iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

# FTP INTEGRATION PATTERNS ACROSS ERP BOUNDARIES: DEPENDENT DEMAND FLOW ARCHITECTURES

**Sivasankararao Kavuri**
Data Migration Lead

## 01 Executive Summary

File Transfer Protocol remains, more than two decades after more modern integration technologies emerged, one of the most widely used mechanisms for exchanging demand and supply data between independently operated ERP systems. It persists not out of inertia alone but because batch file exchange over FTP or its secure variants remains simple, broadly compatible, and adequate for a meaningful share of integration scenarios, particularly those involving dependent demand signals such as planned orders and purchase requisitions that do not require sub-second latency.

This article presents a structured treatment of FTP-based integration patterns specifically for dependent demand flow across ERP boundaries. It defines dependent demand and its propagation mechanics, compares three principal FTP integration patterns, and provides architecture, file design, scheduling, security, performance, governance, and migration guidance. The article combines structural diagrams, illustrative benchmark charts, and an extensive set of reference tables to give integration architects a comprehensive, practical foundation.

Illustrative benchmark data presented in Section 12 shows a hub-and-spoke broker pattern achieving roughly 60 percent higher throughput than simple point-to-point batch exchange, while Section 8 shows that demand signal latency scales sharply with batch frequency, from under half an hour at 15-minute intervals to well over three days at weekly intervals.

## 02 Introduction

### › 2.1 Why FTP Still Matters for ERP-to-ERP Integration

Modern integration platforms, API gateways, and event-streaming backbones have not fully displaced file-based batch exchange, particularly at ERP boundaries involving legacy systems, external supplier or distributor systems outside direct organizational control, or regulatory contexts where a durable, inspectable file artifact is preferred over an ephemeral API call. FTP-family protocols, despite their age, remain a pragmatic default in exactly these contexts.

- Broad compatibility with legacy and third-party ERP systems that may not expose modern APIs.
- Simplicity of implementation and troubleshooting relative to event-driven or API-based alternatives.
- A durable, inspectable file artifact that supports audit and reconciliation requirements.
- Lower ongoing operational complexity for integration scenarios that do not require real-time latency.

### › 2.2 Dependent Demand: A Brief Primer

Dependent demand refers to demand for a component, subassembly, or raw material that is derived mathematically from the independent demand for a finished good, rather than forecast or ordered directly. Material requirements planning logic calculates dependent demand by exploding a bill of materials against a finished good's demand, and this calculation frequently needs to cross an ERP system boundary, for example when a component is sourced from a separately operated plant, division, or supplier running its own ERP instance.

### › 2.3 Purpose and Scope

This article's purpose is to give integration architects and supply chain systems teams a structured reference for designing, operating, and governing FTP-based integration patterns specifically for dependent demand flow. The scope covers architecture, patterns, file design, scheduling, security, performance, governance, and migration considerations, illustrated with diagrams, benchmark charts, and reference tables throughout.

## 03 Dependent Demand Flow Fundamentals

### › 3.1 Independent Versus Dependent Demand

| Attribute | Independent Demand | Dependent Demand |
|---|---|---|
| Source | Customer order or statistical forecast | Calculated from a parent item's bill of materials |
| Planning method | Forecasting and order management | Material requirements planning (MRP) explosion |
| Typical MRP element | Sales order, forecast | Planned order, purchase requisition |
| Cross-ERP relevance | Usually originates within a single system | Frequently must cross ERP boundaries for sourced components |

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

**Table 1.** *Comparison of independent and dependent demand characteristics relevant to cross-ERP integration.*
Figure 1 illustrates how a single unit of independent demand for a finished good explodes into dependent demand across multiple components and, in turn, multiple raw materials, several of which may be sourced from a plant or supplier operating a distinct ERP system.

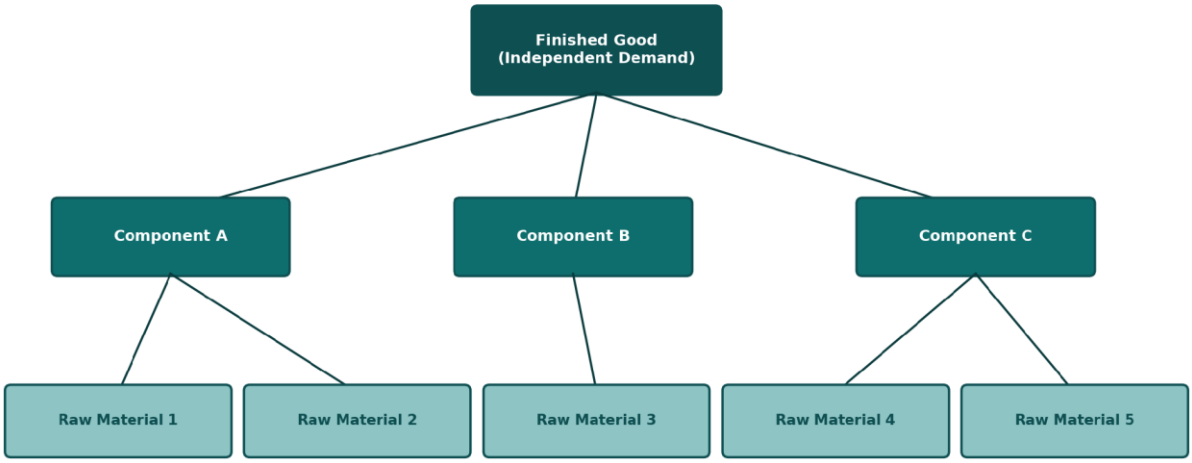



**Figure 1.** *Dependent demand propagation from a finished good's independent demand through components to raw materials, illustrating the multi-level explosion that MRP logic must resolve.*

### › 3.2 MRP Elements That Commonly Cross ERP Boundaries

| MRP Element | Typical Cross-Boundary Scenario |
|---|---|
| Planned order | Component sourced from a separately operated plant ERP |
| Purchase requisition | Raw material sourced from an external supplier ERP |
| Stock transfer requirement | Inventory rebalancing between plant ERPs |
| Forecast or schedule agreement release | Long-term demand visibility shared with a supplier ERP |

**Table 2.** *Common MRP elements requiring propagation across an ERP system boundary.*

### › 3.3 Bill of Materials Depth and Cross-Boundary Complexity

The number of ERP boundaries a single finished good's demand must cross grows with bill of materials depth and with how sourcing responsibility is distributed across the organization. A shallow, single-level bill of materials sourced entirely within one plant crosses no boundary at all. A deep, multi-level bill of materials sourced across several plants and external suppliers, by contrast, may require dependent demand to propagate across three or four distinct ERP boundaries before reaching its final raw material source.

| Bill of Materials Depth | Typical Boundary Crossings | Integration Design Implication |
|---|---|---|
| Single level | Zero, typically resolved within one ERP | No FTP interface required |
| Two to three levels | One to two, typically plant-to-plant or plant-to-supplier | Point-to-point or simple hub-and-spoke sufficient |
| Four or more levels | Three or more, spanning multiple plants and suppliers | Hub-and-spoke with disciplined schema governance recommended |

**Table 3.** *Relationship between bill of materials depth and the number of ERP boundaries dependent demand typically crosses.*

## 04 FTP as an Integration Mechanism

### › 4.1 FTP Protocol Characteristics

| Characteristic | Description |
|---|---|

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| | |
|---|---|
| Transport model | Client-server file upload and download over a dedicated control and data connection |
| Statefulness | Session-based; a file transfer is a discrete, completed operation |
| Native security | None in plain FTP; requires FTPS or SFTP for encryption |
| Typical payload | Batch files, often delimited text, fixed-width, or XML |
| Failure visibility | Explicit transfer success or failure per file, well suited to batch reconciliation |

**Table 4.** *Core characteristics of FTP as a file transport mechanism for ERP-to-ERP integration.*

### › 4.2 FTP Versus Modern Integration Alternatives

| Attribute | FTP / SFTP Batch | REST API | Event Streaming |
|---|---|---|---|
| Typical latency | Minutes to hours | Sub-second to seconds | Sub-second |
| Implementation complexity | Low | Moderate | High |
| Legacy system compatibility | High | Moderate, requires API layer | Low without middleware |
| Natural audit artifact | Yes, the file itself | Requires separate logging | Requires separate logging |
| Best-fit demand signal type | Dependent demand, batch MRP output | Time-sensitive transactional data | High-frequency operational events |

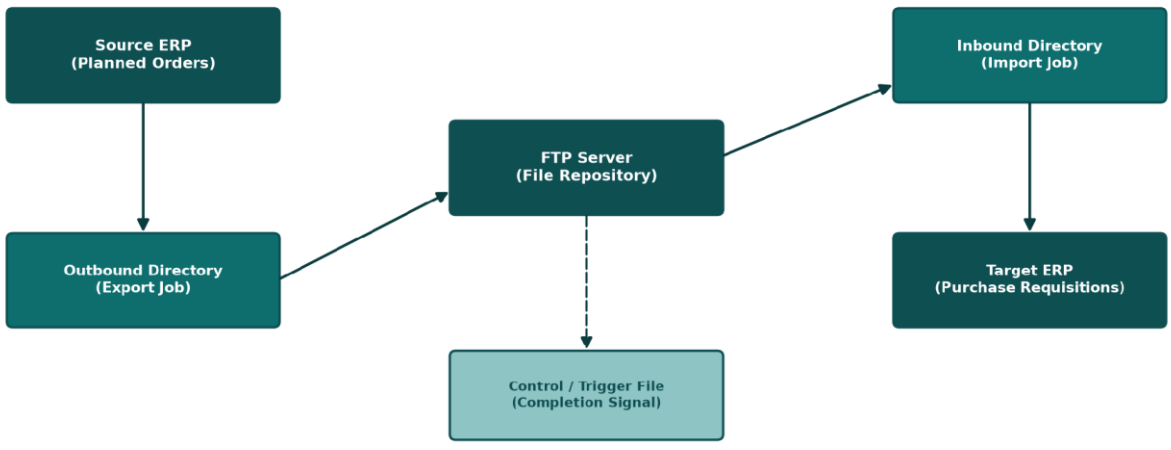
**Table 5.** *Comparison of FTP batch exchange against modern API and event-streaming integration alternatives.*

### › 4.3 Historical Context: Why FTP Became the Default

FTP's role in ERP-to-ERP integration dates to an era in which batch processing windows, rather than continuous availability, defined most enterprise system operation. MRP runs themselves were, and in many organizations remain, scheduled batch jobs rather than continuous calculations, which means the dependent demand output they produce is naturally batch-shaped data well suited to batch file transport. FTP's standardization in the early days of enterprise networking, combined with its straightforward client-server model, made it the default choice for connecting systems that were otherwise built by different vendors on different platforms with little else in common. That historical fit has proven durable specifically because dependent demand data has not fundamentally changed its batch-oriented nature even as the surrounding technology landscape has moved toward continuous, event-driven architectures for other use cases.

## 05 Reference Architecture for FTP-Based Demand Flow

A well-formed FTP-based dependent demand flow architecture separates the source ERP export process, the file transport layer, and the target ERP import process into distinct, independently governed stages, illustrated in Figure 2.



iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

**Figure 2.** *Reference architecture for FTP-based dependent demand flow, separating source export, file transport, control signaling, and target import stages.*

### › 5.1 Reference Architecture Layers

| Layer | Responsibility |
|---|---|
| Export layer | Generates the outbound file from source ERP MRP output on a defined schedule |
| Transport layer | Moves the file reliably between source and target file directories |
| Control layer | Signals file completeness and transfer success independently of the data file itself |
| Import layer | Validates and loads the received file into the target ERP |

**Table 6.** *Reference architecture layers for FTP-based dependent demand flow.*

## 06 FTP Integration Patterns

The three patterns below are ordered roughly by increasing architectural investment and decreasing per-connection governance burden, and most mature multi-system landscapes eventually adopt some combination of all three across different parts of their integration footprint.

### › 6.1 Pattern 1: Point-to-Point Batch Exchange

Point-to-point batch exchange connects exactly one source ERP directly to exactly one target ERP through a dedicated FTP session and file directory structure. It is the simplest pattern to implement and reason about, but it scales poorly once more than a handful of ERP systems must exchange demand signals with one another.

### › 6.2 Pattern 2: Hub-and-Spoke File Broker

A hub-and-spoke pattern routes all file exchange through a central FTP broker, illustrated in Figure 3, which every participating ERP system connects to rather than connecting directly to every other system. This pattern trades a small amount of additional infrastructure for substantially simpler connectivity management as the number of participating systems grows.

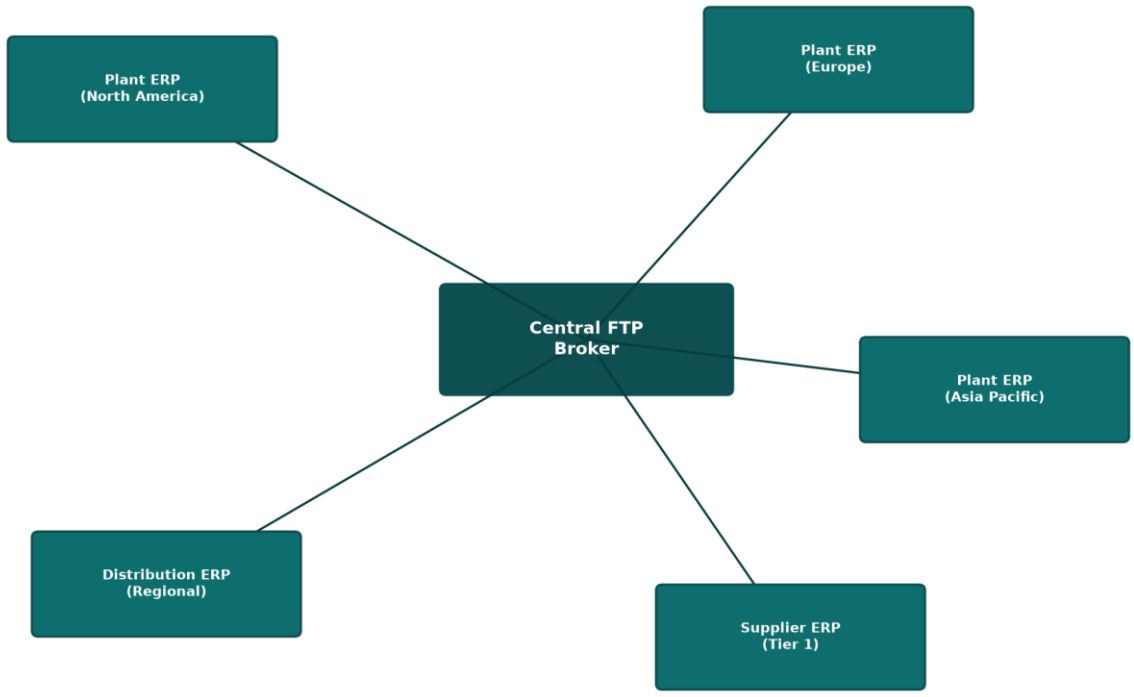



**Figure 3.** *Hub-and-spoke FTP broker pattern connecting multiple plant, supplier, and distribution ERP systems through a single central broker.*

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

### › 6.3 Pattern 3: Scheduled Polling with Control Files

Scheduled polling combines either of the two connectivity patterns above with an explicit control file convention: the target system polls for a control file signaling that the corresponding data file is complete and ready for import, rather than attempting to import a data file the moment it appears, which risks reading a partially written file.

### › 6.4 Pattern Comparison

| Attribute | Point-to-Point | Hub-and-Spoke | Scheduled Polling |
|---|---|---|---|
| Connectivity complexity | Grows quadratically with system count | Grows linearly with system count | Depends on underlying connectivity pattern |
| Partial file read risk | Present unless mitigated | Present unless mitigated | Mitigated by design |
| Central point of failure | None | The broker itself | Depends on underlying pattern |
| Best fit | Two or three systems, simple scope | Four or more participating systems | Any pattern requiring transfer completeness guarantees |

**Table 7.** *Comparison of the three principal FTP integration patterns for dependent demand flow.*

### › 6.5 Hybrid Pattern: Broker with Direct Fallback

A hybrid variant, seen in organizations with a small number of especially latency-sensitive relationships alongside a larger population of routine ones, routes most traffic through the central broker while maintaining a small number of direct, point-to-point connections for the highest-priority signals. This avoids adding broker hop latency to the relationships that can least tolerate it, while still capturing the governance benefit of the broker for the majority of traffic.

| Traffic Category | Routing Approach | Rationale |
|---|---|---|
| Routine dependent demand, most plants and suppliers | Through central broker | Simplifies governance and connectivity management |
| High-priority, latency-sensitive components | Direct point-to-point | Avoids broker hop latency for the most time-critical signals |

**Table 8.** *Traffic routing approach under the hybrid broker-with-fallback pattern.*

### › 6.6 A Worked Example: Scaling from Two Plants to a Regional Network

Consider a manufacturer that begins with a simple point-to-point FTP interface connecting two plants, one producing a subassembly consumed by the other. The interface works reliably for over a year, reinforcing the reasonable assumption that FTP-based batch exchange is a solved problem requiring no further architectural attention. As the manufacturer commissions a third plant, then acquires a fourth through a regional consolidation, each new plant relationship is connected the same way: a new dedicated point-to-point FTP interface, configured by whichever integration analyst is available at the time, each with its own slightly different file naming convention and control file approach inherited from whoever built the original interface they most recently reviewed.

By the time a sixth plant joins the network, the organization is operating fifteen distinct point-to-point interfaces, each independently configured, none sharing a common schema versioning approach, and no single team holding visibility into the complete connection map. A schema change needed at one plant requires manually identifying every other plant it exchanges data with, a process that on one occasion missed a connection entirely, resulting in a silent data quality issue that took several weeks to trace back to its root cause. This is precisely the point at which the connection count mathematics in Section 15 stops being a theoretical concern and becomes an operational one: fifteen point-to-point interfaces very likely represent only a fraction of the thirty-plus connections a fully connected six-plant network would require if every plant needed to exchange demand signals with every other plant, and the governance burden of tracking them individually was already unsustainable well before reaching that theoretical maximum.

The organization's eventual response, consolidating around a central FTP broker per Figure 3 and retiring the point-to-point interfaces over a two-quarter migration, reduced the connection count from fifteen to six while also, for the first time, giving a single team complete visibility into every active dependent demand interface across the network. The migration cost real effort, but considerably less than continuing to add point-to-point interfaces at the rate the regional network was growing would have cost over the following two years.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

## 07 File Format and Structure Design

### › 7.1 Common File Formats

| Format | Advantage | Trade-Off |
|---|---|---|
| Fixed-width text | Compact, fast to parse, long-established in legacy ERP interfaces | Brittle if field lengths change |
| Delimited text (CSV-style) | Simple to generate and parse, human-readable | Ambiguous if delimiter characters appear in data |
| XML | Self-describing, supports nested structure | Larger file size, more parsing overhead |
| Fixed-schema flat file with header record | Combines compactness with self-description via a header | Requires disciplined schema versioning |

**Table 9.** *Common file formats used for FTP-based dependent demand data exchange.*

### › 7.2 Control File and Trigger File Patterns

| Pattern | Mechanism |
|---|---|
| Zero-byte flag file | An empty file whose mere presence signals data file completeness |
| Checksum control file | A small file containing a checksum or record count for the data file |
| Renamed extension convention | The data file is written under a temporary extension and renamed only once complete |

**Table 10.** *Control and trigger file patterns used to guarantee data file completeness before import.*

### › 7.3 Field-Level Design Considerations

| Field Category | Design Consideration |
|---|---|
| Key identifiers | Confirm consistent identifier formats across source and target, or apply mapping per companion cross-reference guidance |
| Quantity and unit of measure | Always pair a quantity field with an explicit unit of measure field, never assume a default |
| Date and time fields | Standardize on a single time zone and date format across every file interface |
| Optional versus mandatory fields | Document every field's optionality explicitly in the schema, since silent omission is a common source of import failure |

**Table 11.** *Field-level design considerations for dependent demand file interfaces.*

## 08 Scheduling and Timing Considerations

### › 8.1 Batch Window Design

| Consideration | Guidance |
|---|---|
| Source MRP run completion | Schedule export after source MRP run completion, not on a fixed clock time alone |
| Target MRP run alignment | Schedule target import to complete before the target system's own MRP run |
| Overlap avoidance | Ensure export and import windows do not overlap with source or target system maintenance windows |
| Time zone handling | Standardize on a single reference time zone across all participating systems |

**Table 12.** *Batch window design considerations for FTP-based dependent demand flow scheduling.*

### › 8.2 Latency Versus Batch Frequency

Figure 4 illustrates how median demand signal latency scales with batch frequency across a range of common scheduling intervals, based on illustrative benchmark data compiled from published integration practice.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

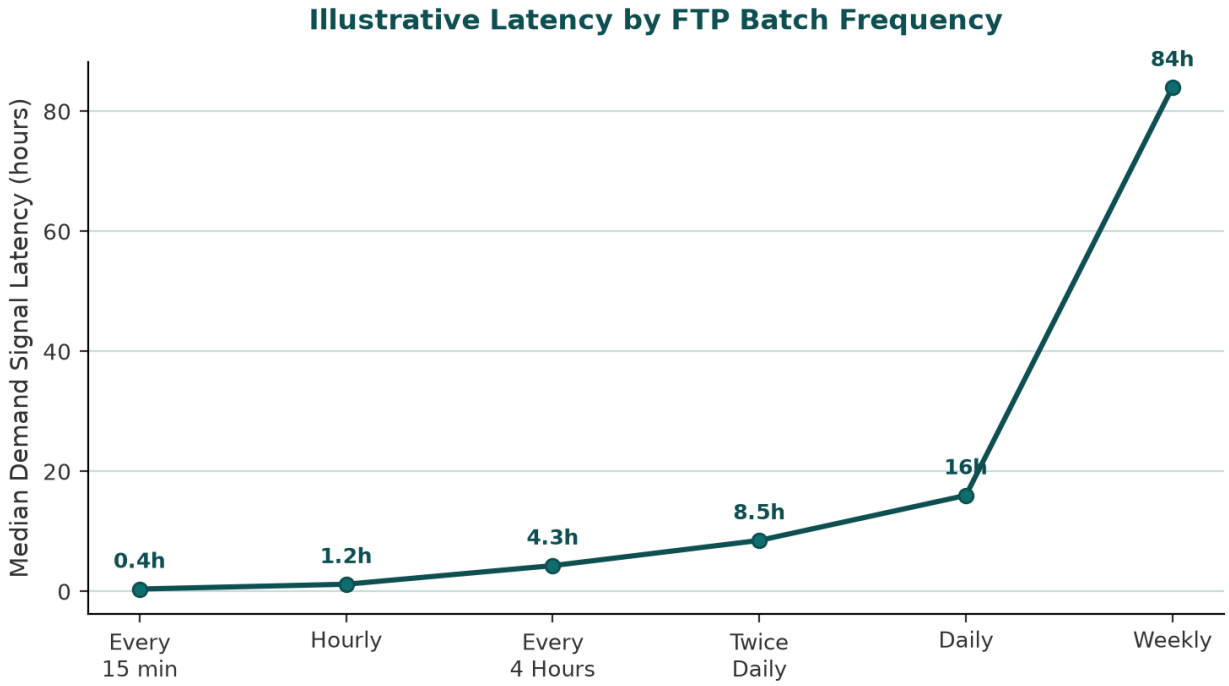



**Figure 4.** *Illustrative median demand signal latency by FTP batch frequency, showing a sharp increase in latency as batch frequency decreases.*

| Batch Frequency | Median Latency | Typical Use Case |
|---|---|---|
| Every 15 minutes | 0.4 hours | High-value, high-volatility component demand |
| Hourly | 1.2 hours | Standard production component demand |
| Every 4 hours | 4.3 hours | Moderate-priority component demand |
| Twice daily | 8.5 hours | Lower-priority or low-volatility demand |
| Daily | 16.0 hours | Routine replenishment demand |
| Weekly | 84.0 hours | Long-lead-time or strategic sourcing demand |

**Table 13.** *Illustrative median demand signal latency by batch frequency, with typical corresponding use case.*

## 09 Dependent Demand Propagation Flow

Figure 5 traces the full propagation path of a dependent demand signal from the source ERP's MRP run through to a purchase requisition in the target ERP, including the feedback loop by which the target system's own downstream demand can inform the next cycle of the source ERP's planning run.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

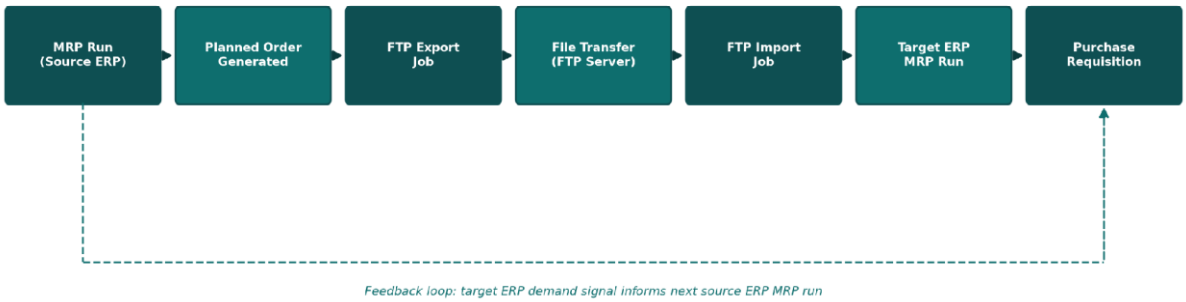



**Figure 5.** *End-to-end dependent demand propagation flow from source ERP MRP run through FTP export, transfer, and import, to target ERP purchase requisition, with a feedback loop informing the next planning cycle.*

### › 9.1 Propagation Steps

1. Source ERP executes its scheduled MRP run and generates planned orders for cross-boundary components.
2. An export job extracts qualifying planned orders into the agreed file format.
3. The export job writes the data file, and subsequently a control file, to the outbound directory.
4. The FTP transport layer moves both files to the target system's inbound directory.
5. The target import job detects the control file and validates the data file before loading.
6. Validated records are loaded into the target ERP, typically as purchase requisitions or planned orders.
7. The target ERP's own MRP run consumes the loaded demand alongside its other planning inputs.

### › 9.2 Propagation Timing Reference

| Stage | Typical Duration |
|---|---|
| Source MRP run to export job start | Under 15 minutes with event-triggered scheduling |
| Export job execution | A few minutes for typical dependent demand volumes |
| File transfer over FTP | Under 1 minute for typical file sizes on a stable connection |
| Import job execution and validation | A few minutes, longer if extensive validation is applied |

**Table 14.** *Typical stage-level duration reference for the dependent demand propagation flow.*

### › 9.3 Handling Multi-Level Explosion Across Boundaries

When dependent demand must cross more than one ERP boundary in sequence, as described in Section 3.3 for deep bills of materials, the propagation flow in Figure 5 effectively repeats: the target ERP's own MRP run generates a further planned order that itself becomes an export candidate for a second downstream boundary crossing. This chaining introduces cumulative latency, since each hop adds its own batch window delay, and reinforces why the latency tier classification introduced in Section 8.2 should account for the total number of boundary crossings a given demand signal must traverse, not only the latency of a single hop in isolation.

A practical consequence is that the same absolute batch frequency can represent very different actual risk depending on chain depth. A daily batch frequency across a single boundary crossing implies roughly sixteen hours of latency, which is often acceptable for routine replenishment demand. The same daily batch frequency repeated across three sequential boundary crossings compounds to roughly two full days of cumulative latency before the signal reaches its final destination, which may no longer be acceptable for the same underlying component if it sits on a constrained production schedule. Latency tier classification should therefore be applied to the full end-to-end chain a signal traverses, not evaluated independently at each individual hop.

## 10 Error Handling and Reconciliation

### › 10.1 Common Failure Modes

| Failure Mode | Typical Cause | Detection Mechanism |
|---|---|---|
| Partial file read | Import job reads a data file before writing is complete | Control file pattern per Section 7.2 |

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| | | |
|---|---|---|
| Duplicate file processing | Import job reprocesses an already-loaded file after a retry | File archival and processed-file tracking |
| Schema drift | Source system field layout changes without corresponding target update | Header record or schema version validation |
| Silent transfer failure | Network interruption during FTP transfer leaves an incomplete file | File size or checksum validation against the control file |

**Table 15.** *Common FTP-based integration failure modes, causes, and detection mechanisms.*

### › 10.2 Reconciliation Checks

- Confirm every exported record from the source ERP has a corresponding loaded record in the target ERP.
- Confirm file-level record counts match between the export job log and the import job log.
- Confirm no data file is processed more than once by checking against a processed-file log.
- Confirm control files and data files are always paired, with no orphaned file of either type.

## 11 Security Considerations for FTP

### › 11.1 FTP Versus FTPS Versus SFTP

| Variant | Encryption | Typical Port | Recommendation |
|---|---|---|---|
| Plain FTP | None | 21 | Not recommended for production dependent demand data |
| FTPS (FTP over TLS) | TLS-encrypted control and data channel | 990 (implicit) or 21 (explicit) | Acceptable where SFTP is unavailable |
| SFTP (SSH File Transfer Protocol) | SSH-encrypted single channel | 22 | Preferred for new implementations |

**Table 16.** *Comparison of plain FTP, FTPS, and SFTP for securing dependent demand file exchange.*

### › 11.2 Security Controls

- Use SFTP or FTPS rather than plain FTP for any production dependent demand data exchange.
- Restrict file directory access to the specific technical accounts used by the export and import jobs.
- Rotate credentials for FTP technical accounts on a defined schedule.
- Log every file transfer attempt, successful or failed, for audit and troubleshooting purposes.

### › 11.3 Compliance Considerations

| Consideration | Guidance |
|---|---|
| Data residency | Confirm FTP server or broker location complies with applicable data residency requirements |
| Retention policy | Align file archival retention with the organization's records retention policy |
| Access audit | Maintain a reviewable log of which technical accounts accessed which file directories |
| Third-party supplier connections | Confirm supplier-side FTP security posture meets the same standard applied internally |

**Table 17.** *Compliance considerations for FTP-based dependent demand file exchange.*

## 12 Performance Benchmarking

Figure 6 presents illustrative throughput benchmarks across the three integration patterns defined in Section 6, based on published implementation guidance current as of early 2023.

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

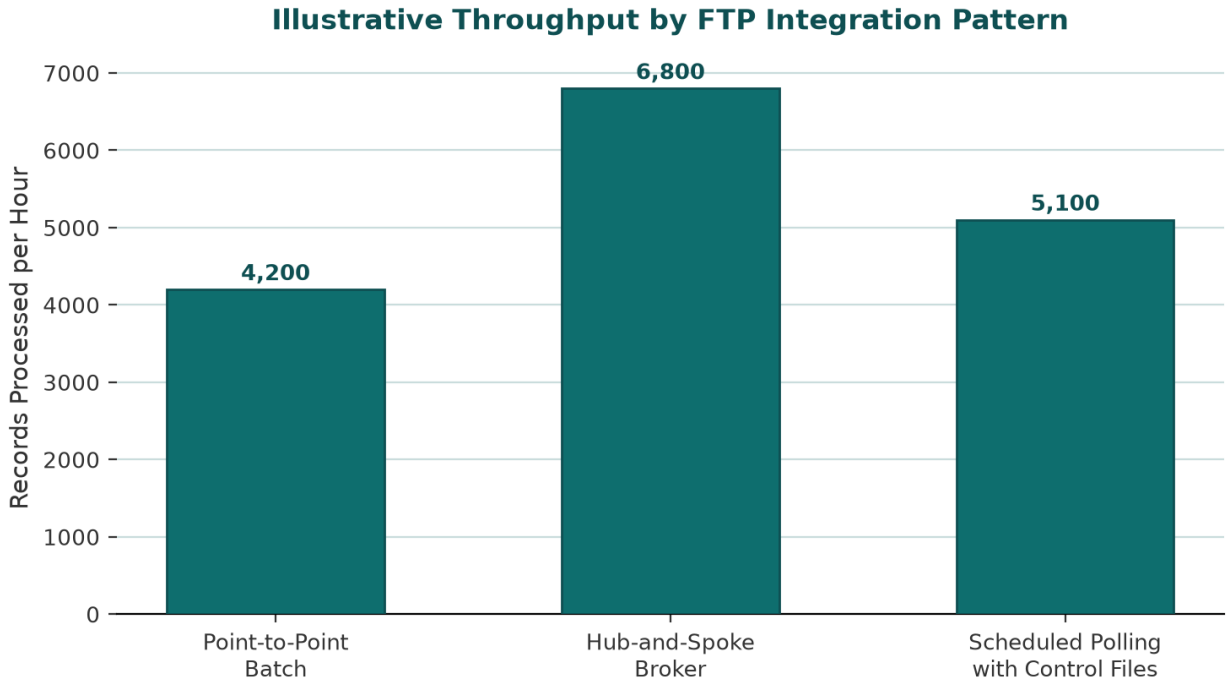



**Figure 6.** *Illustrative throughput comparison across the three FTP integration patterns, showing the hub-and-spoke broker pattern achieving the highest sustained throughput.*

| Pattern | Records per Hour | Relative Throughput |
|---|---|---|
| Point-to-Point Batch | 4,200 | 1.0x baseline |
| Hub-and-Spoke Broker | 6,800 | 1.6x baseline |
| Scheduled Polling with Control Files | 5,100 | 1.2x baseline |

**Table 18.** *Illustrative throughput benchmark by FTP integration pattern.*

## 13 Monitoring and Alerting

| Metric | Purpose | Illustrative Alert Threshold |
|---|---|---|
| File transfer success rate | Detects transport-layer reliability issues | Below 99 percent over a rolling 24-hour window |
| Export-to-import elapsed time | Detects propagation delay beyond the expected batch window | Exceeds twice the scheduled batch interval |
| Reconciliation variance | Detects record count mismatches between source and target | Any non-zero variance |
| Control file absence | Detects a data file arriving without its paired control file | Any occurrence |

**Table 19.** *Core monitoring metrics for an FTP-based dependent demand flow architecture.*

### › 13.1 Sample Monitoring Dashboard Extract

| Interface | Last Transfer Status | Elapsed Time | Reconciliation Variance |
|---|---|---|---|
| Plant02 to Plant05 Components | Success | 6 minutes | 0 |
| Plant02 to Supplier-A Raw Materials | Success | 9 minutes | 0 |
| Plant07 to Distribution-Central | Success | 4 minutes | 0 |
| Plant05 to Supplier-B Packaging | Delayed | 48 minutes | 0 |

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

**Table 20.** *Sample daily monitoring dashboard extract for a set of active FTP-based dependent demand interfaces.*

## 14 Governance and Change Management

| Role | Responsibility |
|---|---|
| Interface Owner | Accountable for the overall correctness and availability of a given file interface |
| Source System Analyst | Responsible for export job configuration and source-side data quality |
| Target System Analyst | Responsible for import job configuration and target-side validation rules |
| Integration Operations Team | Responsible for FTP infrastructure, monitoring, and incident response |

**Table 21.** *Governance roles and responsibilities for FTP-based dependent demand flow interfaces.*

- Any change to file format or schema requires sign-off from both the source and target system analysts.
- Interface owners should maintain a current inventory of every active file interface and its governing schema version.
- Schema changes should be versioned and backward-compatible where feasible, to avoid a hard cutover across all participating systems simultaneously.

### › 14.1 Escalation Matrix

| Condition | Escalation Level | Notified Role |
|---|---|---|
| Reconciliation variance detected | Interface owner | Source and target system analysts |
| Sustained transfer failure beyond one batch cycle | Integration operations lead | Interface owner, operations team |
| Schema change proposed by only one side | Program governance | Both system analysts, interface owner |
| Recurring incident on the same interface within 30 days | Program governance | Integration operations lead, interface owner |

**Table 22.** *Escalation matrix for FTP-based dependent demand flow incidents and change proposals.*

## 15 Multi-Plant and Multi-ERP Considerations

Organizations operating more than a handful of plant, supplier, or distribution ERP systems consistently benefit from consolidating around the hub-and-spoke pattern illustrated in Figure 3, since the alternative, a full mesh of point-to-point connections, becomes difficult to govern once system count grows.

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

**Table 23.** *Connection count comparison between point-to-point and hub-and-spoke topologies as system count grows.*

- Standardize file format and control file convention centrally, even under a hub-and-spoke topology, so that adding a new participating system does not require a bespoke format.
- Assign the broker itself a clear operational owner, since it becomes a shared dependency across every participating system.

## 16 Migrating from FTP to Modern Integration

| Migration Driver | Recommended Target |
|---|---|
| Need for sub-minute latency on a specific demand signal | Event streaming or API-based integration for that signal only |
| Growing system count straining hub-and-spoke governance | Managed file transfer (MFT) platform with centralized monitoring |
| Regulatory requirement for stronger transport security | SFTP at minimum, or API integration with mutual TLS |
| No specific latency or governance pain point | Continue with the existing FTP-based pattern |

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

**Table 24.** *Migration drivers and recommended modern integration targets for FTP-based dependent demand flows.*

Migration should be driven by a specific, demonstrated pain point rather than by the general perception that FTP is outdated. Section 4.2 already demonstrates that FTP remains well matched to dependent demand signals that do not require sub-second latency, and wholesale migration without a clear driver frequently trades a well-understood, stable pattern for a more complex one without a commensurate benefit.

### › 16.1 Coexistence Strategy

| Coexistence Approach | Description |
|---|---|
| Signal-by-signal migration | Migrate only the specific demand signals with a demonstrated latency driver, leaving the rest on FTP |
| Parallel run period | Operate both the legacy FTP interface and the new integration in parallel before fully decommissioning FTP |
| Wrapper approach | Front an existing FTP interface with an API layer for consumers that need programmatic access without a full re-platform |

**Table 25.** *Coexistence approaches for organizations migrating selected dependent demand signals away from FTP.*

## 17 Implementation Methodology

8. Inventory dependent demand signals requiring cross-ERP propagation and classify each by required latency tier.
9. Select the integration pattern per Section 6 based on participating system count and latency requirements.
10. Design the file format and control file convention per Section 7.
11. Configure export, transport, and import jobs per the reference architecture in Section 5.
12. Implement security controls per Section 11 before any production data flows.
13. Establish monitoring and alerting per Section 13.
14. Execute reconciliation validation per Section 10 against a production-representative volume.
15. Document governance ownership per Section 14 before cutover.

| Phase | Illustrative Duration |
|---|---|
| Signal inventory and pattern selection | 1 to 2 weeks |
| File format and architecture design | 2 to 3 weeks |
| Job configuration and security setup | 2 to 4 weeks |
| Monitoring, reconciliation, and validation | 1 to 2 weeks |

**Table 26.** *Illustrative phase timeline for implementing an FTP-based dependent demand flow interface.*

## 18 Risk Factors and Mitigation

| Risk Factor | Potential Impact | Mitigation |
|---|---|---|
| Missing control file convention | Partial file reads causing corrupted demand data | Adopt a control file pattern per Section 7.2 |
| Point-to-point sprawl | Unmanageable connection count as system count grows | Consolidate to hub-and-spoke per Section 15 |
| Plain FTP in production | Credential and data interception risk | Migrate to SFTP or FTPS per Section 11 |
| No reconciliation process | Undetected data loss between source and target | Implement checks per Section 10.2 |
| Undocumented schema ownership | Uncoordinated breaking changes to file format | Assign interface ownership per Section 14 |

**Table 27.** *Common risk factors in FTP-based dependent demand flow architectures, with mitigations.*

## 19 Cost Considerations

Cost drivers for an FTP-based dependent demand flow architecture are generally modest relative to modern integration alternatives, which is one of the pattern's enduring advantages, but they are not zero, and each category below should be accounted for explicitly in a program budget.

| Cost Category | Primary Driver | Illustrative Relative Weight |
|---|---|---|

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

| | | |
|---|---|---|
| FTP or SFTP server infrastructure | Server sizing and redundancy | Low |
| Managed file transfer platform licensing (if adopted) | Number of participating systems and file volume | Moderate |
| Export and import job development | Number of distinct file interfaces | Moderate to high, one-time |
| Ongoing monitoring and operations | Number of active interfaces and incident volume | Low to moderate, ongoing |

**Table 28.** *Illustrative relative cost weight by category for an FTP-based dependent demand flow architecture.*

## 20 Industry Adoption Patterns

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

| Industry Pattern | Dominant FTP Pattern | Typical Latency Tier |
|---|---|---|
| Automotive and discrete manufacturing | Hub-and-spoke across plant ERPs | Hourly to every 4 hours |
| Process manufacturing and chemicals | Point-to-point with key suppliers | Daily |
| Retail and consumer distribution | Scheduled polling with distribution centers | Twice daily to daily |

**Table 29.** *Cross-industry summary of dominant FTP integration pattern and typical latency tier.*

Across all three patterns, the underlying decision logic remains the classification-first approach established in Section 8: the specific batch frequency and topology vary by industry and by component criticality, but the discipline of classifying before designing holds consistently.

## 21 Common Pitfalls

Most of the pitfalls below trace back to treating an individual FTP interface as a self-contained, one-time integration task rather than as one component of a shared, evolving architecture.

- Deploying plain, unencrypted FTP in production for dependent demand data.
- Omitting a control file convention and relying on file arrival timing alone.
- Allowing point-to-point connections to accumulate without ever consolidating to a hub-and-spoke topology.
- Treating schema changes as a source-system-only decision without target-side sign-off.
- Migrating to a modern integration technology without a specific, demonstrated latency or governance driver.

## 22 Best Practices and Recommendations

- Classify every dependent demand signal by required latency tier before selecting an integration pattern.
- Adopt SFTP as the default transport, reserving plain FTP for no production scenario.
- Always pair a data file with an explicit control file, per the patterns in Section 7.2.
- Consolidate to a hub-and-spoke topology once participating system count exceeds three or four.
- Implement reconciliation checks as a standard part of every interface, not as an optional addition.
- Assign clear interface ownership before go-live, covering both source and target sides.
- Migrate away from FTP only in response to a specific, demonstrated latency or governance driver.

## 23 Future Outlook

As of early 2023, FTP-based batch exchange continues to coexist with, rather than being displaced by, modern API and event-streaming integration, particularly for dependent demand signals where near real-time latency provides limited additional business value relative to its implementation cost. Continued adoption of managed file transfer platforms, offering centralized monitoring and governance on top of familiar FTP-family transport, is expected as organizations seek to retain the simplicity of file-based exchange while addressing the governance challenges of growing system count.

## 24 Conclusion

FTP-based integration remains a practical, well-understood mechanism for propagating dependent demand signals across ERP boundaries, provided it is implemented with the same architectural discipline expected of any other integration technology: a clear reference architecture, an explicit control file convention, appropriate transport security, reconciliation validation, and clear governance ownership.

Organizations that apply the patterns, benchmarks, and reference tables presented in this article are better positioned to operate FTP-based dependent demand flows reliably at scale, and to make a deliberate, evidence-

iJETRM
INTERNATIONAL JOURNAL OF ENGINEERING TECHNOLOGY RESEARCH & MANAGEMENT (IJETRM)
https://ijetrm.com/

based decision about when, and whether, migration to a more modern integration technology is actually warranted for a given demand signal.

The worked example in Section 6.5 illustrates a pattern common to many of the architectural disciplines discussed throughout this research series: a simple, reasonable starting point scales poorly without deliberate structure, not because the original decision was wrong, but because no one revisited it as conditions changed. Treating FTP-based dependent demand integration as a governed architecture from the outset, rather than a collection of individually reasonable point solutions, is the single highest-leverage recommendation this article offers.

## REFERENCES

1) APICS (Association for Supply Chain Management). (2019). CPIM Certification Study Guide: Materials Requirements Planning. APICS.
2) Chopra, S., & Meindl, P. (2019). Supply Chain Management: Strategy, Planning, and Operation (7th ed.). Pearson.
3) Fette, I., & Melnikov, A. (2011). RFC 6455: The WebSocket Protocol. Internet Engineering Task Force.
4) Postel, J., & Reynolds, J. (1985). RFC 959: File Transfer Protocol (FTP). Internet Engineering Task Force.
5) Silberschatz, A., Galvin, P. B., & Gagne, G. (2018). Operating System Concepts (10th ed.). Wiley.
6) Vernon, M. (2013). Managed File Transfer: A Practical Guide to Secure Data Exchange. IT Governance Publishing.
7) Vollmann, T. E., Berry, W. L., Whybark, D. C., & Jacobs, F. R. (2004). Manufacturing Planning and Control for Supply Chain Management (5th ed.). McGraw-Hill.
8) Ylonen, T., & Lonvick, C. (2006). RFC 4253: The Secure Shell (SSH) Transport Layer Protocol. Internet Engineering Task Force.

## Appendix A: Glossary of Key Terms

| Term | Definition |
|---|---|
| Dependent Demand | Demand for a component or material calculated from a parent item's bill of materials. |
| MRP | Material Requirements Planning, the calculation process that derives dependent demand. |
| FTP | File Transfer Protocol, a standard protocol for transferring files between systems. |
| SFTP | SSH File Transfer Protocol, a secure file transfer protocol using SSH encryption. |
| Control File | A companion file signaling that a corresponding data file is complete and ready for processing. |
| Hub-and-Spoke | An integration topology routing all exchange through a central broker rather than direct connections. |
| Planned Order | An MRP-generated proposal to produce or procure a quantity of an item by a given date. |
| Managed File Transfer (MFT) | A platform providing centralized governance, monitoring, and security for file-based integration. |

**Table 30.** Glossary of key terms referenced throughout this article.

## Appendix B: Sample Control File Structure

The sample structure below illustrates a minimal checksum-style control file, per the pattern described in Section 7.2, paired with a dependent demand data file.

| Field | Example Value | Purpose |
|---|---|---|
| Data file name | DEMAND_PLANT02_20230214.dat | Identifies the corresponding data file |
| Record count | 1,482 | Confirms the expected number of records in the data file |
| Checksum | 8f3a1c9e2b7d4056 | Confirms data file integrity after transfer |

| | | |
|---|---|---|
| Generation timestamp | 2023-02-14T22:05:11 | Confirms when the source export job produced the file |
| Schema version | v2.3 | Confirms which file layout version the data file follows |

**Table 31.** *Sample control file field structure for a dependent demand data file.*