frends

Enterprise Migration Approach

Table of Contents

1. Executive summary

This whitepaper provides a comprehensive strategy for enterprises migrating from their current integration platform to Frends. The document is meant to offer practical guidance and addresses all critical dimensions of the migrating process, including technical implementation, organizational change management, process optimization, stakeholder engagement and risk mitigation.

With this migration strategy, enterprises ensure business continuity while maximizing the benefits of the new platform. Frends is a modern, cloud-native integration platform as a service (iPaaS) with robust capabilities for enterprise integration, API management and workflow automation.

1.1 Key Objectives

Migrating from any integration platform is a complex process that requires careful planning, which is why Frends created this list of key objectives. From planning the migration process to onboarding your new iPaaS, this what companies should aim for:

- Ensure seamless transition with minimal disruption to business operations
- Maintain data integrity and integration reliability throughout the migration
- Upskill teams and establish sustainable operational practices
- Mitigate risks through careful planning and phased implementation
- Achieve improved performance, maintainability and cost efficiency

1.2 Expected outcomes

The transition to Frends is more than a platform replacement. Choosing the right platform is a strategic modernization of an organization's digital backbone. By following this migration strategy, companies can expect to achieve significant gains, such as:

• Superior cost-effectiveness

Organizations typically achieve a 30-75% lower Total Cost of Ownership (TCO) compared to traditional middleware. This is driven by a transparent, flat-tier pricing model that eliminates per-connector or per-message fees.

• Higher operational visibility

The platform provides real-time monitoring and a unified view of all data flows, reducing troubleshooting time by up to 60-70% and ensuring high availability across hybrid environments.

• Increased AI adoption

Companies gain access to Agentic AI capabilities, allowing for autonomous workflows, intelligent exception management and AI-powered process automation that future-proofs the integration landscape.

• Accelerated time-to-value

Frends enables 5x faster integration development compared to custom coding and up to 55% faster project delivery through its low-code, visual BPMN 2.0 interface.

- ## Higher operational visibility

The platform provides real-time monitoring and a unified view of all data flows, reducing troubleshooting time by up to 60-70% and ensuring high availability across hybrid environments.

- ## European digital sovereignty

As a European-native platform, Frends ensures full GDPR and NIS2 compliance, offering flexible deployment options (Cloud, Hybrid, or On-Premises) that keep sensitive data within secure regional borders.

By aligning technical execution with organizational change management, this strategy ensures that the transition to Frends delivers immediate ROI, often within 3-6 months, while building a scalable foundation for long-term digital growth.

2. Technical implementation

The technical dimension of migration encompasses architecture design, development, testing and deployment activities. A structured, checklist-driven approach ensures quality, reduces implementation risk, and provides clear accountability throughout the migration process.

2.1 Architecture assessment and design

Before migration commences, conduct a thorough assessment of the current integration landscape and design the target platform architecture.

Current state documentation

- Document all existing integrations with unique identifiers and business owners
- Map data flows, including source systems, target systems, and intermediate transformations
- Capture transformation logic, business rules, and data mapping specifications
- Document error handling mechanisms, retry policies, and alerting configurations
- Record scheduling configurations, trigger mechanisms, and dependencies
- Identify authentication methods and credential management approaches
- Assess data volumes, transaction rates, and performance SLAs

Target architecture design

Design integration workflows

leveraging the platform's native capabilities (e.g., BPMN 2.0 if supported)

Define deployment strategy

leveraging the platform's native capabilities (e.g., BPMN 2.0 if supported)

Identify and catalog

required pre-built connectors from the platform library

Determine custom connector

or component development requirements (if pre-built options are unavailable)

Design API trigger strategies

using platform's REST, SOAP or event-driven capabilities

Plan runtime agent

/node distribution and workload management strategy

Define connection

string and credential management approach using platform's secure storage

Establish orchestration

patterns and reusable component architecture

2.2 Development standards and practices

Establish consistent development standards to ensure maintainability, scalability and quality across all migrated integrations.

Standards and Conventions

- Define integration and component naming conventions (e.g., [Domain]_[System]_[Function]_[Version])
- Establish folder structure and organizational taxonomy within the platform
- Create reusable sub-components for common patterns (error handling, logging, notifications)
- Define parameter naming standards for inputs, outputs and variables
- Document code commenting and annotation requirements
- Establish error message formatting and logging standards
- Define versioning strategy and deprecation procedures

Version Control and DevOps

- Integrate platform with version control systems (Git, Azure DevOps, etc.)
- Establish branching strategy (e.g., GitFlow: dev, test, main branches)
- Define code review and approval workflows
- Implement automated deployment pipelines using platform APIs or CLI tools
- Configure environment-specific parameters and secrets management
- Establish rollback procedures for failed deployments
- Set up automated backup strategies for integration definitions

Reusability and Efficiency

- Build a library of reusable components for common operations
- Create templates for standard integration patterns
- Document design patterns (e.g., batch processing, event-driven, scheduled sync)
- Establish shared utility functions (date formatting, data validation, encryption)

2.3 Testing strategy

A comprehensive, multi-level testing strategy is essential for migration success. Each testing phase must have clear entry/exit criteria and documented outcomes.

Testing Level	Description
Unit testing	Validate individual process components and transformation logic in isolation
Integration testing	Verify end-to-end data flows and system connectivity
Regression testing	Ensure migrated integrations produce identical outputs to legacy systems
Performance testing	Validate throughput, latency and resource utilization under expected loads
User Acceptance testing	Business validation of integration outcomes and data accuracy

2.4 Verification and validation

Implement rigorous verification processes to ensure data integrity and functional equivalence between legacy and new platform implementations.

Parallel running (recommended)

- Execute both legacy and new platform integrations simultaneously for a defined period
- Capture outputs from both systems for comparison
- Establish automated reconciliation scripts to compare datasets
- Define acceptable variance thresholds for data differences
- Document root cause analysis for any discrepancies
- Obtain stakeholder approval before decommissioning legacy integration

Alternative: Documentation-based verification

- Ensure legacy integration specifications are comprehensively documented
- Capture baseline outputs and expected results from legacy systems
- Execute new platform integrations and compare against documented baselines
- Validate business logic implementation matches specifications
- Obtain business owner sign-off on migrated integration behavior

Data reconciliation

- Implement automated data comparison scripts (record counts, checksums, field-level validation)
- Define reconciliation frequency (real-time, daily, weekly)
- Establish exception handling for reconciliation failures
- Create dashboards for reconciliation status visibility
- Document and resolve all identified data integrity issues before cutover

2.5 Environment strategy

Establish a robust environment management strategy to support controlled development, testing, and production deployment.

Environment setup

- Provision separate platform environments: Development, Testing, Staging, Production
- Deploy dedicated runtime resources per environment (or shared with clear separation)
- Configure environment management features for deployment targeting
- Establish network connectivity to all integrated systems per environment
- Set up environment-specific parameter stores and secrets management

Connected systems coordination

- Identify environment availability for each connected system (Dev, Test, Prod)
- Document technical contacts and escalation paths per system
- Confirm test data availability and data masking requirements
- Coordinate firewall rules and network access approvals
- Validate SLAs and maintenance windows for connected systems

Promotion and deployment procedures

- Define environment promotion workflow (Dev → Test → Staging → Prod)
- Establish approval gates and sign-off requirements per environment
- Implement automated deployment scripts using platform APIs or deployment tools
- Configure environment-specific variables (connection strings, URLs, credentials)
- Document rollback procedures for failed promotions
- Set up post-deployment smoke tests and health checks

Configuration management

- Centralize configuration using platform's secure parameter/configuration store
- Use environment-specific parameters for connection strings and endpoints
- Implement secret management (avoid hardcoded credentials)
- Document configuration change control procedures
- Establish configuration backup and recovery processes

2.6 Technical readiness checklist

Before proceeding to migration waves, confirm the following technical readiness criteria:

- All platform environments are provisioned and operational
- Development standards and naming conventions are documented and communicated
- Version control integration is configured and tested
- Testing strategy and acceptance criteria are defined
- Parallel running or verification approach is established
- Technical team has completed platform training
- Monitoring, alerting, and incident response procedures are in place

3. People and organizational change

Successful platform migration extends beyond technical implementation. Organizational readiness, skill development and change management are critical success factors.

3.1 Onboarding and training program

Develop a structured training program tailored to different roles within the organization. Training should address both technical and operational aspects of the Frends platform.

Role	Training focus
Developers	Process design, connectors, API development, debugging
Operations	Monitoring, alerting, incident response, scheduling
Architects	Platform capabilities, design patterns, governance
Business Analysts	Process overview, reporting, business requirements

3.2 Knowledge sharing and documentation

Establish comprehensive documentation practices to preserve institutional knowledge and support ongoing operations:

1. Technical documentation

Process specifications, data mappings, configuration guides

2. Operational runbooks

Standard operating procedures, troubleshooting guides, escalation paths

3. Architecture documentation

System diagrams, data flow documentation, dependency maps

4. Knowledge base

FAQ, lessons learned, best practices repository

3.3 Change management

1
Apply structured change management principles to facilitate adoption.

2
Communicate early and frequently about the migration timeline, impacts and benefits.

3
Address resistance through stakeholder engagement and demonstrate quick wins to build momentum.

4
Identify and empower change champions within affected teams.

4. Process and business analysis

Understanding the current state and defining the target state requires thorough business analysis. This ensures migrations deliver business value and support strategic objectives.

4.1 Current state assessment

Document each integration comprehensively, including its business purpose, data sources and targets, transformation requirements, scheduling patterns, error handling behavior and service level agreements.

Categorize integrations by criticality, complexity and migration priority.

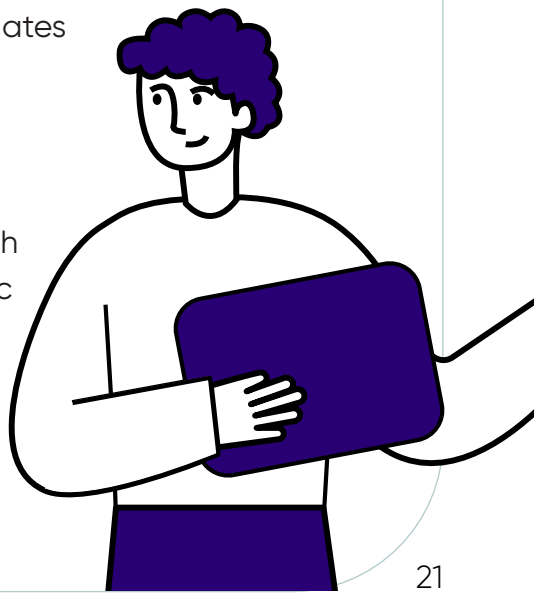
4.2 Business purpose and value analysis

For each integration, validate the ongoing business need.

Note that some legacy integrations may be candidates for retirement rather than migration.

Identify opportunities to consolidate, simplify or enhance integrations during the migration process.

Align migration priorities with business value and strategic importance.



4.3 Critical integration identification

Identify integrations that are mission-critical to business operations. These require enhanced attention, additional testing and carefully planned cutover strategies. Critical integrations typically include:

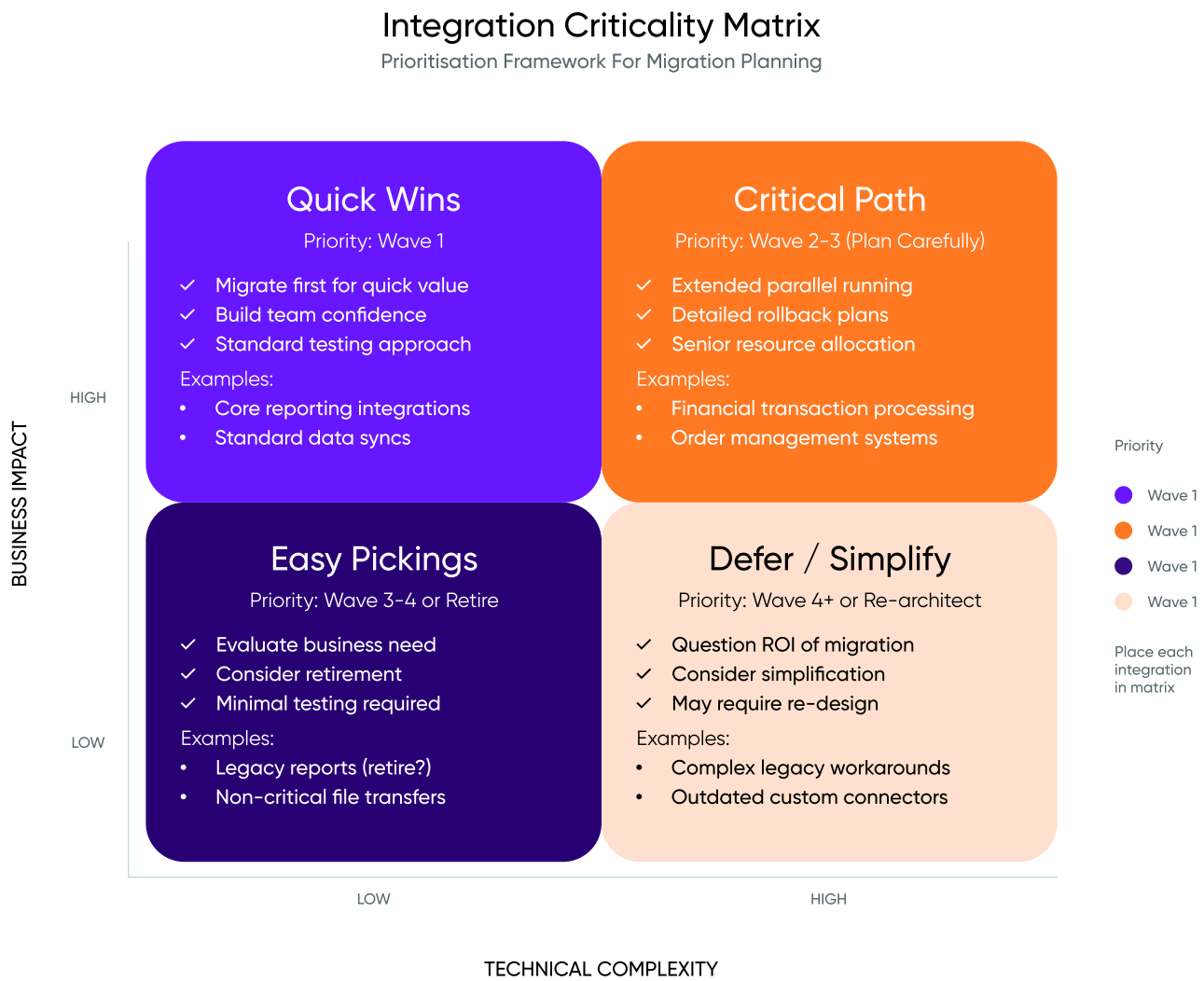
- Financial transaction processing and reporting
- Customer-facing services and order processing
- Regulatory compliance and reporting integrations
- Core ERP and CRM synchronization
- Supply chain and logistics integrations

4.4 Strategic migration planning

Develop a migration roadmap that sequences integrations based on criticality, complexity, dependencies and business timing constraints.

Consider seasonal business cycles and avoid migration activities during peak periods. Plan for contingencies and define clear rollback procedures.

Assign integrations into strategic categories to allow for better migration planning. The following is an example of a categorization matrix for integrations:



The four categories identified by the diagram will help plan the migration process by identifying their importance for business versus technical implementation complexity. This ensures faster time to value and overall migration process confidence.

Quick wins

technically simple integrations that have a high business impact. Such integrations are best candidates to be implemented with higher priority.

Critical path integrations

complex integrations with high business impact, that need careful migration planning and execution. Best executed after quick wins, after making sure that technical resources are well aligned on development practices, domain knowledge and overall execution of the migrations.

Easy pickings

low impact simple integrations that can be often retired or easily transferred.

Defer/simplify

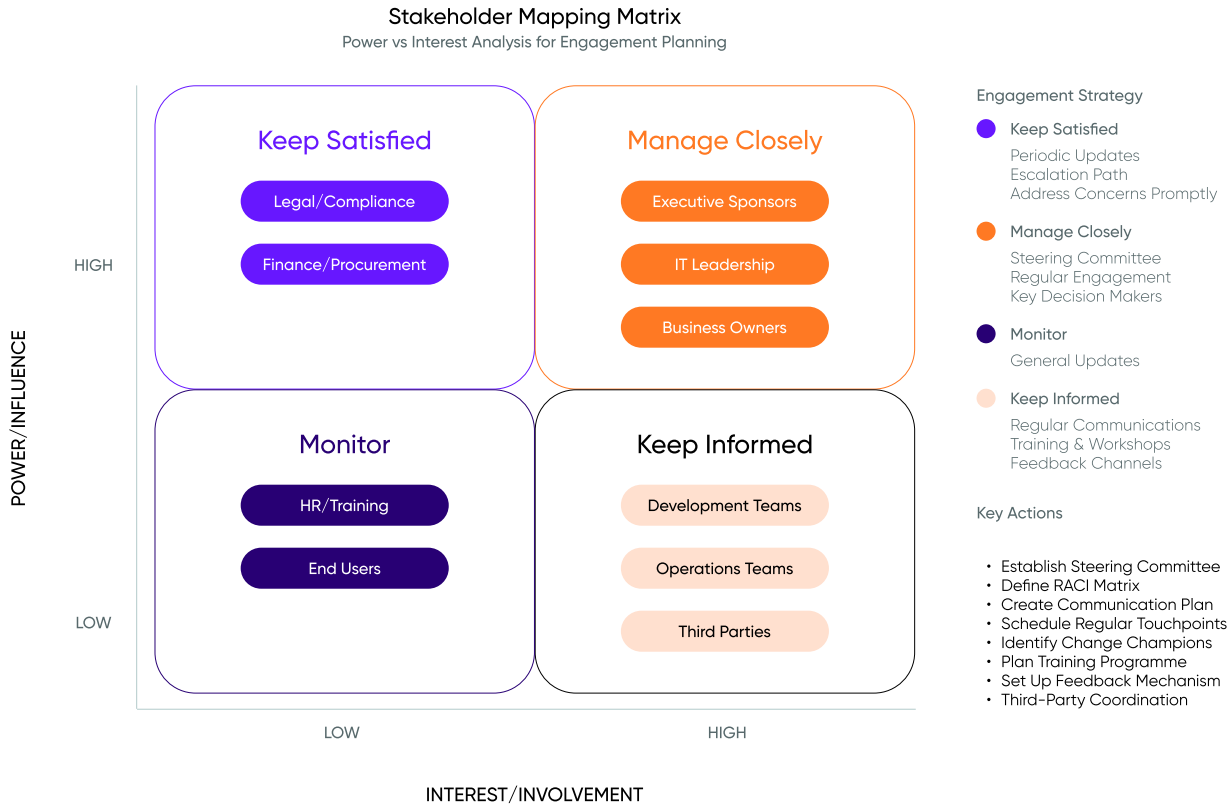
integrations that are technically complex but have little or no business impact that can be either retired, redesigned or in some cases, simplified.

5. Stakeholder management

Effective stakeholder management ensures alignment, commitment and support throughout the migration programme.

5.1 Key stakeholder identification

Identify and engage all stakeholders who have an interest in or are affected by the migration. Map their influence, interest and potential concerns.



Stakeholder Group	Interest Area	Engagement Approach
Executive sponsors	ROI, strategic alignment, risk	Steering committee, executive briefings
IT leadership	Technical feasibility, resource needs	Regular status reviews, escalation path
Business owners	Service continuity, timeline	Business impact assessments, UAT involvement
Development teams	Technical approach, training	Workshops, hands-on training, collaboration
Operations teams	Monitoring, support processes	Runbook development, knowledge transfer
Third parties	Interface changes, testing	Coordination meetings, joint testing

5.2 Applications and systems involved

- Document interface specifications, data contracts and technical owners.
- Create a comprehensive inventory of all applications and systems that interact with the integration platform.
- Coordinate with application teams regarding any changes required on their side.

5.3 Third-party coordination

- Communicate migration plans and timelines. Coordinate testing activities and agree on cutover procedures.
- Identify all external parties involved in integrations, including vendors, partners and service providers.
- Ensure contractual obligations and SLAs are maintained throughout the transition.

5.4 Roles and responsibilities

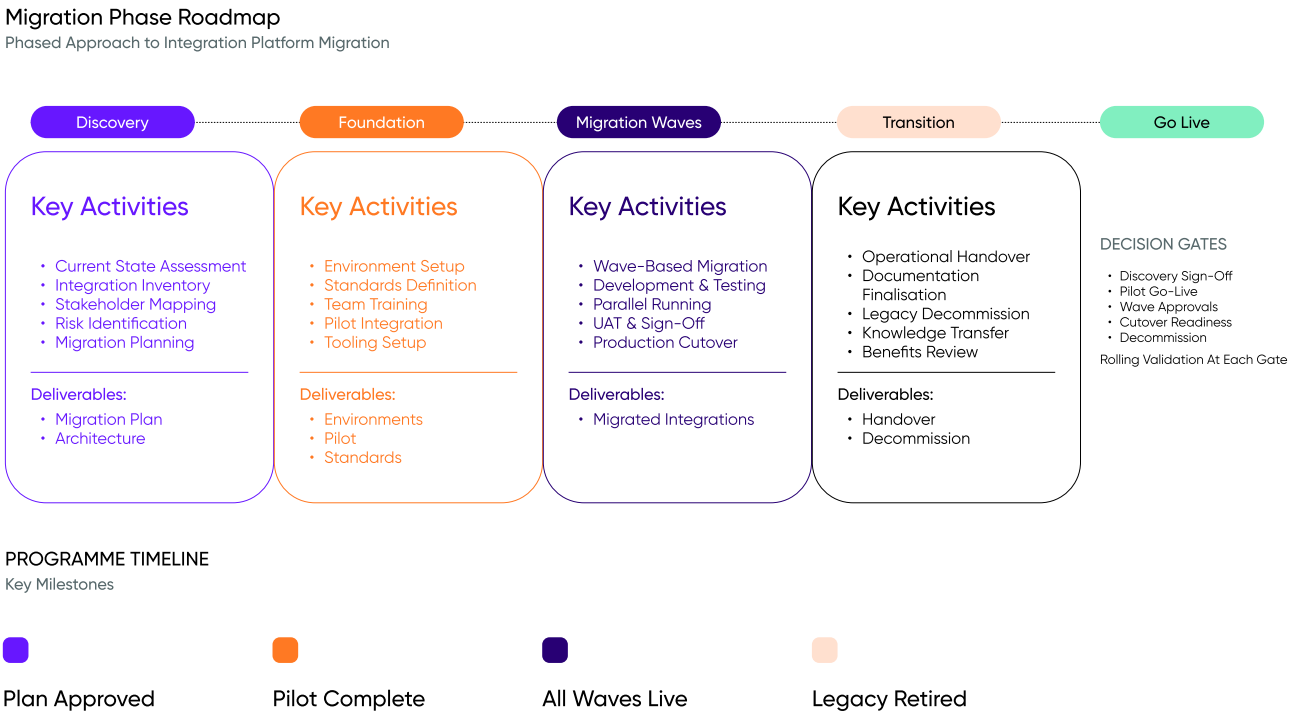
- Ensure accountability for all migration activities, decisions and deliverables.
- Define clear roles and responsibilities using a RACI matrix.
- Common roles include Programme Manager, Technical Lead, Business Analyst, Developer, Tester, Operations Lead and Change Manager.

6. Migration planning and execution

A well-structured plan with clear milestones, deliverables and governance ensures controlled execution and stakeholder confidence.

6.1 Project approach and strategy

Adopt a phased migration approach that balances risk mitigation with timely delivery. The recommended strategy employs iterative waves, migrating groups of related integrations together while maintaining operational stability.



Phase	Activities	Key Deliverables
Discovery	Assessment, planning, design	Migration plan, architecture
Foundation	Environment setup, standards, pilot	Environments, standards, pilot integrations
Migration waves	Iterative development and deployment	Migrated integrations per wave
Transition	Handover, documentation, decommissioning	Operational handover, legacy retirement

6.2 Key dates and milestones

Establish key dates aligned with business constraints and dependencies. Consider the following milestone categories:

- 1. Project initiation and governance establishment
- 2. Environment readiness and technical foundation completion
- 3. Pilot integration go-live and lessons learned
- 4. Wave completion milestones with business sign-off
- 5. Parallel running completion and production cutover
- 6. Legacy platform decommissioning
- 7. Programme closure and benefits realization review

6.3 Migration strategy selection

Selecting the appropriate migration strategy is one of the most consequential decisions in the programme. The chosen approach affects timeline, risk profile, resource requirements and business disruption. Organizations must evaluate their specific context to determine the optimal strategy.

Three primary migration strategies exist, each with distinct characteristics and trade-offs. The following analysis provides guidance for strategy selection.

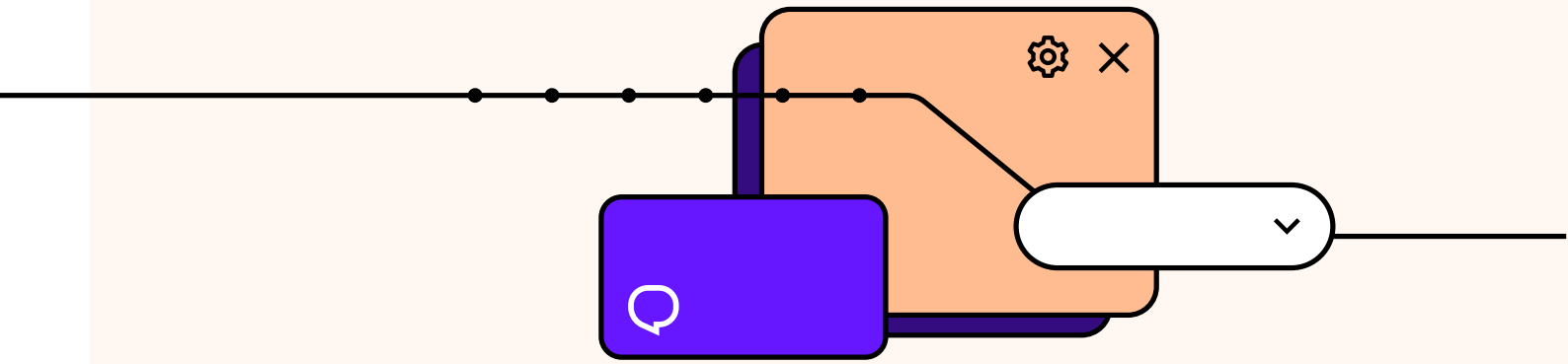
6.3.1 Big bang migration

All integrations are migrated and switched over in a single coordinated event. The legacy platform is decommissioned immediately following successful cutover.

Advantages	Disadvantages
• Single cutover event simplifies coordination	• High risk concentration at cutover
• Shorter overall programme timeline	• Complex rollback if issues arise
• Clean break from legacy platform	• Requires extensive preparation and testing
• Lower total cost if execution is successful	• Significant business disruption potential
• No complex inter-platform interfaces required	• All-or-nothing pressure on teams

Best suited for:

Small integration portfolios (fewer than 20 integrations), organizations with strong testing capability and risk tolerance, low-complexity environments with minimal interdependencies.



Nurminen Logistics ▶▶▶

Nurminen Logistics: Complete BizTalk-to-Frends migration

When Petri Luurila became CIO at Nurminen Logistics, a 140-year-old Finnish freight forwarding company, he inherited what he calls "integration spaghetti."

The company was running three different integration platforms simultaneously, including Microsoft BizTalk.

With no in-house integration resources and mounting maintenance costs, they needed a solution fast.

The migration

Nurminen committed to moving 100% of integrations to Frends and consolidating three platforms into one.

"Switching from BizTalk to Frends was a game-changer," Luurila says. "Frends fits perfectly into our Microsoft setup and the transition was very smooth. The low-code approach sped up our projects, cut costs, and gave us better control."

The key to success was Frends' .NET-native architecture, which allowed the team to reuse existing BizTalk assets (C# code, transformations and orchestration logic), without starting from scratch. Throughout the migration, Nurminen maintained continuous operations with zero disruption.

The outcome

Nurminen achieved complete integration transformation:

- 100% of integrations moved with complete platform replacement
- Three platforms consolidated into one
- 43% improvement in operational efficiency
- Enhanced process transparency, significantly speeding up incident resolution
- Created a digital foundation for AI, APIs, and hyperautomation

"Frends is the reliable core of all data—our digital integration hub," Luurila explains. **"It's helping us develop ecosystem thinking in the traditional logistics field, where sharing data benefits all parties."**

6.3.2 Phased migration by application or domain

Integrations are grouped by business application or functional domain and migrated in successive waves. For example, all CRM-related integrations are migrated first, followed by ERP integrations, then financial systems.

Advantages	Disadvantages
• Manageable scope per wave • Domain expertise can be focused • Easier testing within business context • Clear ownership and accountability per phase • Reduced risk through incremental delivery • Lessons learned applied to subsequent waves	• Temporary dual-platform operation required • Interface complexity between migrated and non-migrated systems • Extended overall timeline • Higher infrastructure costs during transition • Requires careful dependency management

Best suited for:

Medium to large integration portfolios, environments with complex interdependencies, organizations preferring controlled risk exposure, situations where business domains have clear boundaries.

Fazer: Consolidating SAP PI and multiple platforms into Frends

When Fazer, one of the Nordic's largest food companies with €1.1 billion in revenue and operations across eight countries, looked at their integration landscape in 2018, they knew right away they needed to act.

Fazer's integration challenges stemmed from their organizational structure. Operating across diverse business units, each with unique requirements in bakery, confectionery and plant-based foods, the company had accumulated various integration systems over time.

"When something broke, incidents took forever to locate and resolve,"

one stakeholder noted. With SAP announcing the discontinuation of PI/PO maintenance by 2027, Fazer needed to act.

SAP PI handled ERP integrations, other tools managed EDI, and additional systems connected their finance operations.

The phased approach

Fazer committed to consolidating all integrations under Frends, but they took a methodical, domain-by-domain approach:

Wave 1: EDI integrations

Fazer started with their Electronic Data Interchange integrations, the backbone of their customer transactions. Working across different business units, they reimplemented all order, delivery and invoicing EDI integrations with their diverse retail customers. This wave was itself broken down by business unit, with each unit having its own schedule.

"The EDI project was rolled out incrementally with own schedules for each business unit,"

ensuring that rollouts didn't disrupt normal operations. By the end of this wave, Fazer had integrated over 15 different EDI partners and was processing thousands of daily messages through Frends.

Wave 2: SAP PI migration

With EDI success under their belt, Fazer tackled the more complex SAP integrations. They migrated integrations from SAP PI to Frends, consolidating what had been a separate platform with its own supplier. This centralized monitoring capabilities both within the SAP ecosystem and beyond.

Wave 3: Finance system overhaul:

The final major wave involved overhauling finance integrations alongside enabling the adoption of a new finance system. This demonstrated how the phased approach allowed Fazer to not just migrate, but modernize, using each wave as an opportunity to rethink and improve their processes.

The outcome

By taking a phased, domain-driven approach, Fazer achieved:

- Complete platform consolidation: from three integration platforms to one
- Enhanced process transparency: significantly speeding up incident resolution
- Faster supplier onboarding: from weeks to days
- Eliminated vendor lock-in: from SAP PI
- Cost savings: through supplier reduction and operational efficiency
- Cloud readiness: while maintaining hybrid on-premise/cloud architecture

Perhaps most importantly, the phased approach allowed Fazer to apply lessons learned from each wave to subsequent migrations. The reusable components created during the EDI wave, for example, became templates for faster onboarding of new partners in later phases.

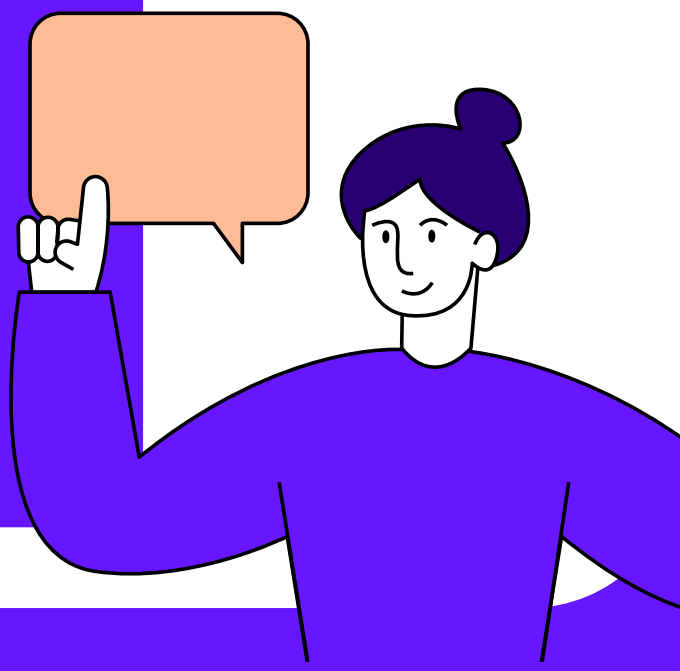
6.3.3 Phased migration by criticality and complexity

Integrations are sequenced based on their risk profile rather than business domain. Migration begins with low-risk, low-complexity integrations and progressively moves to more critical systems as team experience and confidence grow.

Advantages	Disadvantages
• Early wins build team confidence • Team learns platform on simpler integrations • Risk is graduated over time • Approach validated before critical migrations • Issues discovered early with lower impact • Training effectiveness can be assessed	• Critical integrations delayed until later phases • May not align with business priorities • Potential for difficult integrations to be perpetually deferred • Cross-domain complexity in each wave

Best suited for:

Risk-averse organizations, teams with limited Frends experience, complex legacy environments with technical debt, situations where building internal capability is a priority.



PRIMA PET

Prima Pet Premium: From pilot to production

When Prima Pet Premium, a Finnish manufacturer of premium pet food products, decided to modernize their integration landscape, they faced a common challenge:

A small IT team with no prior iPaaS experience, managing critical business operations that couldn't afford disruption.

Prima Pet's IT Development Manager knew that jumping straight into mission-critical ERP integrations would be risky.

The team had excellent knowledge of their business systems but zero experience with modern integration platforms.

"We didn't just want a partner to build everything for us, we wanted to stay in control," they explained. "But we also knew we needed to learn the platform properly before tackling our most complex integrations."

The company started with a carefully selected pilot integration: a non-critical but valuable use case that would allow the team to learn Frends without putting core business operations at risk. They chose a straightforward data synchronization between two systems that, while important, wouldn't halt operations if something went wrong during the learning phase.

Learning by doing: The pilot phase

Working alongside Frends' Professional Services team, the Prima Pet developers didn't just watch, they built. The pilot was structured as a hands-on learning experience with Frends experts providing guidance rather than simply delivering a finished solution. This approach served multiple purposes:

- **Technology validation:** Confirmed Frends could handle their technical requirements
- **Team training:** Developers gained practical experience in a low-stakes environment
- **Confidence building:** Early success demonstrated they could master the platform
- **Pattern establishment:** Created reusable components and standards for future integrations

The pilot completed successfully, and crucially, the team now understood both Frends' capabilities and their own capacity to work with it.

"The pilot gave us the confidence that this was manageable," the IT Development Manager noted. **"We proved to ourselves we could do this."**

Graduated complexity: Moving to business-critical systems

With the pilot validated and the team trained, Prima Pet moved to Phase 2: Core business integrations. But even here, they didn't attempt everything at once. Instead, they sequenced their ERP and warehouse management integrations based on complexity and business impact:

Wave 1: Foundation integrations (lower complexity)

Prima Pet began with essential but relatively straightforward integrations: master data synchronization and basic inventory updates. These integrations were critical enough to matter but structured enough that the team could handle them with their newly acquired skills.

Wave 2: Transactional integrations (medium complexity)

With foundational integrations running smoothly, the team tackled order processing and customer data flows. These required more sophisticated error handling and business logic, but the team was now experienced enough to manage the complexity. Importantly, they applied lessons learned from earlier waves, reusing components, refining their testing approach and improving their documentation practices.

Wave 3: Advanced integration and CRM expansion (higher complexity)

By the time Prima Pet approached their most complex integrations with real-time inventory management and CRM system connections, the team was operating with full confidence.

They had built a library of reusable components, established clear development standards and proven their ability to support integrations independently.

The outcome: Capability as the foundation

Prima Pet's phased approach delivered more than just working integrations, it built internal capability that became a strategic asset:

- **Self-sufficiency:** The IT team can now design, build and maintain integrations independently
- **Agility:** New integration requirements are handled quickly using established patterns
- **Clarity and transparency:** Frends' visual BPMN interface means everyone understands how systems connect
- **Foundation for growth:** The platform now supports their expanding business, including plans for broader CRM integration

Perhaps most importantly, Prima Pet avoided the common pitfall of over-reliance on external consultants.

"We wanted to stay in control," their IT Development Manager emphasized. **"Frends has helped us do that with confidence, speed, and transparency."**

6.4 Migrating data

Data is the lifeblood of any integration. Migrating to a new platform is not just about moving logic; it is about ensuring that the data flowing through the new system remains accurate, secure and actionable.

6.4.1 How to get data ready for a new platform

Preparation is the most critical phase of data migration. Moving "dirty" or poorly structured data into a modern platform will propagate existing issues. So first, make sure the data is ready.

- **Data auditing & profiling:** Conduct a thorough audit of existing data sets. Identify inconsistencies, duplicates and obsolete records. Use this migration as an opportunity to purge data that no longer serves a business purpose.
- **Schema mapping & transformation:** Document the "as-is" data structures in the legacy system and map them to the "to-be" structures. Identify where data needs to be transformed, enriched or normalized to fit the new platform's requirements.
- **Establish a "golden record" strategy:** Define which system holds the master version of specific data entities (e.g., Customer, Product, Inventory). Ensure the new integration logic respects these authoritative sources.
- **Security & compliance review:** Identify sensitive data (PII, financial records) and ensure that the new platform's encryption, masking and access control settings are configured to meet GDPR, NIS2 or other regulatory standards before the first byte is moved.
- **Baseline creation:** Capture "snapshots" of data outputs from the legacy system. These will serve as the benchmark for regression testing to ensure the new platform produces identical results.

6.4.2 What to avoid

Many migration projects fail or exceed budgets due to common, avoidable pitfalls:

- **"Lift and shift" mentality:** Avoid simply copying legacy logic and data structures into the new platform. Legacy systems often contain "workarounds" for technical limitations that no longer exist in modern iPaaS. Use the migration to optimize, not just replicate.
- **Ignoring data latency:** Do not assume the new platform will handle data timing the same way. If the legacy system relied on specific batch windows or sequential processing, ensure the new architecture accounts for these dependencies to avoid race conditions.
- **Hardcoding configurations:** Never hardcode connection strings, URLs or credentials within the data transformation logic. Use environment-specific parameters and secure vaults to ensure data can move between Dev, Test and Prod without manual code changes.
- **Underestimating volume:** Avoid testing with only "happy path" or small data samples. Migrations often break when hit with full production volumes or unexpected "edge case" data formats that weren't present in the test set.

- **Lack of rollback planning:** Never initiate a data cutover without a tested rollback plan. If data corruption occurs during the migration, you must be able to revert to the last known "good" state immediately.

6.4.3 After the migration

The work does not end at the "Go-Live" event. Post-migration activities ensure long-term stability and value realization.

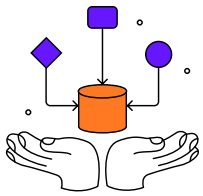
- **Hyper-care monitoring:** Implement enhanced monitoring for the first 30 days post-migration. Track error rates, latency and data throughput closely to catch subtle issues that may not have appeared during UAT.
- **Data reconciliation loops:** Run automated scripts to compare record counts and checksums between source and target systems for a set period. This confirms that no data was lost or corrupted during the "flight" through the new platform.
- • **Decommissioning legacy access:** Once the new platform is validated, revoke write-access to the legacy integration points to prevent "shadow" data updates that could lead to synchronization conflicts.
- **Performance optimization:** Analyze the execution logs in the new platform. Identify bottlenecks in data transformation or slow-responding API endpoints and optimize them using the platform's native performance-tuning features.
- **Knowledge transfer & documentation:** Update the operational runbooks with the new data flow diagrams and troubleshooting steps. Ensure the support team understands how to monitor data health in the new environment.

7. Resourcing and implementation partner

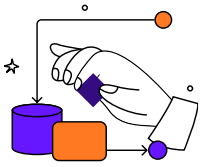
Adequate resourcing and, where appropriate, partner engagement are essential for successful delivery.

7.1 Resource requirements

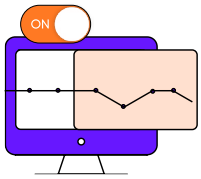
Assess resource requirements across all workstreams. Consider the following resource categories:



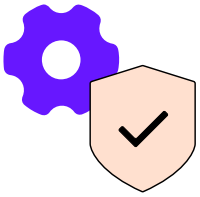
Programme management and governance



Solution architecture and design



Frends development and configuration



Testing and quality assurance

Change management and training

Operations and support transition

7.2 Implementation partner selection

Consider engaging an implementation partner with proven Frends expertise, particularly if internal capabilities are limited. Evaluate potential partners against the following criteria:

Criteria	Evaluation Factors
Frends expertise	Certified consultants, successful implementations, partner status
Industry experience	Relevant sector knowledge, understanding of business context
Migration methodology	Proven approach, tools and accelerators, risk management
Knowledge transfer	Training capability, documentation standards, handover process
Cultural fit	Collaboration style, communication, flexibility
Commercial terms	Pricing model, contractual flexibility, SLAs

7.3 Engagement model

Define the engagement model with the implementation partner. Options include full outsourcing for complete delivery responsibility, co-delivery for joint teams with knowledge transfer and advisory for guidance and quality assurance with internal delivery. The co-delivery model is often preferred as it builds internal capability while leveraging external expertise.

8. Risk management

Proactive risk management reduces the likelihood and impact of issues during migration. Establish a risk register and review it regularly throughout the programme.

8.1 Key risks and mitigation strategies

Risk Category	Risk Description	Mitigation Strategy
Technical	Data loss or corruption during migration	Comprehensive backup strategy, parallel running, automated data reconciliation, rollback procedures
Technical	Performance degradation in new platform	Performance testing, capacity planning, optimization sprints, monitoring implementation
Schedule	Timeline slippage affecting business operations	Realistic planning, buffer allocation, phased approach, clear escalation paths
Resource	Insufficient skills or availability	Early training, partner engagement, cross-training, contingency resources
Business	Service disruption to critical operations	Prioritized testing, extended parallel running, detailed cutover plans, business continuity provisions
Change	User adoption resistance	Change management programme, communication, training, champion network
Third Party	External dependencies causing delays	Early engagement, contractual commitments, alternative solutions, dependency tracking
Knowledge	Loss of institutional knowledge	Comprehensive documentation, knowledge transfer sessions, retention planning

8.2 Risk governance

Establish risk governance processes, including regular risk review meetings, defined risk ownership and escalation thresholds, risk appetite definition aligned with business tolerance, contingency budget and time allocation and incident response procedures for materialized risks.

8.3 Contingency planning

Develop contingency plans for high-impact scenarios.

Define rollback procedures for failed migrations.

Establish fallback communication channels.

Maintain legacy platform capability until migration is validated.

Document lessons learned and adjust plans iteratively.

9. Migration checklist

1. Technical assessment

- Have we documented all existing integrations, APIs and workflows?
- Have we identified legacy systems that need to be replaced or connected?
- Do we have access to current system architecture diagrams?
- Are there any known integration bottlenecks or performance issues in the current platform?
- Have we conducted an assessment of data formats, protocols, and endpoints?

2. Architecture & platform fit

- Have we reviewed Frends' hybrid capabilities for cloud/on-prem environments?
- Are Frends Agents installable in required environments (cloud, on-premise, hybrid)?
- Is the organization aligned with BPMN 2.0 modeling, or does it require onboarding?
- Do we have existing monitoring and observability tools to integrate with Frends?

9.1.1 People & training

- Do we have internal integration developers or a partner team in place?
- Have we planned onboarding and training on Frends' low-code and visual tools?
- Are there change champions identified in IT and business units?
- Is there a communications plan to inform and involve stakeholders?

9.1.2 Process and business analysis

- Have we documented critical business processes tied to integrations?
- Have we prioritized which integrations are business-critical for go-live?
- Have we mapped desired future state workflows (with optimization goals)?
- Is there a clear business case or ROI for the migration?

9.1.3 Migration planning

- Have we selected a migration strategy (Big Bang, phased by domain, complexity)?
- Are we aware of migration tooling or accelerators offered by Frends?
- Have key milestones and go-live criteria been defined?
- Have rollback and contingency plans been developed?

9.1.4 Governance & risk

- Has a risk register been created with mitigation plans?
- Are there defined roles and responsibilities (project owner, tech lead, QA)?
- Are compliance, security and audit requirements documented?
- Have we set up governance for ongoing integration lifecycle post-migration?

9.1.5 Vendor & stakeholder coordination

- Are third-party vendors and SaaS partners informed and involved?
- Have we confirmed data access and SLAs for connected systems?
- Do we have a dedicated Frends implementation partner or success manager?

10. Future-proofing your integration strategy after the migration

You completed the migration? Congratulations, but you're not done. The real value emerges in what you do next.

Organizations that treat migration as a finish line watch their shiny new platform devolve into the same chaos they just escaped. Those that treat it as a starting point build strategic capabilities that compound over time.

10.1 Get your governance right (without killing agility)

Here's the uncomfortable truth: without governance, your integration platform becomes a free-for-all within 18 months. With too much governance, it becomes a bottleneck that everyone routes around. The sweet spot is standards that enable speed, not bureaucracy that kills it.

Build a living integration handbook

Create one source of truth that covers:

- **Development standards:** Naming conventions, folder structures, parameter management
- **Architecture patterns:** Approved integration templates and reusable components
- **Security and compliance:** Data handling requirements, encryption standards, audit logging
- **Operational procedures:** Deployment workflows, incident response, escalation paths

Make it accessible, keep it updated and treat it as collaborative, instead of handed down from on high. When your developers can reference consistent standards instead of reinventing solutions, quality goes up and technical debt goes down.

Keep decision-making lightweight

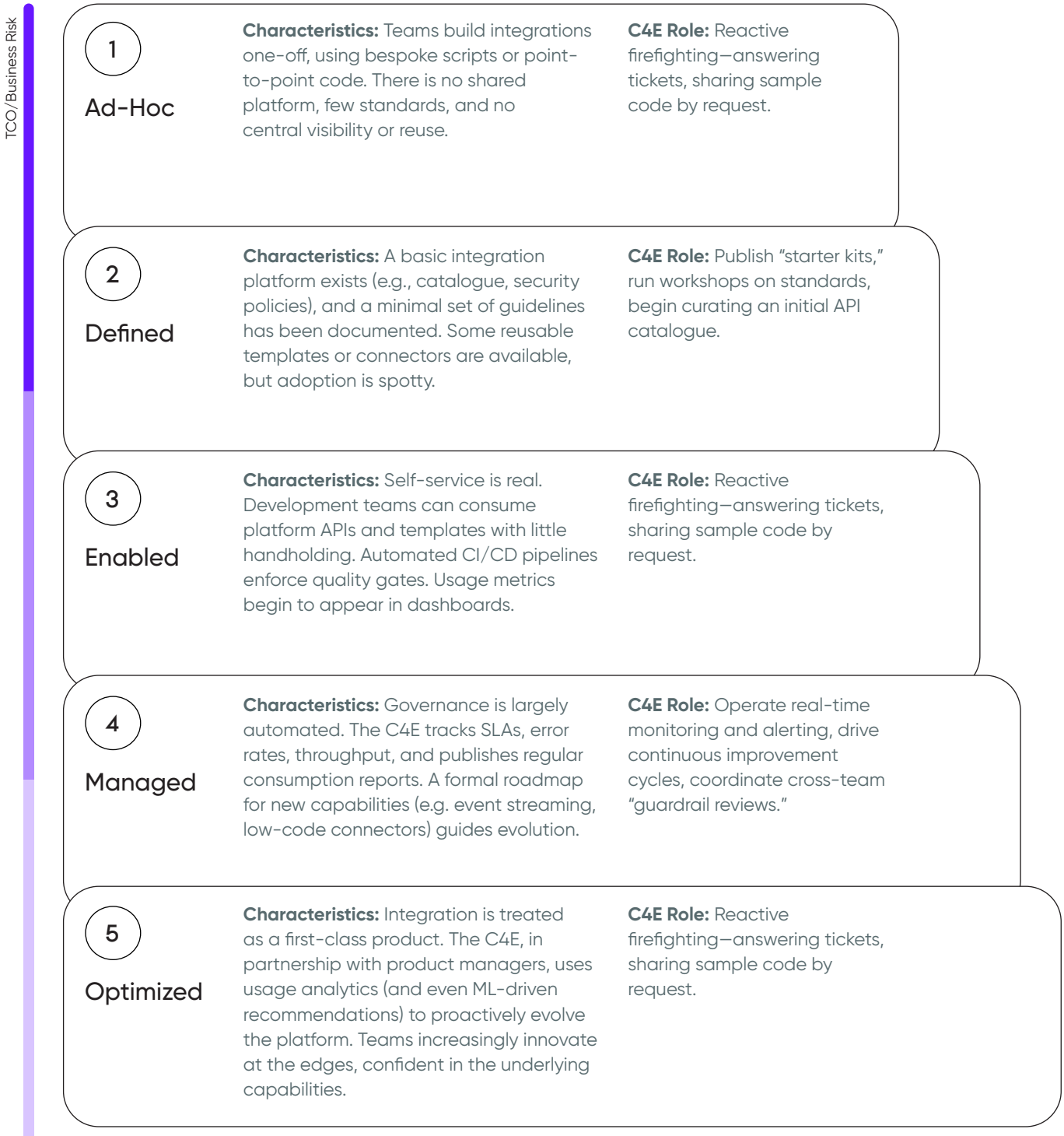
You need structure, not committees. Consider:

- **Integration steering committee:** Quarterly reviews of strategy and priorities (IT, business, security, compliance)
- **Best practices review group:** Architects and senior developers who evaluate complex designs and capture lessons learned
- **Change advisory board:** Only if compliance demands it. Ootherwise, trust your deployment pipeline

The goal? Enable, don't obstruct. Governance should accelerate delivery, not slow it down.

10.2 Scale through enablement, not centralization

The traditional model, with one central integration team doing all the work, doesn't scale. Demand will always outpace capacity, creating backlogs and bottlenecks. Smart organizations shift from control to enablement through an Integration Center for Enablement (C4E)



What a C4E actually does:

- Manages platform health (environments, security, upgrades)
- Provides training and onboarding that makes teams self-sufficient
- Develops reusable templates and proven patterns
- Facilitates knowledge sharing across teams
- Advocates for strategic adoption of new capabilities (like AI)

What a C4E doesn't do:

- Execute every integration project (that's the old bottleneck model)
- Control everything through approval gates
- Hoard expertise instead of distributing it

This works when you have multiple business units needing integration capabilities and want to scale fast without scaling headcount proportionally. It requires a mindset shift from gatekeepers to force multipliers.

10.3 Invest in continuous learning (or watch your platform stagnate)

Your team's proficiency determines your ROI. Period. Integration isn't a skill you learn once during migration training and never revisit.

Make learning ongoing:

- Quarterly refresher training for developers, architects and operations
- Hands-on workshops when new features drop (AI agents, new connectors, monitoring tools)
- Certifications where available, they validate skills and build confidence

Document everything that matters:

- Operational runbooks for troubleshooting, performance tuning, incident escalation
- Clear integration documentation: purpose, data flows, dependencies, error handling
- Lessons learned repository: capture what worked and what didn't

Build community:

- Quarterly refresher training for developers, architects and operations
- Hands-on workshops when new features drop (AI agents, new connectors, monitoring tools)
- Certifications where available, they validate skills and build confidence

- Quarterly refresher training for developers, architects and operations
- Hands-on workshops when new features drop (AI agents, new connectors, monitoring tools)
- Certifications where available, they validate skills and build confidence

Organizations that view learning as overhead rather than investment never extract full value from their platform.

10.4 Stay aligned with platform evolution (but avoid shiny object syndrome)

Integration platforms evolve fast. Stay aligned with your vendor's roadmap to benefit from innovation without constant custom development.

Engage proactively:

- Participate in beta programs for early access and influence
- Quarterly roadmap reviews with your vendor's account team
- Actively provide feedback as vendors prioritize based on customer demand

Adopt AI strategically:

Modern platforms offer AI-driven capabilities (autonomous agents, intelligent error handling, predictive analytics). Don't rush to implement everything:

- Start with high-volume, low-risk use cases (invoice exception handling, ticket routing)
- Maintain transparency and auditability (critical for regulated industries)
- Pilot, learn, refine, then scale

Resist the temptation:

Not every new feature needs immediate adoption. Evaluate innovations against business priorities, risk tolerance and team readiness. Strategic adoption delivers measurable value. Technology adoption for its own sake delivers invoices.

10.5 Manage your platform like you mean it

Post-migration, the platform transitions to business-as-usual, but that's not permission to neglect it. Proactive management prevents technical debt and ensures long-term performance.

Monitor intelligently:

- Define normal performance baselines (throughput, latency, error rates)
- Configure alerts that distinguish signal from noise (reduce alert fatigue)
- Build dashboards that show integration health and business impact

Optimize continuously:

- Quarterly performance reviews to identify bottlenecks and inefficiencies
- Refactor resource-hungry integrations and retire unused processes
- Forecast capacity needs (agent scaling, database sizing) before issues hit

Control technical debt:

- Schedule regular refactoring sprints (don't let debt accumulate)
- Update deprecated connectors, consolidate redundant integrations
- Clean up old versions and obsolete configurations

Leverage your vendor:

Don't hesitate to involve vendor architects or consultants for complex challenges. It's not an admission of inadequacy. You should see it as smart resource management.

10.6 Build the right culture

Technology without the right culture delivers mediocre results. Integration excellence requires teams that see their work as strategic capability, not technical plumbing.

What works:

- Celebrate wins and share success stories (build momentum)
- Create sandbox environments for safe experimentation
- Foster cross-functional collaboration (integration work spans boundaries)
- Maintain a customer-centric mindset (integrations serve business processes, not technical elegance)

What doesn't:

- Punishing failure (kills innovation)
- Siloed ownership (integration requires collaboration)
- Technology decisions disconnected from business value

10.7 Treat your vendor as a strategic partner

Your relationship with your platform vendor shouldn't end after migration. Organizations that view vendors as transactional software suppliers miss opportunities for collaboration and value creation.

Make it a partnership:

- Regular business reviews (quarterly or bi-annually) to discuss roadmap alignment and optimization opportunities
- Leverage professional services strategically (architectural reviews, health checks, strategic guidance)
- Stay connected to the vendor community (user conferences, webinars, customer networks)

The most successful vendor relationships are characterized by mutual investment and open communication, not transactional purchasing.

11. Conclusion

Migration to Frends represents a significant undertaking that requires careful planning and execution across technical, organizational and process dimensions. Success depends on clear strategy, strong governance, stakeholder alignment and disciplined execution.

With this whitepaper, you get a comprehensive framework for approaching the migration. Organizations should adapt this guidance to their specific context, considering their unique integration landscape, organizational capabilities and business constraints.

Key success factors include executive sponsorship and sustained commitment, realistic planning with appropriate contingencies, comprehensive testing and validation, effective change management and training and clear communication throughout the programme.

By addressing these factors and following the structured approach outlined in this document, organizations achieve a successful migration to Frends while maintaining operational stability and realizing the benefits of a modern integration platform.

frends

Thank you

for more, visit **frends.com**

Frends iPaaS

Where data flows, business grows.