International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

MS-to-MS FTP Integration: Design Patterns for Multi-System Batch Data Exchange

Sivasankararao Kavuri

Data Migration Lead, USA

ABSTRACT: Multi-system, or MS-to-MS, FTP integration describes the broader design problem that arises once an organization moves beyond a single pair of systems exchanging batch files and must instead coordinate file-based data exchange across three, five, or dozens of systems simultaneously. What works cleanly for two systems, a simple point-to-point file drop, becomes progressively harder to govern, secure, and troubleshoot as system count grows, and the design pattern chosen early in an integration program's life has lasting consequences for its long-term maintainability.
This article presents a structured set of design patterns for MS-to-MS FTP integration, covering point-to-point exchange, hub-and-spoke brokering, and publish-subscribe file distribution, along with a decision framework for selecting among them. It addresses the full supporting discipline required to operate any of these patterns reliably at scale: file lifecycle management, scheduling models, naming conventions, error handling, security architecture, monitoring, governance, and performance. The article is illustrated throughout with simple structural diagrams and a dense set of reference tables covering the full integration lifecycle.

The framework's central argument is that pattern selection should follow directly from system count and topology, illustrated concretely in Section 18, where a five-system point-to-point mesh requires ten separate connections while the equivalent hub-and-spoke topology requires only five, a difference that compounds quickly as additional systems join the integration landscape.

KEYWORDS: MS-to-MS FTP Integration, Multi-System Data Exchange, Batch File Transfer, FTP Integration, Enterprise Data Integration, Automated Data Exchange, Secure File Transfer

I. INTRODUCTION

1.1 Why Multi-System Batch Exchange Is a Distinct Design Problem

A two-system FTP integration is, in most respects, a solved problem: establish a connection, agree on a file format, schedule a transfer, and confirm receipt. Extending this pattern to a third, fourth, and fifth system by simply repeating it independently for each new pair is where the design problem actually begins, since the number of independent connections grows far faster than the number of systems involved, illustrated concretely in Section 18.

1.2 Purpose and Scope

This article's purpose is to give integration architects a structured set of design patterns and supporting practices for MS-to-MS FTP integration, applicable across ERP, warehouse, logistics, financial, and other enterprise system landscapes exchanging batch data. The scope covers integration topology patterns, file lifecycle and scheduling design, security, governance, and operational practices. Application-specific data content, such as the specific business meaning of a given file's fields, is referenced only generically, since this article addresses the integration pattern layer rather than any single application domain.

- In scope: point-to-point, hub-and-spoke, and publish-subscribe MS-to-MS design patterns.
- In scope: file lifecycle, scheduling, naming, security, and governance practices supporting any of the three patterns.
- Out of scope: application-specific file content and business data semantics.
- Out of scope: real-time messaging and event-streaming integration, addressed in companion research.

1.3 Relationship to General Integration Governance

The patterns and practices presented in this article assume a program already operating under a general data migration and integration governance discipline covering value mapping, reconciliation cadence, and change control of the kind established in companion research on cross-environment value mapping and logistics master data migration. This article extends that general discipline with the structural considerations unique to multi-system batch file exchange, rather than restating governance principles that apply equally to any integration technology.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

II. FOUNDATIONS OF MS-TO-MS BATCH EXCHANGE

2.1 How This Article Differs From Two-System Integration Guidance

Readers familiar with general FTP integration guidance covering a single source and target system will recognize much of the underlying transport mechanics addressed in this article. What distinguishes MS-to-MS integration specifically is not the transport protocol, which remains FTP or a secure variant throughout, but the coordination problem introduced once a third, fourth, or twentieth system joins the landscape. Every section from this point forward addresses that coordination problem directly: which topology to choose, how to keep conventions consistent across many independently built integrations, and how to govern a shared component, such as a hub, that no single system team fully controls.

2.2 Defining MS-to-MS Integration

MS-to-MS integration refers to any batch file exchange scenario involving more than two participating systems, whether those systems exchange files directly with one another or through an intermediary. The defining characteristic is not the specific protocol used, though this article focuses on FTP and its secure variants, but the multi-party coordination challenge that a two-system integration does not face.

2.3 Core Building Blocks

Building Block	Role
Source system	Produces the outbound file containing data to be shared
Target system	Consumes the inbound file and loads its content
Transport mechanism	Moves the file between source and target, directly or through an intermediary
Control artifact	Signals file completeness and supports reconciliation

Table.1. Core building blocks common to every MS-to-MS FTP integration, regardless of pattern.

Every design pattern examined in this article, point-to-point, hub-and-spoke, and publish-subscribe, is ultimately an arrangement of these same four building blocks. What differs between patterns is not the presence of a source, target, transport, and control artifact, but how many of each participate in a given exchange and how they are connected to one another. Keeping this in mind while reading the pattern-specific sections that follow helps clarify that the patterns are variations on a shared foundation rather than fundamentally different technologies requiring separate skill sets to implement.

2.4 Why FTP Remains a Common Transport Choice

Reason	Explanation
Broad compatibility	Nearly every enterprise system, legacy or modern, supports FTP or a secure variant
Operational simplicity	File-based exchange is easier to inspect, log, and troubleshoot than many alternatives
Natural batch alignment	Many source systems already produce output in batch form, aligning naturally with file transport
Durable audit artifact	The file itself serves as a durable record of what was exchanged and when

Table.2. Common reasons FTP remains a widely used transport choice for MS-to-MS batch integration.

2.5 A Composite Anti-Pattern: The Accidental Hub

A pattern familiar to experienced integration teams illustrates how an unplanned topology emerges even when no one deliberately designed a hub. It typically begins when System A already has a point-to-point connection to System B, and a new requirement emerges to share the same data with System C. Rather than building a proper hub, the fastest

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

path is to have System B simply forward a copy of the file it already receives from System A onward to System C, since System B already possesses the file and the forwarding logic seems trivial to add.

System B has now become a hub in practice, without anyone having designed it as one, without the monitoring, governance, or documented ownership a deliberately designed hub would carry, and without System B's own team necessarily being aware that their system has taken on this additional responsibility. When System B is later decommissioned or upgraded, the forwarding logic is easily overlooked, since it was never documented as a formal integration component, and System C's data feed silently breaks.

The mitigation is not to prohibit convenient forwarding entirely, but to recognize the moment a system begins relaying data on behalf of a third party as the trigger to formalize that relationship explicitly, either as a deliberately designed hub-and-spoke pattern per Section 5 or as a direct point-to-point connection between the actual source and the new consumer, rather than allowing an accidental hub to persist undocumented.

III. DESIGN PATTERN 1: POINT-TO-POINT EXCHANGE

Figure 1 illustrates the simplest MS-to-MS pattern: a direct connection between exactly two systems, with files pushed from one side, pulled from the other, or both.

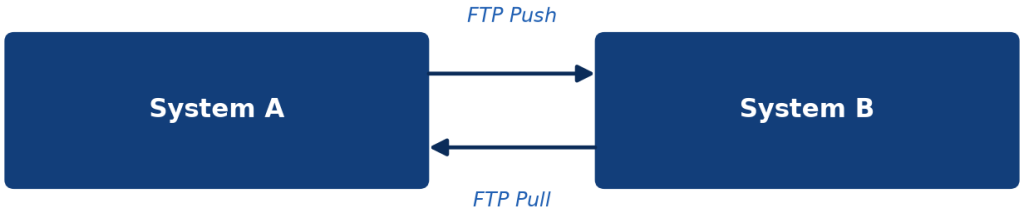

Figure.1. Point-to-point pattern between two systems, showing both push and pull transfer directions.

3.1 Strengths and Limitations

Dimension	Point-to-Point Assessment
Implementation simplicity	High for a small number of system pairs
Connectivity complexity as system count grows	Grows quadratically, per Section 18
Governance visibility	Low once more than a few connections exist independently
Best-fit scenario	Two or three systems with a stable, unlikely-to-grow integration landscape

Table.3. Strengths and limitations of the point-to-point MS-to-MS design pattern.

3.2 When Point-to-Point Remains the Right Long-Term Choice

It is worth stating plainly that point-to-point is not merely a starting pattern every landscape should outgrow. A genuinely stable two-system relationship, such as a dedicated integration between a core ERP system and a single long-standing external partner's system, can remain point-to-point indefinitely without ever needing to migrate to a broker pattern, provided no third system ever needs to participate in that specific exchange. The migration pressure described elsewhere in this article applies specifically once a third participant enters the picture, not as a universal countdown timer attached to every point-to-point connection regardless of how it actually evolves.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

3.3 Implementation Considerations

- Document every point-to-point connection in a shared inventory, even when the pattern otherwise carries low governance overhead.
- Confirm both systems agree on push versus pull responsibility explicitly, rather than each side assuming the other initiates transfer.
- Revisit the pattern choice, per Section 7, the moment a third system needs the same data.

IV. DESIGN PATTERN 2: HUB-AND-SPOKE BROKER

Figure 2 illustrates the hub-and-spoke pattern, routing all file exchange through a central broker rather than direct system-to-system connections.

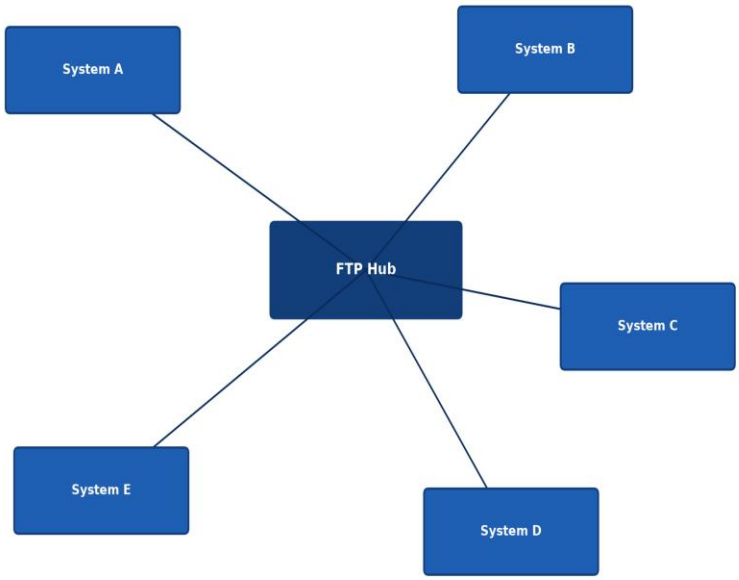

Figure.2. Hub-and-spoke pattern routing file exchange for five systems through a single central FTP hub.

4.1 Strengths and Limitations

Dimension	Hub-and-Spoke Assessment
Implementation simplicity	Moderate; requires initial hub setup investment
Connectivity complexity as system count grows	Grows linearly, per Section 18
Governance visibility	High; all traffic passes through one governed point
Best-fit scenario	Four or more participating systems, or an actively growing landscape

Table.4. Strengths and limitations of the hub-and-spoke MS-to-MS design pattern.

4.2 Hub Governance as a Shared Service

Once a hub is in place, it functions operationally much like any other shared enterprise service, similar in governance character to a shared database platform or a shared network segment used by multiple applications. This means the team operating the hub should apply the same service-level discipline expected of any shared infrastructure: a published service level target for availability and processing latency, a defined intake process for onboarding a new spoke, and a clear escalation path distinct from any individual spoke system's own incident process. Treating the hub informally, as though it were simply an extension of whichever team happened to build it first, tends to produce exactly the kind of undocumented, single-point-of-knowledge risk this article warns against in the context of the accidental hub anti-pattern described in Section 3.4.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

4.3 Hub Resilience Considerations

Consideration	Guidance
Single point of failure risk	Design the hub with redundancy proportional to the criticality of the data flows it carries
Hub capacity planning	Size against peak aggregate load across all spokes, not any single spoke's average load
Hub ownership	Assign a dedicated operational owner distinct from any individual spoke system's owner

Table.5. Hub resilience considerations for the hub-and-spoke design pattern.

V. DESIGN PATTERN 3: PUBLISH-SUBSCRIBE FILE DROP

Figure 3 illustrates the publish-subscribe pattern, in which a single publisher writes to a shared location that multiple subscriber systems independently read from, without the publisher needing awareness of how many subscribers exist.

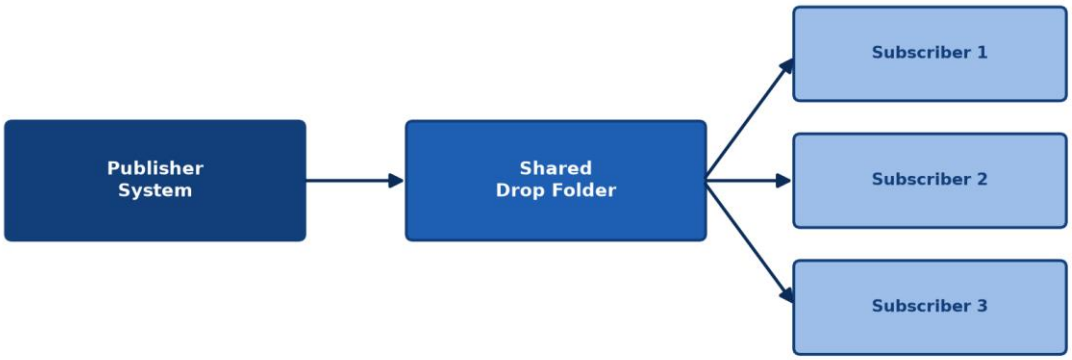

Figure.3. Publish-subscribe pattern with one publisher writing to a shared drop folder consumed independently by three subscriber systems.

5.1 Strengths and Limitations

Dimension	Publish-Subscribe Assessment
Implementation simplicity	Moderate; requires shared location access management
Publisher awareness of consumers	None required; new subscribers can be added without publisher changes
Governance visibility	Moderate; requires monitoring subscriber consumption independently
Best-fit scenario	One authoritative source feeding many independent consumers

Table.6. Strengths and limitations of the publish-subscribe MS-to-MS design pattern.

5.2 A Worked Example: The Publisher Who Didn't Know Its Own Audience

Consider a product master data system that began publishing daily catalog updates to a shared drop folder for a single original consumer, an early-generation e-commerce platform, several years ago. Over time, four additional systems independently discovered the same drop folder and began consuming the same files, each added by a different team at a different point, none of whom formally notified the product master data team of the new dependency. The publishing team, when planning a schema change to add a new required field, reasonably believed they had exactly one consumer to coordinate with and proceeded on that basis.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

The schema change broke three of the four undocumented consumers immediately, each of which had built parsing logic that assumed the old, narrower schema and had no tolerance for an added field. The publishing team had no way to have known this in advance, since nothing in the publish-subscribe pattern itself required these consumers to register their dependency anywhere the publisher could see it. Recovering required an emergency rollback of the schema change and a subsequent audit effort to identify every actual consumer before attempting the change again, an effort that a simple subscriber registry, maintained from the very first additional consumer onward, would have made entirely unnecessary.

5.3 Subscriber Independence and Its Trade-Offs

The defining advantage of publish-subscribe, that subscribers can be added without publisher changes, carries a corresponding governance trade-off: the publisher has no built-in mechanism for knowing which systems currently depend on its output. A subscriber can quietly begin consuming the shared drop location without the publishing team's knowledge, and a subsequent change to the publisher's file format can break that unknown subscriber with no advance warning to either party. Maintaining an explicit subscriber registry, even though the pattern does not technically require one, is the practical mitigation for this trade-off.

VI. PATTERN SELECTION FRAMEWORK

Scenario	Recommended Pattern
Two systems, stable long-term relationship	Point-to-point
Four or more systems exchanging data with each other	Hub-and-spoke
One authoritative source, many independent consumers	Publish-subscribe
Growing landscape with uncertain future system count	Hub-and-spoke, for governance scalability

Table.7. Pattern selection framework mapping common integration scenarios to a recommended design pattern.

- Reassess pattern choice whenever system count crosses a threshold of roughly four to five participants.
- Mixed landscapes, combining more than one pattern for different data domains, are common and appropriate.

6.1 A Worked Example: Choosing Between Hub-and-Spoke and Publish-Subscribe

Consider an organization with a central product master data system feeding six downstream systems: an e-commerce platform, a warehouse management system, three regional ERP instances, and a business intelligence platform. All six consume the same product data, but none of them send data back to the product master system as part of this particular flow. This is a textbook one-to-many scenario, and the pattern selection framework in Table 6 points toward publish-subscribe rather than hub-and-spoke, since a hub is designed to broker many-to-many exchange and would add unnecessary intermediary complexity for a flow that is structurally only ever one-directional from a single source.

Now consider the same organization's order data, which flows from the e-commerce platform to the warehouse system, from the warehouse system back to the regional ERP instances for financial posting, and from the regional ERP instances to the business intelligence platform for reporting. This is a genuine many-to-many pattern with data flowing in multiple directions across the same system population, and here hub-and-spoke is the better fit, since a shared drop folder model does not naturally express the directional, multi-hop routing this data actually requires.

The two scenarios above sit within the very same organization, drawing on the same integration team and the same underlying transport infrastructure, and yet they warrant two different design patterns. This is the central practical lesson of the selection framework in Table 6: pattern choice is a property of the data flow being designed, not a property of the organization or its integration team's general preference. A team that has standardized on hub-and-spoke for its transactional flows should not feel compelled to force a genuinely one-to-many master data flow through the same pattern purely for consistency's sake, and a team comfortable with publish-subscribe for master data distribution should not attempt to stretch that same pattern to express multi-directional transactional routing it was never designed to carry.

VII. FILE LIFECYCLE MANAGEMENT

Figure 4 illustrates the five-state lifecycle a well-governed file should pass through, with an explicit rejection path for files that fail validation.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

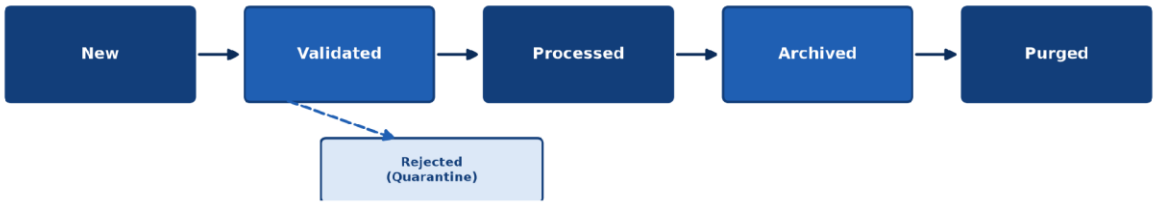

Figure.4. File lifecycle states from creation through validation, processing, archival, and eventual purge, with a rejection path to quarantine.

7.1 Why an Explicit Lifecycle Matters More at Scale

In a single point-to-point integration, an implicit file lifecycle, arrive, process, delete, causes little practical harm even when it is never written down anywhere, since there is only one team and one pair of systems that need to agree on what happens to a file after it lands. That informal approach stops scaling the moment a hub or publish-subscribe pattern introduces multiple independent teams reasoning about the same file population, each potentially making different assumptions about when it is safe to delete a processed file, how long a rejected file should sit in quarantine before someone is expected to look at it, or whether an archived file remains queryable or has effectively been removed from active use.

An explicit, documented lifecycle, of the kind illustrated in Figure 4, resolves this ambiguity once, for every integration in the landscape, rather than requiring each new integration to independently reinvent an answer to the same set of questions. This is another instance of the general principle introduced in Section 10: the specific lifecycle model matters less than its consistent, documented application across every integration a landscape operates.

7.2 Lifecycle State Definitions

State	Definition
New	File has arrived but not yet been validated
Validated	File has passed structural and content validation
Processed	File's content has been successfully loaded into the target system
Archived	File is retained for audit purposes but no longer active
Purged	File has been removed per the organization's retention policy
Rejected (Quarantine)	File failed validation and is held for manual review

Table.8. File lifecycle state definitions for MS-to-MS batch exchange.

7.3 Retention Policy by Lifecycle State

State	Illustrative Retention Guidance
Processed	Retain in place for a short operational window, typically 24 to 72 hours
Archived	Retain per organizational audit policy, often 1 to 7 years depending on data domain
Rejected (Quarantine)	Retain until manually resolved, with an escalation trigger for aged items

Table.9. Illustrative retention guidance by file lifecycle state

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

VIII. SCHEDULING MODELS: POLLING AND EVENT TRIGGERS

Figure 5 compares the two dominant scheduling models used to detect and process arriving files.

Figure.5. Comparison of polling and event-trigger scheduling models for detecting and processing arriving files.

8.1 Scheduling Model Comparison

Attribute	Polling	Event Trigger
Latency	Bounded by polling interval	Near-immediate
Implementation complexity	Low	Moderate, requires event infrastructure
Resource usage	Periodic overhead regardless of activity	Activity-proportional
Best-fit scenario	Simple landscapes, moderate latency tolerance	Latency-sensitive or high-volume landscapes

Table.10. Comparison of polling and event-trigger scheduling models.

8.2 Latency Budgets Across a Multi-Hop Flow

When a file passes through a hub or a chain of subscribers rather than moving directly between two systems, the scheduling model chosen at each hop contributes independently to total end-to-end latency. A flow using polling at both the source-to-hub hop and the hub-to-target hop, each on a fifteen-minute interval, can accumulate up to thirty minutes of latency purely from polling delay before accounting for actual transfer and processing time, even though each individual hop looks perfectly reasonable in isolation. Programs setting an end-to-end latency requirement for a multi-hop flow should budget this compounding effect explicitly, deciding which specific hops can tolerate polling delay and which need event-triggered processing to keep the cumulative latency within the required bound, rather than applying a single scheduling model uniformly across every hop without regard to how the delays stack up.

8.3 Hybrid Scheduling Approaches

Many landscapes benefit from applying both models selectively rather than standardizing on a single approach across every integration. High-volume, latency-sensitive flows justify the additional complexity of event-triggered processing, while lower-volume, latency-tolerant flows are typically well served by simple polling on a fixed interval, avoiding the operational overhead of event infrastructure where it provides little practical benefit.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

IX. NAMING AND DIRECTORY CONVENTIONS

Convention Element	Recommendation
File name structure	Include source system, data domain, and timestamp
Directory structure	Separate inbound, outbound, processed, and quarantine directories per integration
Control file naming	Mirror the data file name with a distinct extension or suffix
Versioning	Include a schema version identifier in the file name or header record

Table.11. Recommended naming and directory conventions for MS-to-MS FTP integration.

Consistent naming conventions applied uniformly across every integration in a landscape, rather than independently invented per integration, materially reduce the effort required to build shared monitoring and governance tooling across the full integration portfolio.

9.1 Sample Naming Convention

Component	Example
Source system code	ERP1
Data domain	ORDERS
Timestamp	20240415T0630
Full file name	ERP1_ORDERS_20240415T0630.dat
Control file name	ERP1_ORDERS_20240415T0630.ctl

Table.12. Sample naming convention applied consistently across an integration landscape.

A convention's value comes almost entirely from its consistent application, not from any particular structural choice being objectively superior to another reasonable alternative. A landscape that picks a slightly different but internally consistent convention will realize nearly all the same governance and tooling benefits as one that adopts the exact structure shown in Table 12. The failure mode this section warns against is not choosing the wrong convention, but allowing each new integration to invent its own, producing a landscape where no two integrations follow the same pattern and no shared tooling can be built to monitor or manage them uniformly.

X. FILE FORMAT AND SCHEMA DESIGN

Format	Advantage	Trade-Off
Delimited text	Simple, widely supported, human-readable	Ambiguous if delimiter appears in data
Fixed-width text	Compact, fast to parse	Brittle to field length changes
XML	Self-describing, supports nested structure	Larger file size, more parsing overhead
JSON	Widely supported by modern tooling, flexible structure	Less common in legacy batch contexts

Table.13. Common file formats for MS-to-MS batch data exchange, with advantages and trade-offs.

10.1 Schema Governance Across Multiple Systems

Schema governance is considerably harder in an MS-to-MS landscape than in a two-system integration, since a schema change proposed by one system's team may affect several other systems simultaneously, each with its own release cycle and change approval process. A schema registry, even a simple shared document listing every active schema version and its consuming systems, gives the landscape a single reference point for assessing the blast radius of a proposed change before it is approved.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

chema Governance Practice	Purpose
Central schema registry	Tracks every active schema version and its consuming systems
Backward-compatible change preference	Minimizes the need for simultaneous multi-system cutover
Deprecation window policy	Gives consuming systems defined lead time before an old schema version is retired

Table.14. Schema governance practices for multi-system batch data exchange.

XI. ERROR HANDLING AND RETRY STRATEGY

Figure 6 illustrates a complete retry and error handling flow, from initial transfer attempt through retry escalation to a dead letter queue for transfers that never succeed.

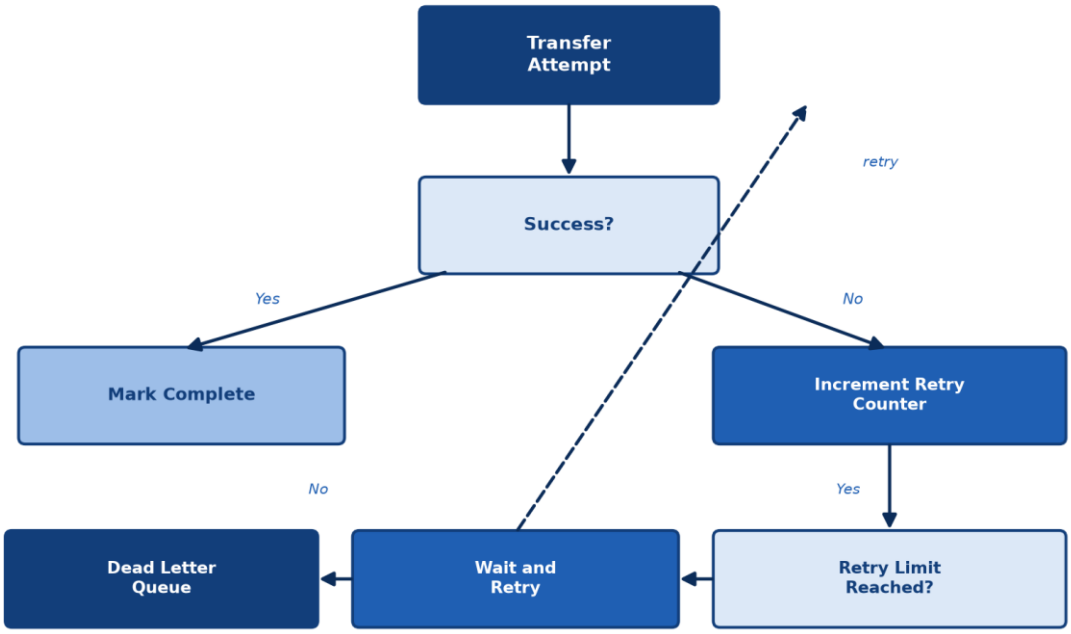

Figure.6. Retry and error handling flow, escalating a failed transfer through a bounded retry loop before routing to a dead letter queue.

11.1 Retry Policy Parameters

Parameter	Illustrative Guidance
Retry count limit	3 to 5 attempts before dead-lettering
Backoff interval	Increasing delay between attempts, not a fixed interval
Dead letter review cadence	At least once per business day
Alert threshold	Immediate alert on dead-letter, not only on final failure count

Table.15. Illustrative retry policy parameters for MS-to-MS FTP transfers.

11.2 Retry Storms and Their Prevention

A poorly designed retry policy can itself become an incident. If a target system experiences an extended outage and every affected integration retries on a short, fixed interval without backoff, the resulting surge of near-simultaneous retry attempts, arriving the moment the target system begins recovering, can overwhelm it again before it has had a

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

chance to stabilize, a phenomenon commonly described as a retry storm. The increasing backoff interval recommended in Table 10 exists specifically to prevent this: by spacing out retry attempts progressively rather than firing them all at the same fixed cadence, a recovering target system receives a manageable, staggered trickle of retry traffic rather than a synchronized flood.

This risk is particularly acute in a hub-and-spoke topology, where a single hub outage can simultaneously affect every spoke's retry logic at once. Programs operating a hub-and-spoke landscape should confirm that backoff intervals are randomized slightly across spokes, in addition to increasing over successive attempts, so that even a large spoke population does not inadvertently synchronize its retry attempts against a recovering hub.

11.3 Worked Example: Distinguishing Transient From Persistent Failures

Consider a hub-and-spoke integration where a spoke system experiences a brief network interruption lasting under two minutes. A transfer attempt during that window fails, but the bounded retry policy in Table 10 succeeds on the second attempt once the network recovers, and the failure never becomes operationally visible beyond a routine retry log entry. Compare this against a scenario where the spoke system's credentials have expired: every retry attempt fails identically, the retry counter exhausts its limit, and the transfer correctly routes to the dead letter queue.

The distinction matters for alerting design. A monitoring configuration that raises an alert on every individual failed attempt would generate noise for the first scenario, where the system is working exactly as intended, while a configuration that only alerts on dead-letter events, per the threshold in Table 10, correctly stays silent for the transient case and correctly escalates the persistent one. Designing alerts around the dead-letter threshold rather than individual attempt failures is what keeps this distinction from becoming a source of alert fatigue.

XII. SECURITY ARCHITECTURE

Figure 7 illustrates a layered security architecture, with each layer addressing a distinct threat surface.

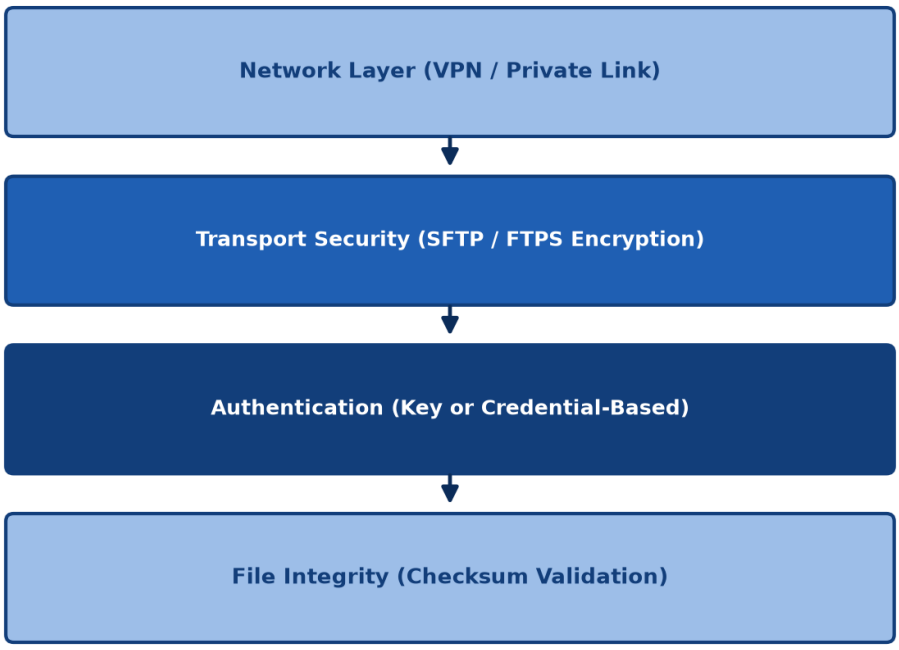

Figure.7. Layered security architecture for MS-to-MS FTP integration, spanning network, transport, authentication, and file integrity layers

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

12.1 Security Layer Definitions

Layer	Purpose
Network layer	Restricts network-level reachability between systems
Transport security	Encrypts data in transit using SFTP or FTPS
Authentication	Confirms the identity of the connecting system or user
File integrity	Confirms the transferred file was not altered in transit

Table.16. Security layer definitions for MS-to-MS FTP integration

12.2 Credential Management Across Multiple Systems

Practice	Rationale
Dedicated technical account per integration	Limits the blast radius of a single compromised credential
Scheduled credential rotation	Reduces the window of exposure if a credential is compromised undetected
Centralized credential vault	Avoids credentials being stored in plain text within scheduler or script configuration
Least-privilege directory access	Restricts each technical account to only the specific directories it requires

Table.17. Credential management practices for MS-to-MS FTP integration.

XIII. MONITORING AND ALERTING

Metric	Purpose	Illustrative Alert Threshold
Transfer success rate	Detects transport reliability issues	Below 99 percent over 24 hours
Dead letter queue depth	Detects unresolved persistent failures	Any non-zero depth after review cadence
File arrival delay	Detects a missing or late expected file	Beyond the scheduled window plus tolerance
Quarantine volume	Detects a validation or upstream data quality issue	Above a defined daily threshold

Table.18. Core monitoring metrics for MS-to-MS FTP integration.

13.1 Sample Monitoring Dashboard Extract

Integration	Last Transfer Status	Elapsed Time	Dead Letter Queue Depth
ERP1 to WMS (Orders)	Success	4 minutes	0
ERP1 to Hub (Master Data)	Success	6 minutes	0
Hub to Regional ERP (EMEA)	Delayed	38 minutes	0
Hub to BI Platform (Reporting)	Success	3 minutes	1

Table.19. Sample daily monitoring dashboard extract for a set of active MS-to-MS integrations.

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

XIV. GOVERNANCE AND CHANGE MANAGEMENT

Figure 8 illustrates a standard change governance workflow applicable to any change affecting a shared MS-to-MS integration component.

Figure.8. Standard change governance workflow for MS-to-MS integration changes, from request through post-change validation

14.1 Governance Roles

Role	Responsibility
Integration Owner	Accountable for the overall health of a given integration
Source System Analyst	Responsible for export configuration and source-side data quality
Target System Analyst	Responsible for import configuration and target-side validation
Integration Operations Team	Responsible for transport infrastructure and monitoring

Table.20. Governance roles and responsibilities for MS-to-MS FTP integration

14.2 Escalation Matrix

Condition	Escalation Level	Notified Role
Dead letter queue depth above zero for over 4 hours	Integration operations lead	Integration owner
Schema change proposed affecting more than one consuming system	Program governance	All affected system analysts
Recurring incident on the same integration within 30 days	Program governance	Integration owner, operations team

Table.21. Escalation matrix for MS-to-MS integration incidents and change proposals.

XV. DATA QUALITY AND RECONCILIATION

Reconciliation Check	Purpose
Record count reconciliation	Confirms source and target record counts match
Checksum validation	Confirms file content integrity after transfer
Duplicate file detection	Confirms a file is not processed more than once
Schema version validation	Confirms the file conforms to the expected schema version

Table.22. Core reconciliation checks for MS-to-MS FTP integration.

15.1 Reconciliation Ownership in a Multi-Hop Landscape

Reconciliation is straightforward for a single point-to-point connection: compare source and target counts directly. It becomes considerably more involved once data flows through a hub or a chain of subscribers, since a discrepancy could originate at the original source, be introduced during hub processing, or occur during final delivery to a specific spoke.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

Reconciliation checks should therefore be implemented at every hop in a multi-hop flow, not only at the final destination, so that a discrepancy can be localized to the specific hop that introduced it rather than requiring an end-to-end investigation across the entire chain every time.

XVI. PERFORMANCE AND SCALABILITY

Scaling Dimension	Consideration
File size growth	Plan for compression or file splitting as volume grows
Concurrent transfer volume	Size hub or broker infrastructure against peak, not average, load
Directory scan performance	Avoid excessive file accumulation in a single directory
Downstream processing capacity	Confirm target system load capacity keeps pace with arrival rate

Table.23. Performance and scalability considerations for MS-to-MS FTP integration

16.1 Illustrative Throughput Benchmarks

Pattern	Illustrative Sustained Throughput	Primary Constraint
Point-to-point	3,500 to 5,000 records per hour per connection	Individual connection bandwidth
Hub-and-spoke	6,000 to 9,000 records per hour, aggregate across spokes	Hub processing capacity
Publish-subscribe	5,000 to 7,000 records per hour, per subscriber	Shared storage read contention under high subscriber count

Table.24. Illustrative throughput benchmarks by design pattern, compiled from published implementation guidance.

XVII. MULTI-SYSTEM TOPOLOGY MANAGEMENT

Figure 9 illustrates why point-to-point connectivity becomes unmanageable as system count grows: five systems connected point-to-point require ten separate connections, each independently configured, secured, and monitored.

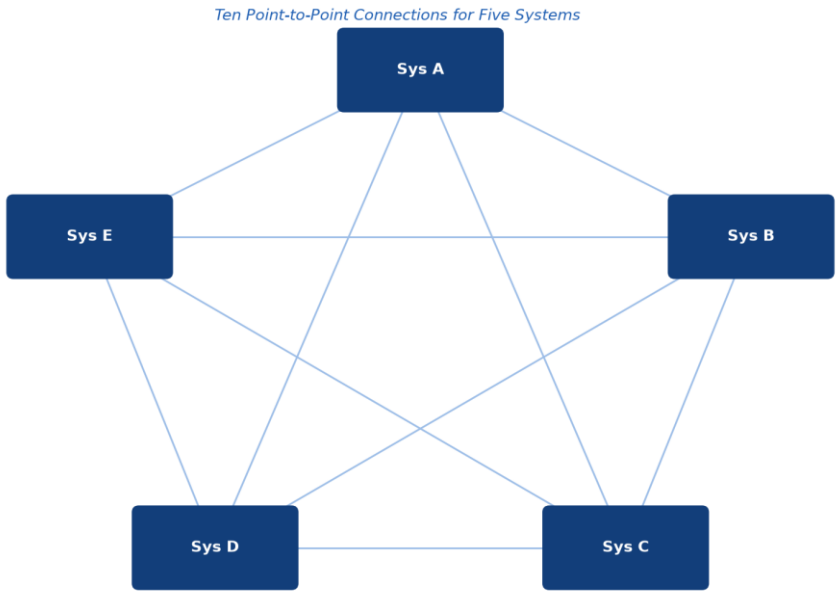

Figure.9. Point-to-point mesh topology for five systems, illustrating the ten separate connections required, each needing independent configuration and governance.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

17.1 Why This Growth Pattern Surprises Most Teams

The quadratic growth in point-to-point connection count is easy to state as a fact but surprisingly easy to underestimate in practice, because a landscape rarely adds several systems at once. It typically adds one system at a time, over months or years, and each individual addition feels like a small, incremental change: one new connection, configured, tested, and deployed like the last one. The cumulative effect of many individually small additions is what produces the underlying quadratic curve, and because each step feels manageable in isolation, teams frequently do not notice they have crossed into unmanageable territory until a cross-cutting need, such as a security audit requiring a complete connection inventory, forces someone to count every connection at once and confront the actual total.

This dynamic is precisely why the pattern selection framework in Section 7 recommends reassessing topology choice at a defined system count threshold, rather than waiting for a crisis to force the reassessment. A deliberate checkpoint at four or five systems, built into the standard integration intake process, catches the transition while migration to hub-and-spoke is still a modest, planned effort rather than an emergency response to an already unmanageable mesh.

17.2 Connection Count by Topology

System Count	Point-to-Point Connections	Hub-and-Spoke Connections
3 systems	3	3
5 systems	10	5
8 systems	28	8
12 systems	66	12

Table.25. Connection count comparison between point-to-point and hub-and-spoke topologies as system count grows.

17.3 Topology Migration Approach

Migrating an existing point-to-point mesh to a hub-and-spoke topology is rarely done as a single cutover across every connection simultaneously. A staged approach, migrating one system's connections to the hub at a time while its remaining point-to-point connections continue operating unchanged, reduces cutover risk and allows each migrated connection to be validated independently before the next is attempted.

1. Select the highest-value or highest-risk system pair as the first migration candidate.
2. Build and validate the equivalent hub-and-spoke connection in parallel with the existing point-to-point connection.
3. Cut over that specific connection once validated, retiring the point-to-point link.
4. Repeat for each remaining system, prioritizing by risk and value.

XVIII. TOOLING AND MANAGED FILE TRANSFER PLATFORMS

Tooling Category	Role
Managed file transfer (MFT) platform	Centralized transport, monitoring, and governance across many integrations
Workflow orchestration tooling	Sequences multi-step file processing pipelines
Enterprise scheduler	Coordinates polling-based scheduling across systems
Centralized logging platform	Aggregates transfer and processing logs across the integration portfolio

Table.26. Tooling categories supporting MS-to-MS FTP integration at scale

International Journal of Research Publications in Engineering, Technology and Management (IJRPETM)

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

18.1 Build Versus Buy Considerations

Consideration	Favors Building Custom Tooling	Favors a Commercial MFT Platform
Integration count	Small, stable count	Large or actively growing count
Compliance requirements	Minimal	Extensive audit and compliance reporting needs
Internal engineering capacity	Available and willing to maintain custom tooling long-term	Limited, preferring vendor-supported tooling

Table.27. Build-versus-buy considerations for managed file transfer tooling.

XIX. IMPLEMENTATION METHODOLOGY

The methodology below sequences pattern selection, convention design, and validation into a coherent implementation path applicable whether the program is building a new MS-to-MS landscape or consolidating an existing one.

1. Inventory participating systems and classify the integration scenario per the framework in Section 7.
2. Select the design pattern per Section 7 and confirm it against expected future system count growth.
3. Design file lifecycle, naming, and schema conventions per Sections 8, 10, and 11.
4. Implement error handling and retry logic per Section 12.
5. Implement the security architecture per Section 13 before any production data flows.
6. Establish monitoring and alerting per Section 14.
7. Execute reconciliation validation per Section 16 against a production-representative volume.
8. Document governance ownership per Section 15 before cutover.

Phase	Illustrative Duration
System inventory and pattern selection	1 to 2 weeks
Convention design and security setup	2 to 3 weeks
Build and initial testing	3 to 4 weeks
Monitoring, reconciliation, and validation	1 to 2 weeks

Table.28. Illustrative phase timeline for implementing an MS-to-MS FTP integration program.

XX. RISK FACTORS AND MITIGATION

The risk factors below recur across most MS-to-MS FTP landscapes, each paired with the section of this article addressing the corresponding mitigation in full.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

Risk Factor	Potential Impact	Mitigation
Point-to-point sprawl	Unmanageable connection count as system count grows	Consolidate to hub-and-spoke per Section 18
Missing control file convention	Partial file reads and corrupted data	Adopt control file conventions per Section 8
Plain FTP in production	Credential and data interception risk	Migrate to SFTP or FTPS per Section 13
No reconciliation process	Undetected data loss between systems	Implement checks per Section 16
Undocumented schema ownership	Uncoordinated breaking changes	Assign interface ownership per Section 15

Table.29. Common risk factors in MS-to-MS FTP integration, with mitigations

XXI. COST AND RESOURCING CONSIDERATIONS

Cost Driver	Scaling Factor
Connection configuration effort	Scales with connection count, per the topology comparison in Table 15
Managed file transfer platform licensing	Scales with participating system count and file volume
Monitoring and operations effort	Scales with active integration count and incident volume
Security control implementation	Largely fixed per integration, regardless of topology

Table.30. Cost and resourcing scaling factors for MS-to-MS FTP integration

Programs frequently underbudget monitoring and operations effort specifically, since it is the cost driver most sensitive to incident volume rather than to integration count alone. A landscape with well-designed conventions, per Sections 8 through 11, and disciplined error handling, per Section 12, generates substantially fewer incidents per integration than one without, which is why investing in those foundational practices early tends to reduce total cost of ownership over the life of the landscape rather than merely shifting cost from build to run.

XXII. MIGRATION AND MODERNIZATION PATHWAYS

Modernization Driver	Recommended Pathway
Point-to-point sprawl reaching unmanageable scale	Consolidate to hub-and-spoke per Section 18
Need for sub-minute latency on a specific data flow	Selectively migrate that flow to event-based integration
Growing governance burden across many integrations	Adopt a managed file transfer platform per Section 19
No specific pain point	Continue with the existing pattern; modernization should be driven by evidence, not by age alone

Table.31. Modernization drivers and recommended pathways for MS-to-MS FTP integration

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

XXIII. A WORKED EXAMPLE: SELECTIVE MODERNIZATION

Consider a landscape of eight systems connected through a mature hub-and-spoke topology, where one specific data flow, inventory availability signals feeding a customer-facing website, has grown business-critical enough that the existing fifteen-minute polling interval no longer meets the business requirement for near-real-time stock visibility. Rather than replacing the entire hub architecture, which serves the other seven data flows adequately, the organization migrates only this one flow to an event-based mechanism, leaving the hub-and-spoke pattern in place for everything else.

This selective approach reflects the modernization principle established throughout this section: the trigger for change is a specific, demonstrated business requirement attached to a specific data flow, not a general belief that newer integration technology is inherently preferable to file-based batch exchange. The seven other data flows continue to be well served by the existing pattern, and migrating them would add complexity without a corresponding business benefit.

XXIV. INDUSTRY ADOPTION PATTERNS

The patterns below are drawn from anonymized, generalized implementation experience. No specific organization is identified.

Industry Pattern	Dominant Design Pattern	Typical System Count
Manufacturing and multi-plant operations	Hub-and-spoke	5 or more plant and supplier systems
Financial services back-office processing	Point-to-point for core systems, hub-and-spoke for peripheral feeds	Mixed, often 3 to 10
Retail and distribution	Publish-subscribe for master data, hub-and-spoke for transactional data	5 or more

Table.32. Cross-industry summary of dominant MS-to-MS design pattern and typical system count.

Despite these differing emphases, all three patterns share the same underlying discipline established throughout this article: select a design pattern deliberately based on actual system count and data flow direction, per Section 7, rather than defaulting to whichever pattern the first integration in the landscape happened to use.

24.1 A Note on Organizational Readiness

Adopting hub-and-spoke or publish-subscribe patterns is not purely a technical decision; it also requires organizational readiness to accept centralized governance over a component, the hub or shared drop location, that no single system team fully owns. Organizations accustomed to fully autonomous system teams sometimes resist this shift, since it introduces a shared dependency and a shared governance process where none previously existed. Framing this shift explicitly, as a trade-off of some team-level autonomy in exchange for landscape-level manageability, tends to secure buy-in more effectively than presenting the topology change as a purely technical upgrade.

XXV. COMMON PITFALLS

Most of the pitfalls below trace back to treating each new integration as an isolated technical task rather than as one component of a shared, evolving landscape governed by consistent conventions.

- Adding each new system connection as an independent point-to-point link without reassessing topology.
- Omitting a control file convention and relying on file arrival timing alone.
- Deploying plain, unencrypted FTP in production for any system exchanging business data.
- Treating naming and directory conventions as a per-integration decision rather than a landscape-wide standard.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

- Allowing dead letter queues to accumulate without a defined review cadence.

XXVI. BEST PRACTICES AND RECOMMENDATIONS

- Select a design pattern deliberately per Section 7, and revisit that choice as system count grows.
- Apply naming, directory, and schema conventions consistently across every integration in the landscape.
- Adopt SFTP or FTPS as the default transport for every production integration.
- Implement bounded retry logic with a dead letter queue, per Section 12, for every integration.
- Establish monitoring and reconciliation as standard components of every integration, not optional additions.
- Assign clear governance ownership before any integration reaches production.

XVII. FUTURE OUTLOOK

As of early 2024, organizations were increasingly adopting managed file transfer platforms to bring centralized governance to MS-to-MS FTP landscapes that had grown organically over many years, without necessarily replacing the underlying file-based transport itself. Continued growth in hybrid landscapes, combining batch file exchange for stable, high-volume data flows with selective adoption of event-based integration for latency-sensitive flows, is expected as organizations increasingly recognize that pattern choice should be evidence-driven and made per data flow rather than applied uniformly across an entire landscape.

This trend is likely to be reinforced by growing adoption of centralized observability tooling that spans both batch file transfers and event-based integration within a single monitoring pane, reducing the operational cost of running a hybrid landscape and making the evidence-driven, per-flow modernization approach recommended throughout this article increasingly practical to sustain at scale.

XVIII. CONCLUSION

MS-to-MS FTP integration is a distinct design problem from simple two-system file exchange, and treating it as though it were not, by extending point-to-point connections indefinitely as system count grows, is one of the most common and most avoidable sources of long-term integration technical debt. The design patterns, supporting practices, and reference tables presented in this article give integration architects a structured foundation for building MS-to-MS FTP landscapes that remain governable as they scale.

Programs that select a design pattern deliberately, apply consistent conventions across every integration, and treat security, monitoring, and reconciliation as standard rather than optional components are substantially better positioned to operate a growing multi-system integration landscape reliably over the long term.

The recurring theme across the worked examples in this article, the accidental hub in Section 3.4 and the hub-versus-publish-subscribe decision in Section 7.1, is that MS-to-MS landscapes rarely fail because of a single poor decision. They accumulate risk through a series of individually convenient shortcuts, each reasonable in isolation, that no one revisits until the landscape has grown large enough that the accumulated inconsistency becomes impossible to ignore. Applying the structured patterns and conventions in this article from the earliest integrations onward is considerably less costly than retrofitting them onto a landscape that has already grown organically for years.

REFERENCES

1. Erl, T. (2005). Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall.
2. Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley.
3. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
4. Hohpe, G., & Woolf, B. (2003). Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley.
5. International Organization for Standardization. (2013). ISO/IEC 27001:2013, Information Security Management Systems: Requirements. ISO.
6. Josuttis, N. M. (2007). SOA in Practice: The Art of Distributed System Design. O'Reilly Media.

|www.ijrpetm.com | ISSN: 2454-7875 | editor@ijrpetm.com |A Bimonthly, Peer Reviewed & Scholarly Journal|

||Volume 7, Issue 2, March-April 2024||

DOI:10.15662/IJRPETM.2024.0702009

7. Linthicum, D. S. (2003). Next Generation Application Integration: From Simple Information to Web Services. Addison-Wesley.
8. National Institute of Standards and Technology. (2020). NIST Special Publication 800-53 Revision 5: Security and Privacy Controls for Information Systems and Organizations. NIST.
9. Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
10. Richardson, C. (2018). Microservices Patterns: With Examples in Java. Manning Publications.

Appendix A: Glossary of Key Terms

The terms below are used throughout this article with the specific meanings defined here.

Term	Definition
MS-to-MS	Multi-System to Multi-System; batch integration involving more than two participating systems.
Point-to-Point	A direct connection between exactly two systems.
Hub-and-Spoke	A topology routing all exchange through a central broker.
Publish-Subscribe	A pattern where one publisher feeds many independent subscribers.
Control File	A companion file signaling that a corresponding data file is complete.
Dead Letter Queue	A holding area for transfers that failed after exhausting retry attempts.
Managed File Transfer (MFT)	A platform providing centralized governance and monitoring for file-based integration.

Table.33. Glossary of key terms referenced throughout this article.

Appendix B: Sample Design Pattern Selection Checklist

The checklist below consolidates the article's guidance into a single sequential reference for a team selecting and implementing an MS-to-MS FTP design pattern for a new or existing data flow.

- Participating system count confirmed and documented.
- Expected future growth in system count assessed.
- Data flow direction pattern confirmed (one-to-one, one-to-many, many-to-many).
- Latency requirement classified per data flow.
- Design pattern selected per the framework in Section 7 and documented with rationale.
- Naming, directory, and schema conventions confirmed against the landscape-wide standard.
- Security architecture confirmed per Section 13 before production use.
- Monitoring and reconciliation implemented before cutover.