processing capacity should be capable of increasing independently from the source HR platform.

For example, instead of processing thousands of employee changes sequentially, transactions can be divided into manageable processing units. Independent workers can then process these units concurrently while maintaining appropriate ordering requirements where business rules demand it.

Scalability should also account for future expansion. The architecture should accommodate additional benefit providers, countries, employee populations, and benefits products without requiring fundamental redesign.

### 3.3 Interoperability

Enterprise benefits ecosystems frequently contain applications based on different technologies. One provider may expose REST APIs, another may use SOAP services, while another may require encrypted files delivered through secure file transfer.

The integration framework should therefore abstract communication mechanisms from business processing. A common internal representation can be transformed into the specific format required by each target system.

This approach enables the framework to support multiple integration protocols while maintaining a consistent internal processing model.

### 3.4 Data Quality and Validation

Benefits transactions are highly sensitive to data quality. Incorrect employee identifiers, invalid effective dates, missing dependent information, incorrect plan codes, or inconsistent eligibility attributes can lead to downstream rejection or incorrect coverage.

Validation should occur before transmission to external providers. Typical validation categories include:

- Mandatory-field validation
- Employee and dependent identifier validation
- Date and effective-period validation
- Benefits-plan validation

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

- Eligibility-rule validation
- Data-format validation
- Duplicate-transaction detection
- Referential integrity validation

Separating validation from provider-specific transmission logic makes these controls reusable across multiple integrations.

### 3.5 Reliability and Fault Tolerance

External systems cannot always be assumed to be available. Carrier APIs may experience temporary outages, scheduled maintenance, network failures, or throttling. A scalable integration framework should therefore be designed to tolerate transient failures.

Retry mechanisms should distinguish between temporary and permanent failures. A temporary network or service-availability problem may be retried automatically, while an invalid employee record may require business-data correction rather than repeated transmission.

Dead-letter or exception queues can be used to isolate transactions that cannot be processed successfully. This prevents a small number of problematic records from blocking an entire batch.

### 3.6 Idempotency and Duplicate Prevention

Duplicate processing is a significant concern in benefits integration. A transaction may be retried after a timeout even though the target system has already received and processed it. Without appropriate controls, the same enrollment or eligibility event could be submitted multiple times.

Idempotency mechanisms can use transaction identifiers, employee identifiers, event timestamps, version numbers, or business keys to determine whether a transaction has already been processed.

A generalized idempotent model can be represented as:
**Unique Transaction Key → Processing Status → Duplicate Check → Execute or Skip**
Maintaining transaction history also supports auditability and operational investigation.

### 3.7 Security and Privacy

Benefits information can contain sensitive employee and dependent information. Security must therefore be incorporated across the complete integration lifecycle.
Important controls include:

- Encryption during data transmission
- Encryption of sensitive information at rest
- Strong authentication mechanisms
- Role-based or attribute-based authorization
- Least-privilege access
- Secure credential and secret management
- Controlled administrative access
- Audit logging
- Data-retention controls
- Protection against unauthorized data exposure

Sensitive information should also be excluded or masked in application logs wherever possible.

### 3.8 Observability and Monitoring

A large integration landscape requires centralized visibility. Monitoring should provide more than an indication that an interface executed successfully. It should provide information about transaction volume, processing latency, failure rates, rejected records, retries, provider availability, and reconciliation status.
A useful monitoring model can combine:
**Logs + Metrics + Traces + Alerts + Business-Level Transaction Status**
Correlation identifiers should be maintained throughout the transaction lifecycle. This allows an operations team to trace an employee-related transaction from its source event through transformation and delivery to the final provider response.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

### 3.9 Auditability and Traceability

Benefits transactions may require investigation months after their original processing. The framework should therefore retain appropriate transaction metadata, processing timestamps, source references, target references, response information, and exception details.

Audit records should provide enough information to answer questions such as:

1. What change was received?
2. When was it received?
3. Which validation rules were applied?
4. Which transformation was performed?
5. Which provider received the transaction?
6. What response was returned?
7. Was the transaction retried?
8. Was reconciliation completed?

This level of traceability reduces investigation time and strengthens operational governance.

### 3.10 Maintainability and Extensibility

Benefits providers can change their file formats, APIs, business rules, authentication mechanisms, and transmission schedules. The architecture should therefore minimize the impact of such changes.

Reusable components, configuration-driven mappings, standardized interfaces, modular provider adapters, and centralized business rules can reduce maintenance effort.

The architecture should also support incremental modernization. Organizations should be able to introduce API-based integrations, event-driven processing, or improved monitoring without requiring every existing batch interface to be replaced simultaneously.

### 3.11 Performance and Operational Efficiency

Performance should be evaluated at multiple levels, including transaction-processing time, queue latency, transformation performance, provider response time, and end-to-end completion time.

Performance optimization should not focus solely on raw throughput. Benefits processing also requires predictable completion within business deadlines, particularly during enrollment periods and payroll processing windows.

Accordingly, performance requirements should be defined using measurable indicators such as:

- Transactions processed per unit of time
- Average and maximum processing latency
- Percentage of successful transactions
- Retry volume
- Error-processing time
- Reconciliation completion time

These metrics provide a foundation for capacity planning and continuous improvement.

### 3.12 Governance Requirements

Finally, enterprise benefits integrations require architectural and operational governance. Governance should define standards for naming, interface ownership, data classification, authentication, logging, error handling, change management, testing, deployment, and retirement.

A governance model prevents individual integrations from evolving into isolated technical solutions and ensures that new interfaces follow established enterprise standards.

Together, these requirements establish the foundation for a scalable benefits integration framework. The next section can examine the **reference architecture and major architectural components** used to satisfy these requirements.

## IV. REFERENCE ARCHITECTURE FOR SCALABLE WORKDAY BENEFITS INTEGRATION

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

A scalable benefits integration framework should provide a structured architecture that separates employee data management, integration orchestration, transformation, provider connectivity, monitoring, and operational governance. Such separation reduces coupling between systems and allows individual components to evolve without requiring extensive changes across the entire integration landscape.

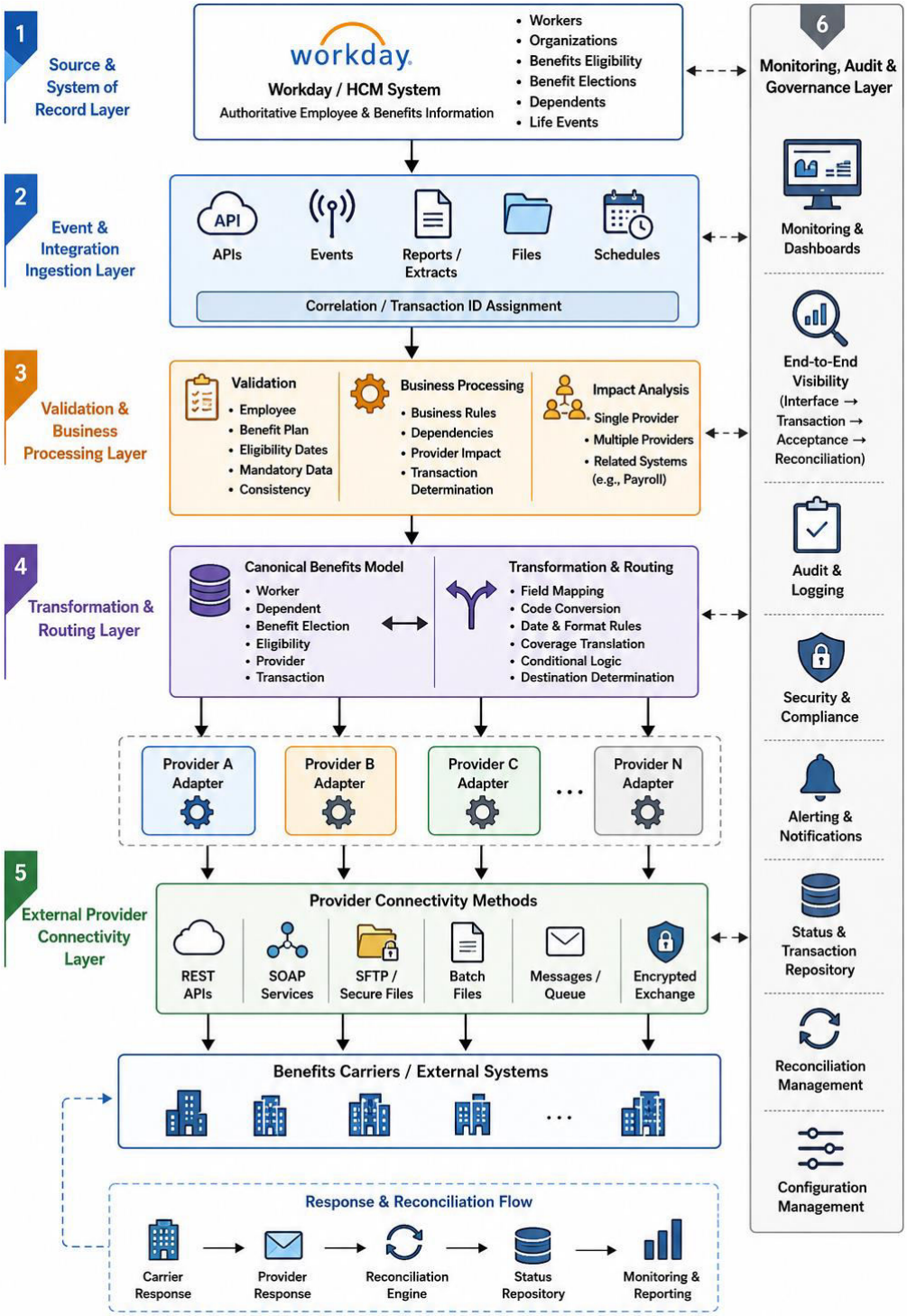
The reference architecture proposed in this section is a generalized model that can be adapted to Workday-centered HR environments and other enterprise HCM ecosystems.

**Fig: Scalable Workday Benefits Integration Reference Architecture**

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078




### 4.1 Architectural Layers

The proposed architecture consists of six logical layers:

1. **Source and System-of-Record Layer**
2. **Event and Integration Ingestion Layer**
3. **Validation and Business Processing Layer**
4. **Transformation and Routing Layer**
5. **External Provider Connectivity Layer**
6. **Monitoring, Audit, and Governance Layer**

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

These layers establish clear boundaries between business data and technical integration mechanisms.

**4.1.1 Source and System-of-Record Layer**

The source layer contains Workday or another enterprise HCM platform that maintains worker, organizational, eligibility, and benefits information.

Typical source information includes:

- Worker identifiers
- Employment status
- Organizational assignment
- Benefits eligibility
- Benefit plan elections
- Coverage levels
- Dependent information
- Effective and termination dates
- Life-event information

The HCM platform should remain responsible for authoritative employee and benefits information, while the integration framework should focus on reliably distributing required information to downstream systems.

**4.1.2 Event and Integration Ingestion Layer**

The ingestion layer detects or receives changes from the source platform. Depending on the integration model, changes may arrive through APIs, events, scheduled extracts, reports, files, or other supported mechanisms.

An event-driven approach can provide efficient processing because only relevant changes need to enter the downstream workflow. Batch ingestion can remain appropriate where provider requirements or business schedules require periodic processing.

The ingestion layer should assign a correlation or transaction identifier to each logical transaction. This identifier can then be propagated throughout subsequent processing stages.

**4.1.3 Validation and Business Processing Layer**

The validation layer evaluates whether incoming information satisfies technical and business requirements before it is transmitted externally.

Validation can include checking employee identifiers, benefit-plan codes, eligibility dates, dependent relationships, mandatory attributes, and transaction consistency.

Business processing may also determine whether a particular change requires transmission to one provider or multiple providers.

For example, a change in an employee's medical plan may require a carrier update and a corresponding payroll deduction change. The processing layer can identify these dependencies without embedding provider-specific communication logic into the business rules.

**4.1.4 Transformation and Routing Layer**

The internal representation of an employee benefits transaction may differ substantially from the format expected by an external provider.

The transformation layer converts standardized internal data into provider-specific representations. Mapping logic can include field transformation, code conversion, date formatting, coverage-level translation, and conditional attribute generation.

Routing determines the appropriate downstream destination based on factors such as:

- Benefit type
- Provider
- Employee population
- Geographic region

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

- Transaction type
- Effective date
- Provider communication method

Separating transformation and routing from business processing improves maintainability and allows provider-specific changes to be isolated.

**4.1.5 External Provider Connectivity Layer**

This layer provides connectivity to external benefits carriers and related applications.
It may support multiple communication patterns, including:

- REST APIs
- SOAP services
- Secure file transfer
- Scheduled batch files
- Encrypted file exchange
- Message-based communication

A provider adapter can encapsulate the technical requirements of each external system. The rest of the architecture can then interact with a consistent internal interface rather than directly managing every provider's protocol.

**4.1.6 Monitoring, Audit, and Governance Layer**

Cross-cutting capabilities should operate across all architectural layers. These include monitoring, logging, security, auditability, alerting, configuration management, and operational reporting.

The monitoring layer should capture both technical and business-level information. For example, a technically successful file transmission does not necessarily mean that every employee transaction was accepted by the provider.

Therefore, monitoring should distinguish between:
**Interface Success → Transaction Success → Provider Acceptance → Reconciliation Success**
This distinction is essential for accurate operational visibility.

**4.2 High-Level Processing Model**

The complete processing model can be represented as:
**Workday → Ingestion → Validation → Business Rules → Transformation → Routing → Provider Adapter → Benefits Carrier**
The response path can then return through:
**Benefits Carrier → Provider Response → Reconciliation → Status Repository → Monitoring and Reporting**
This bidirectional model allows the framework to track both outbound transactions and downstream responses.

**4.3 Canonical Data Model**

A canonical benefits data model can reduce the dependency between source and target systems. Instead of creating unique mappings between every source and provider combination, the architecture establishes a standardized intermediate representation.

For example:
**Workday Data → Canonical Benefits Model → Provider A Format**
and:
**Workday Data → Canonical Benefits Model → Provider B Format**
This pattern reduces the number of direct mappings and simplifies the onboarding of new providers.
The canonical model can contain common entities such as:

| Entity | Representative Attributes |
|---|---|
| Worker | Worker ID, employment status, hire date |
| Dependent | Dependent ID, relationship, eligibility |
| Benefit Election | Plan, coverage level, effective date |
| Eligibility | Eligibility status, eligibility date |

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

| Entity | Representative Attributes |
|---|---|
| Provider | Provider ID, interface type |
| Transaction | Transaction ID, event type, processing status |

The canonical model should contain only information that provides meaningful reuse. Excessive abstraction can create unnecessary complexity, so provider-specific attributes should remain within appropriate adapter layers.

**4.4 Asynchronous Processing**

Asynchronous processing can improve resilience and scalability by separating transaction creation from transaction delivery.

Instead of requiring the source system to wait for an external carrier response, a transaction can be placed into a processing queue. Integration workers can retrieve and process transactions independently.

The conceptual model is:

**Source Event → Queue → Processing Workers → Provider → Response Queue → Reconciliation**

This approach provides buffering during periods of high transaction volume and reduces direct dependency on external provider availability.

**4.5 Resilience through Isolation**

A major architectural principle is failure isolation. A provider outage should not prevent unrelated transactions from continuing.

For example, if a dental carrier becomes temporarily unavailable, transactions destined for medical and retirement providers should continue processing independently where business dependencies permit.

Provider-specific queues, retry mechanisms, circuit-breaking strategies, and dead-letter processing can help isolate failures.

This design prevents localized provider problems from becoming enterprise-wide integration failures.

**4.6 Configuration-Driven Integration**

Whenever possible, integration behavior should be controlled through configuration rather than hard-coded logic.

Configuration can define:

- Provider identifiers
- Plan mappings
- File naming conventions
- Processing schedules
- API endpoints
- Retry policies
- Transformation rules
- Validation parameters
- Notification thresholds

Configuration-driven design allows operational teams to introduce controlled changes without modifying core processing components.

**4.7 Reference Architecture Benefits**

The proposed reference architecture provides several benefits:

- **Reduced coupling:** Source and provider systems are separated through integration services.
- **Improved scalability:** Processing workloads can be distributed independently.
- **Reusable components:** Validation, monitoring, security, and audit capabilities can be shared.
- **Simplified provider onboarding:** New carriers can be integrated through standardized adapters.
- **Improved resilience:** Failures can be isolated by provider or transaction type.
- **Better observability:** End-to-end transaction status can be monitored.
- **Greater maintainability:** Provider-specific changes remain localized.
- **Improved governance:** Common architectural and security standards can be applied consistently.

The reference architecture therefore provides a foundation for implementing scalable benefits integrations while accommodating both modern API-based connectivity and established batch-processing models. The next section can examine the **major integration patterns used for Workday benefits processing**, including API-based, batch, event-driven, and hybrid approaches.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

## V. INTEGRATION PATTERNS FOR WORKDAY BENEFITS PROCESSING

Benefits integration environments rarely operate through a single communication mechanism. Different carriers and enterprise applications impose different technical and operational requirements. A scalable framework should therefore support multiple integration patterns while maintaining common principles for security, validation, monitoring, error handling, and reconciliation.

The major patterns include API-based integration, batch file processing, event-driven integration, scheduled extraction, message-based processing, and hybrid architectures.

### 5.1 API-Based Integration

API-based integration enables applications to exchange information through defined service interfaces. REST APIs are increasingly used where benefits providers support modern service-based connectivity, while SOAP-based services may continue to exist in established enterprise environments.

A typical API transaction follows:

**Workday → Integration Service → Transformation → Provider API → Response → Status Update**

API-based processing is particularly useful when information must be delivered with relatively low latency. For example, selected eligibility or enrollment changes can be transmitted soon after the source transaction becomes available.

However, API integrations must account for provider-specific limitations such as request quotas, response-time constraints, authentication requirements, and temporary service outages. Retry and throttling mechanisms should therefore be incorporated into the integration design.

### 5.2 Batch File Integration

Batch processing remains an important pattern for benefits administration because many external providers continue to support scheduled file exchanges.

A batch workflow can be represented as:

**Workday Extract → Validation → Transformation → File Generation → Encryption → Secure Transfer → Provider Processing → Acknowledgment**

Batch integration is particularly suitable for large transaction volumes because many records can be processed together. It also provides predictable processing windows, which can be useful for organizations that coordinate benefits processing with payroll cycles or carrier schedules.

The primary challenge is latency. A change may not reach the provider until the next scheduled processing window. Batch processes also require strong controls for file completeness, duplicate prevention, encryption, naming conventions, and acknowledgment processing.

### 5.3 Event-Driven Integration

Event-driven integration enables downstream processing to begin when a relevant business event occurs.

Examples of potential events include:

- Worker hired
- Worker terminated
- Benefits eligibility changed
- Benefits election updated
- Dependent added
- Dependent removed
- Qualifying life event recorded

An event-driven architecture can reduce unnecessary processing because only relevant changes are propagated.

A simplified model is:

**Business Event → Event Broker → Benefits Processing Service → Provider Adapter**

The event itself should contain sufficient information to identify the transaction while avoiding unnecessary exposure of sensitive information.

### 5.4 Message-Based Integration

Message-oriented processing introduces an intermediate messaging layer between producers and consumers.

This pattern is valuable when the processing volume is unpredictable or when temporary downstream failures are expected. Messages can remain available until a processing component is ready to consume them.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

For example, during annual enrollment, the volume of benefits transactions may increase significantly. A message queue can absorb the temporary increase and allow processing workers to consume transactions at a controlled rate.
Message-based architectures can also support retry processing, dead-letter queues, prioritization, and workload distribution.

**5.5 Scheduled Extraction**

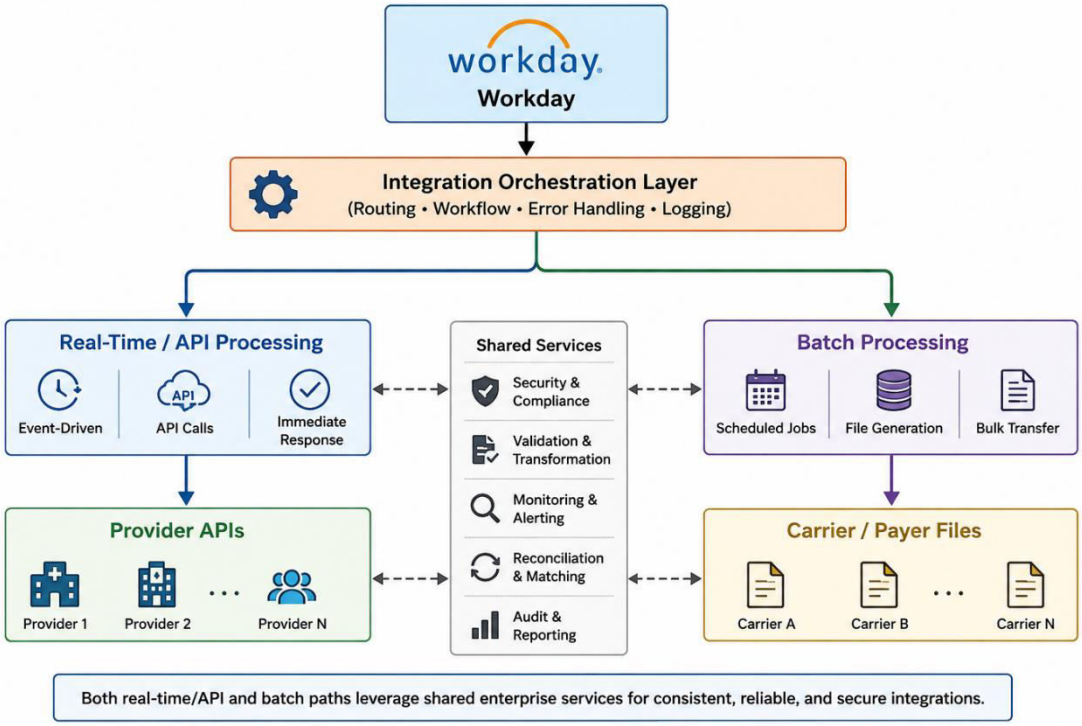
Some benefits processes do not require event-level processing. Scheduled extraction can retrieve employee and benefits information at predefined intervals.
Scheduled processing can be useful for:

- Daily eligibility files
- Periodic enrollment extracts
- Provider reconciliation
- Payroll synchronization
- Reporting feeds
- Historical data exchanges

The schedule should be aligned with business requirements and provider processing windows.

**5.6 Hybrid Integration Pattern**

In practice, enterprise organizations may benefit most from a hybrid model.
For example, high-priority employee changes could use event-driven or API-based processing, while high-volume provider exchanges could continue using scheduled files.
A hybrid architecture may therefore look like:




Both paths can share centralized validation, transformation, security, monitoring, and reconciliation services.
This approach allows organizations to modernize gradually rather than replacing all existing integrations simultaneously.

**5.7 Publish-and-Subscribe Pattern**

A publish-and-subscribe model can be useful when a single benefits event needs to be consumed by multiple downstream systems.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

For example, a benefits-election change could potentially be relevant to:

- A medical carrier
- A payroll system
- A benefits data warehouse
- An employee analytics platform

Instead of creating independent source-to-target integrations, the source event can be published once and consumed by multiple subscribers.

This pattern reduces direct dependencies and allows additional consumers to be introduced without modifying the original event producer.

**5.8 Request-Response Versus Asynchronous Processing**

API integrations often use request-response communication, where the source waits for the provider response. This is straightforward but can create dependency on external response time.

Asynchronous processing separates submission from completion:

**Submit Transaction → Receive Processing Acknowledgment → Continue Processing → Receive Final Status**

This approach can provide greater resilience for long-running or high-volume workflows.

The appropriate pattern depends on the provider's capabilities, business latency requirements, transaction volume, and error-handling model.

**5.9 Pattern Selection Criteria**

The selection of an integration pattern should be based on business and technical requirements rather than technology preference.

| Requirement | Preferred Pattern |
|---|---|
| Low-latency transaction | API or event-driven |
| High-volume scheduled exchange | Batch |
| Temporary provider unavailability | Asynchronous messaging |
| Multiple downstream consumers | Publish-subscribe |
| Predictable daily processing | Scheduled extraction |
| Mixed provider capabilities | Hybrid |
| Long-running processing | Asynchronous workflow |
| Large enrollment-period workload | Queue-based batch processing |

No single pattern is universally optimal. A scalable framework should provide reusable architectural patterns and allow individual integrations to select the mechanism that best matches their operational requirements.

**5.10 Standardization Across Integration Patterns**

Although communication mechanisms may differ, common capabilities should remain standardized. API, batch, and event-driven integrations should use consistent approaches for:

- Data validation
- Transformation
- Authentication
- Encryption
- Correlation IDs
- Logging
- Error classification
- Retry processing
- Monitoring
- Reconciliation
- Audit trails

This standardization is critical because scalability depends not only on processing capacity but also on the ability of engineering and operations teams to manage a growing integration portfolio consistently.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

A well-designed combination of API, batch, event-driven, and messaging patterns allows enterprise HR organizations to accommodate heterogeneous benefits providers while maintaining a common architectural foundation. The next section can focus on **data transformation, validation, and canonical data modeling**, which are central to ensuring accurate benefits transactions across these diverse integration patterns.

## VI. DATA TRANSFORMATION, VALIDATION, AND CANONICAL MODELING

Reliable benefits integration depends heavily on the quality and consistency of data exchanged between the HR platform and downstream providers. Even when an integration is technically operational, differences in data structures, business terminology, coding standards, date formats, and validation rules can result in rejected or incorrectly processed transactions. A scalable framework should therefore treat data transformation and validation as dedicated architectural capabilities.

### 6.1 Importance of Data Standardization

Workday and external benefits providers may represent the same business concept differently. For example, an internal benefits plan identifier may not correspond directly to the identifier used by a carrier. Similarly, employee status, coverage level, relationship type, and effective dates may use different representations across systems.

Without a standardized approach, transformation logic can become embedded separately within every interface. This increases maintenance effort and makes it difficult to apply consistent business rules.

A canonical data model provides an intermediate representation that allows source information to be standardized before it is distributed to individual providers.

The generalized flow is:
**Workday Data → Canonical Benefits Model → Provider-Specific Transformation → External System**
This approach separates enterprise-level business information from provider-specific technical requirements.

### 6.2 Canonical Benefits Data Model

A canonical model can represent the major entities involved in benefits processing.

**Table 2. Representative Canonical Benefits Data Model**

| Entity | Example Attributes | Purpose |
|---|---|---|
| Worker | Worker ID, status, hire date | Identifies employee |
| Organization | Company, business unit, location | Provides organizational context |
| Dependent | Dependent ID, relationship, eligibility | Represents covered dependents |
| Benefit Plan | Plan ID, category, provider | Identifies selected benefit |
| Election | Coverage level, effective date, election status | Represents employee selection |
| Eligibility | Eligibility status, eligibility date | Determines benefit qualification |
| Provider | Provider ID, interface type | Identifies destination |
| Transaction | Transaction ID, event type, timestamp | Tracks processing |
| Response | Status, response code, message | Records downstream result |

The canonical model should remain stable even when individual provider specifications change.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

### 6.3 Data Mapping

Data mapping establishes relationships between source attributes, canonical attributes, and target attributes.
For example:
**Source Attribute → Canonical Attribute → Provider Attribute**
An employee's internal worker identifier may be mapped to a standardized worker identifier and then transformed into the identifier format required by a specific carrier.
Mappings may involve direct assignments, conditional transformations, code translations, concatenation, splitting, or derived values.
Configuration-driven mapping is preferable where practical because it reduces the need to modify application code for routine mapping changes.

### 6.4 Validation Framework

Validation should occur at multiple stages of processing.
**Level 1: Structural Validation**
Structural validation verifies that the incoming message or file conforms to the expected technical format.
Examples include:

- Required fields
- Data types
- Field lengths
- Valid character sets
- File structure
- Message schema

**Level 2: Business Validation**
Business validation determines whether the information is logically acceptable.
Examples include:

- Employee must be active or eligible for the selected benefit.
- Effective dates must satisfy applicable business rules.
- A dependent relationship must be valid.
- Coverage elections must correspond to an available plan.
- Required information must be present for the selected coverage.

**Level 3: Provider Validation**
Provider-specific validation confirms that the transaction satisfies the target system's requirements.
For example, a carrier may require a particular plan code, coverage value, transaction type, or identifier format.
Separating provider validation from enterprise-wide validation prevents provider-specific rules from unnecessarily affecting other integrations.

### 6.5 Validation Outcomes

Validation results should be classified rather than simply marked as successful or failed.
A useful classification includes:

- **Valid:** Transaction can proceed.
- **Warning:** Transaction can proceed but requires attention.
- **Business Error:** Data requires correction.
- **Technical Error:** Processing infrastructure encountered a problem.
- **Provider Error:** Target system rejected the transaction.
- **Duplicate:** Transaction has already been processed.

This classification improves automated handling and operational reporting.

### 6.6 Effective-Date Management

Benefits transactions are highly dependent on dates. A transaction can be technically valid while still producing an incorrect business result if its effective date is incorrect.
Important dates can include:

- Hire date
- Eligibility date
- Election date

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

- Coverage effective date
- Life-event date
- Termination date
- Coverage end date
- Payroll deduction effective date

The integration framework should consistently represent dates and apply appropriate timezone and formatting rules when exchanging information with external systems.

**6.7 Duplicate Detection**

Duplicate transactions can occur because of retries, repeated source events, file regeneration, or manual reprocessing.
A unique transaction key can be generated using one or more business attributes such as:
**Worker ID + Benefit Plan + Event Type + Effective Date + Source Event ID**
The exact key should be determined according to the business semantics of the transaction.
Before processing, the framework can check whether an equivalent transaction has already been completed. This provides an additional safeguard against duplicate enrollments and repeated updates.

**6.8 Data Transformation Pipeline**

A reusable transformation pipeline can be organized into distinct stages:
**Normalize → Validate → Enrich → Map → Format → Verify → Deliver**
Normalization establishes consistent internal representations. Validation identifies invalid data. Enrichment adds required contextual information. Mapping translates canonical attributes into target-specific fields. Formatting prepares the final API message or file. A final verification stage ensures that the generated transaction satisfies target requirements before transmission.
This separation makes transformation logic easier to test and maintain.

**6.9 Exception Handling**

Invalid data should not automatically cause an entire batch to fail. Where business rules allow, individual records can be isolated while valid records continue processing.
For example, if 9,950 of 10,000 transactions are valid, the framework should ideally allow the valid transactions to proceed while routing the remaining 50 records to an exception workflow.
Exception records should include sufficient information for investigation, such as:

- Transaction ID
- Worker reference
- Processing stage
- Error category
- Error description
- Timestamp
- Provider
- Retry status

Sensitive employee information should be minimized or masked in operational logs.

**6.10 Reusable Validation Services**

Common validation capabilities should be implemented as reusable services rather than duplicated across individual integrations.
Examples include:

- Employee identifier validation
- Date validation
- Plan-code validation
- Eligibility validation
- Dependent validation
- Duplicate detection
- Required-field validation

Reusable services provide consistent behavior across carriers and reduce maintenance effort.

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

**6.11 Data Quality Feedback Loop**

Validation should not be limited to preventing bad transactions. Exception information can also be analyzed to identify recurring data-quality problems.

For example, repeated failures involving a particular plan code may indicate an outdated mapping rather than individual employee errors. Similarly, frequent missing dependent information may indicate a source-system data-entry or business-process issue.

A continuous feedback loop can therefore be established:

**Transaction Processing → Validation → Exception Analysis → Root Cause Identification → Data/Rule Improvement → Improved Processing**

This transforms integration monitoring into a mechanism for improving overall HR data quality.

**6.12 Benefits of a Standardized Data Layer**

A standardized transformation and validation architecture provides several benefits:

- Reduced provider-specific coupling
- Improved data quality
- Reusable business rules
- Easier provider onboarding
- Simplified troubleshooting
- Consistent validation
- Improved auditability
- Lower maintenance effort
- Greater adaptability to changing provider requirements

Data transformation and validation therefore form a critical foundation for scalable benefits integration. Once transactions have been standardized and validated, the architecture can focus on reliably processing increasing workloads. The next section will examine **scalability, performance optimization, and high-volume benefits processing**, including annual enrollment and other peak transaction scenarios.

## VII. CONCLUSION

Enterprise benefits administration increasingly depends on reliable integration between cloud-based HCM platforms, benefits carriers, payroll systems, financial applications, analytics platforms, and other external services. As organizations expand their workforce, benefits portfolios, and provider ecosystems, conventional point-to-point integrations become increasingly difficult to scale and maintain.

This article presented a generalized framework for designing scalable Workday benefits integrations around modular architecture, standardized data exchange, validation, transformation, routing, multiple integration patterns, security, monitoring, and reconciliation. The proposed approach separates common integration capabilities from provider-specific requirements, allowing organizations to introduce new carriers and benefits programs without redesigning the complete integration landscape.

A key principle of the framework is the use of a canonical benefits data model. Standardizing employee, dependent, eligibility, election, provider, and transaction information creates a stable internal representation between the HCM platform and downstream systems. Provider-specific mappings can then be isolated within adapter or transformation components. This reduces coupling and supports easier maintenance when external provider specifications change.

The framework also recognizes that enterprise benefits processing requires multiple communication models. API-based integrations can support low-latency transactions, while batch processing remains appropriate for high-volume scheduled carrier exchanges. Event-driven and message-based architectures can provide additional scalability and resilience by decoupling transaction generation from downstream processing. A hybrid approach allows organizations to combine these patterns according to business requirements rather than forcing every integration into a single technical model.

Data validation and exception management are equally important. A technically successful transmission does not necessarily indicate successful benefits processing. The framework therefore emphasizes transaction-level validation, provider acknowledgments, retry mechanisms, duplicate detection, exception queues, and reconciliation. These

| ISSN: 2394-2975 | www.ijarety.in| | Impact Factor: 8.152| A Bi-Monthly, Double-Blind Peer Reviewed & Refereed Journal |

|| Volume 12, Issue 2, March-April 2025 ||

DOI:10.15680/IJARETY.2025.1202078

capabilities provide visibility into the complete transaction lifecycle and help prevent small data-quality problems from becoming larger operational issues.

Security and governance must also be treated as architectural concerns. Benefits integrations can process sensitive employee and dependent information, requiring appropriate authentication, authorization, encryption, auditability, and privacy controls. NIST Cybersecurity Framework 2.0 emphasizes governance, supply-chain considerations, and enterprise cybersecurity risk management, while NIST SP 800-53 Rev. 5 provides a broad catalog of security and privacy controls that can inform implementation decisions.

Overall, scalability should not be interpreted solely as the ability to process a larger number of transactions. A truly scalable benefits integration framework must also be easier to extend, monitor, secure, troubleshoot, and govern as the enterprise grows. By combining modular integration services, reusable validation and transformation components, asynchronous processing, standardized data models, centralized observability, and strong governance, organizations can evolve their benefits integration landscape from a collection of independent interfaces into a resilient enterprise integration capability.

Future implementations can further extend this framework through intelligent exception classification, predictive monitoring, automated reconciliation, configuration-driven onboarding, and AI-assisted analysis of integration failures. Such capabilities can reduce manual operational effort while improving the reliability and adaptability of enterprise HR technology ecosystems.

## REFERENCES

[1] C. Pascoe, S. Quinn, and K. Scarfone, *The NIST Cybersecurity Framework (CSF) 2.0*, NIST Cybersecurity White Paper NIST CSWP 29, National Institute of Standards and Technology, Gaithersburg, MD, Feb. 2024.

[2] Workday, Inc., *Annual Report on Form 10-K for the Fiscal Year Ended January 31, 2024*, Workday, Inc., Pleasanton, CA, 2024.

[3] Workday, Inc., *Annual Report on Form 10-K for the Fiscal Year Ended January 31, 2023*, Workday, Inc., Pleasanton, CA, 2023.

[4] National Institute of Standards and Technology, *NIST Cybersecurity Framework 2.0 Initial Public Draft*, NIST Cybersecurity White Paper CSWP 29, Gaithersburg, MD, 2023. The final CSF 2.0 was subsequently published in 2024.

[5] Workday, Inc., *Annual Report on Form 10-K for the Fiscal Year Ended January 31, 2022*, Workday, Inc., Pleasanton, CA, 2022.

[6] Joint Task Force, *Assessing Security and Privacy Controls in Information Systems and Organizations: Building Effective Assessment Plans*, NIST Special Publication 800-53A Revision 5, National Institute of Standards and Technology, Jan. 2022.

[7] Workday, Inc., *Annual Report on Form 10-K for the Fiscal Year Ended January 31, 2021*, Workday, Inc., Pleasanton, CA, 2021.

[8] National Institute of Standards and Technology, *Security and Privacy Controls for Information Systems and Organizations*, NIST Special Publication 800-53 Revision 5, updated 2021, National Institute of Standards and Technology, Gaithersburg, MD.

[9] R. Ross and V. Pillitteri, *Security and Privacy Controls for Information Systems and Organizations*, NIST Special Publication 800-53 Revision 5, National Institute of Standards and Technology, Gaithersburg, MD, Sep. 2020.

IJARETY

# International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975 Impact Factor: 8.152

www.ijarety.in editor.ijarety@gmail.com