MUZULU

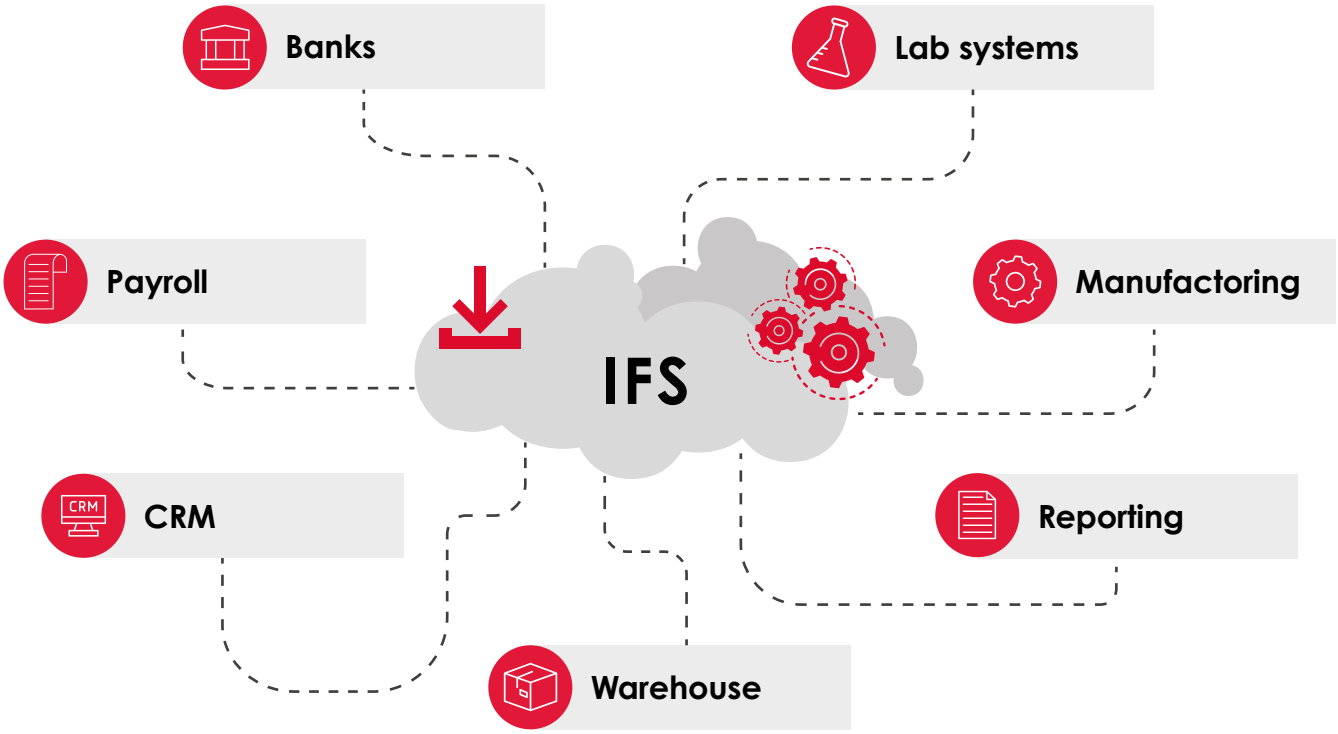
A Brief Practical Guide to IFS APIs & Integrations

How to choose the right approach for connected, validated and successful IFS environments

IFS integration is not simply a technical issue.

Any technology tasked with driving and supporting a business should be viewed as a tool that requires proper tuning towards their goals, aims and development. This is critically important for an ERP with the responsibility to connect all the major functions of organisations where teams are consistently working with the same information.

IFS Cloud is one of the strongest options for organisations that need complex ERP capability, with rich operational functionality for both service-led and product-centric businesses, as well as embedded Enterprise Asset Management and Field Service Management. Its value increases through reliable connections and integrations with the wider systems that a business depends on such as payroll, banking, warehouse, laboratory, reporting and customer facing applications.



Prioritise Process Over Technology

The right approach for IFS APIs and integrations begins with a comprehensive understanding of the following core elements. Crucially, this involves thinking about:

- What business process is being supported?
- What external system is it connecting to?
- What is the direction of data flow?
- What are the timing requirements?
- Are there any governance issues?
- Are there any validation needs
- What is the long-term support capability?
- What happens if the integration fails?
- Will the integration need to scale further down the line?

Why IFS integration Matters

IFS works at the heart of daily company operations. As data moves between the platform and related external systems, the integration methods used play a big part in its overall success. Without getting this right, there can be negative effects on reliability, monitoring, support and upgrade resilience.

All of which easily erodes user confidence.

Older processes would usually combat this through a reliance on manual entry.

Regular file uploads, with teams constantly moving data from one place to another, made this incredibly resource heavy, open to error and would often fail to give up-to-date information on a global level across the company.

Most IFS users expect to see more automated processes to attend to these requirements. Alongside the obvious benefits in terms of time taken, a well-integrated and connected IFS solution also provides much more reliable, and better governed, data movement.

Old Manual Pattern	Integrated IFS pattern
Rekey data into IFS	Data flows into IFS through a governed integration
Upload files manually	Scheduled or monitored file processing
Email spreadsheets between teams	Controlled exchange between systems
Finding errors after sharing	Monitoring, error handling and clear data ownership

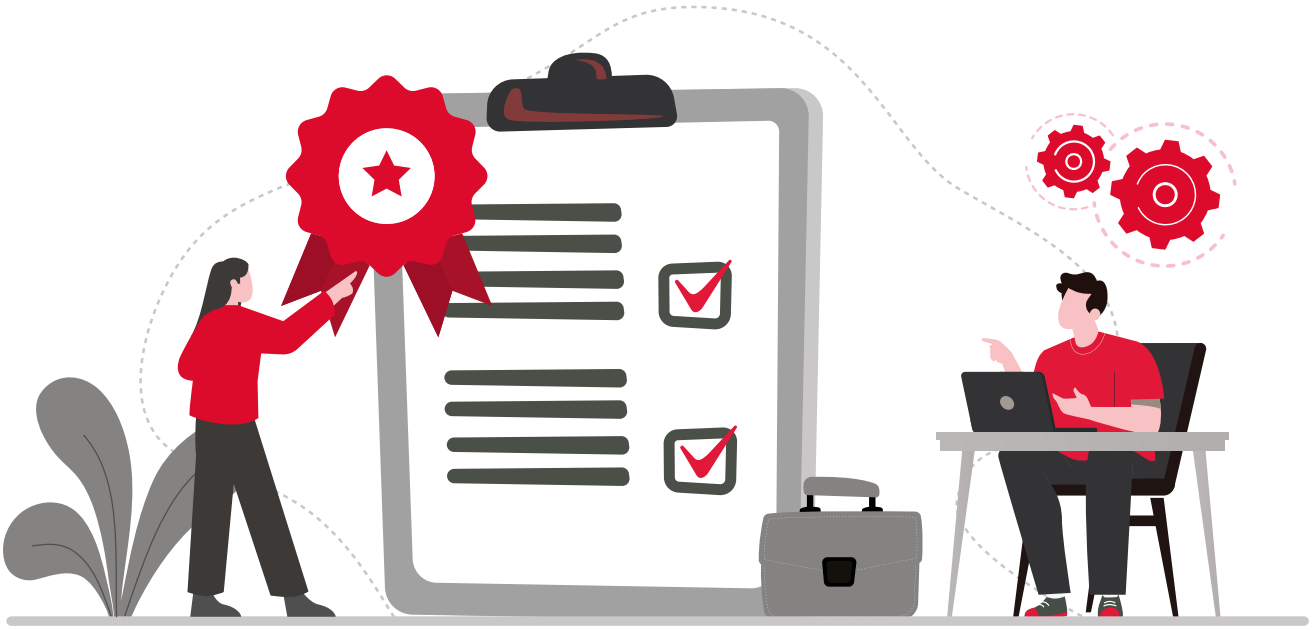
The Language of IFS Integration

IFS integration discussions can quickly become technical.

However, it is likely that conversations around the appropriate solution, and its component parts, need to be shared and understood right across the business.

Here is our simple glossary as a useful reference for the whole process.

Old Manual Pattern	Integrated IFS pattern
REST API	A structured way for systems to exchange data through web services. Often used for modern, transactional integrations.
Event Action	One way for IFS to trigger an outbound action when something happens, such as sending key data to another system or middleware.
Workflow	Another option for IFS to trigger an outbound action when something happens, such as sending key data to another system or middleware.
IFS Connect	An IFS framework used for routing, transforming, sending and receiving messages, including files and service calls.
Middleware / Enterprise Service Bus (ESB)	A platform that sits between systems to manage routing, mapping, transformation and orchestration.
Migration Job	A scheduled way to read or write file-based data. This is useful for batch activity but can be increasingly difficult with Cloud based solutions (especially if the data volumes are large).
External File	A structured file-handling approach used for specific file-based business processes (e.g. import a bank statement or export a tax report).
FTP / SFTP	Methods for transferring files between systems. This is commonly used where APIs are not available or not needed.
SOAP / XML	Older or alternative integration formats that may still exist in legacy or third-party systems.



A Guide to Help You Decide

IFS Cloud can be integrated with other systems in several ways and there is no universally accepted or correct way of doing so. Developing the answers gained from your initial understanding encourages a clear approach to determine which integration route or API choice to consider.

The question should never simply be “Can IFS connect to this system?”

Instead, a more nuanced approach is always required to find out the best course to take.

The Muzulu View

Good IFS integration decisions usually pass four tests:

- Does the integration pattern fit the process?
- Does the integration pattern fit the external system?
- Can it be governed?
- Will it remain manageable after future change?

Start with the Integration Scenario

Before choosing a method, define the integration scenario at work.

This will increase the chances of choosing the right integration technology from the outset.

Question:	What the answer tells you:
Is data going into IFS or coming out of IFS?	Whether the integration is Inbound or Outbound
Is this a live transaction or a scheduled batch?	Whether real-time, event-driven or file-based processing is required
Can the other system use REST APIs?	Whether a direct API approach is realistic
Is middleware available?	Whether orchestration can sit outside IFS
Does the data need to change on the way through?	Whether data transformation or mapping is required
Is the process regulated or validated?	Whether documentation, repeatability and regression testing must be considered early
What happens if it fails?	What monitoring, retry and support model is needed

IFS Integration Options

Once the working scenario is defined and documented, integration options can be properly considered and assessed. Each can be mapped to your process and scenario to find the best fit possible. Always remain aware of the considerations and restrictions of every option when carrying out any initial scoping.

IFS Integration Option	Best Suited To	Considerations
REST API	Modern systems that need structured data exchange	Needs security, monitoring and error handling
Event Action	IFS needs to notify another system when something happens	Keep messages focused and plan for failures
IFS Connect	Routing, transforming, sending or receiving messages and files	Needs clear configuration, monitoring and ownership
Middleware	Several systems need to connect or data needs orchestration	Useful, but should not become a black box
Migration Jobs	Scheduled batch loads or exports	Useful for batch activity, but not every integration
External Files	Structured file-based business processes	Needs careful setup and governance
File / FTP / SFTP	Banks, payroll providers or legacy systems that use files	Batch timing, reconciliation and failed-file handling must be clear
Low-code / no-code approaches	Regulated, repeatable or validation-heavy processes	Can reduce bespoke development and make testing, documentation and change easier to manage

Note: Low-code / no-code has been placed last because it is best understood as a governance and delivery approach, not simply another technical route.

Driving Decisions

A practical IFS integration decision often follows a simple pattern. Conducting thorough research and understanding of what the external system is capable of can act as a primary guide for the integration route to follow.

The Muzulu View

- This is not a hierarchy where one method is always best
- APIs are not automatically the right choice for every IFS integration
- The most successful choice depends on a fully mapped business scenario

Middleware or a service bus is available

Use REST APIs where suitable and handle orchestration in middleware

The external system can make REST web service calls

Use REST-based integration where appropriate

The external system only supports SOAP or XML

Use a gateway, routing or transformation layer to connect into REST-based services

Data is received as files

Use file polling, transformation and routing into the appropriate IFS process

Data is sent as files

Use file, FTP or SFTP-based outbound methods where appropriate

The requirement is a scheduled batch load or export

Consider batch-oriented approaches such as Migration Jobs or External Files

The outbound process is event-driven

Consider Event Actions and outbound integration patterns

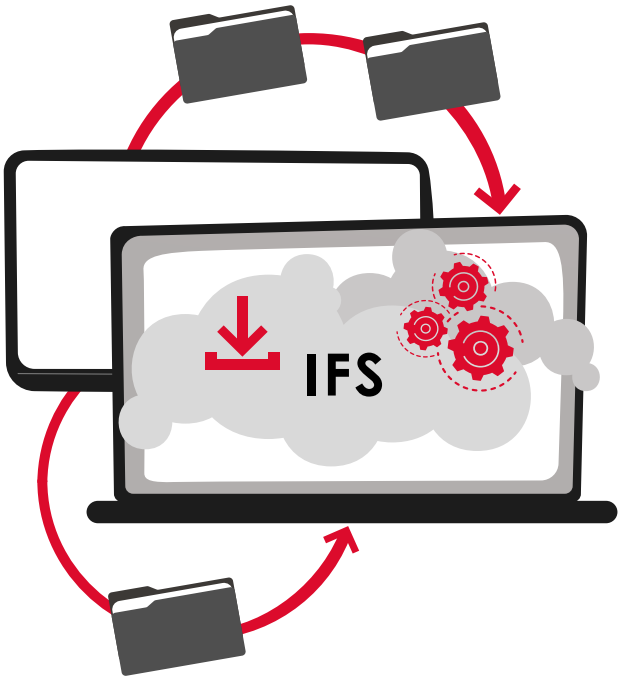
Regulated environments or repeatable processes

Low-code/no-code tools

Middleware and IFS integration

Middleware is particularly useful where IFS needs to exchange data with several systems, or where it is necessary for routing, mapping and orchestration outside the ERP system itself.

As an example, it can be most valuable when different systems speak different languages.



One system may use REST

Use REST APIs for direct, real-time integration with IFS Cloud.

Another may send files

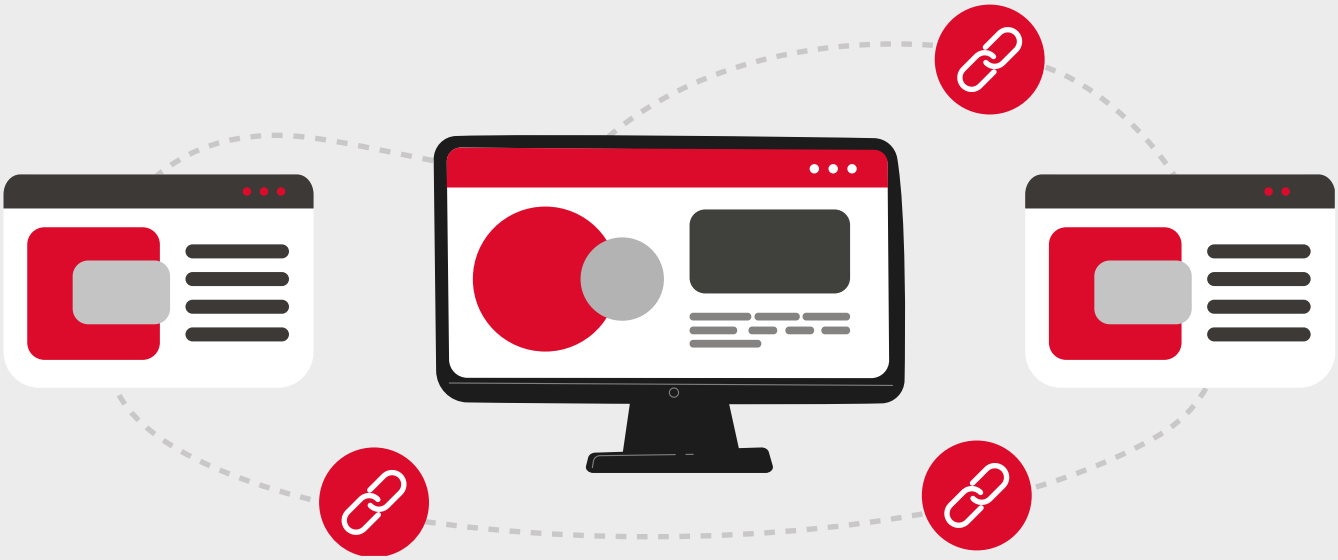
Use file, FTP or SFTP methods to exchange data with IFS cloud.

Another may require XML

Use a gateway or transformation layer to conect into IFS Cloud.

Another may need data transformed before it can use it

Use transformation and mapping to ensure data is in the right format before use in IFS Cloud.



Middleware can help make these differences manageable in several ways:

- Reduce point-to-point integration sprawl
- Keep orchestration outside the core ERP layer
- Standardise patterns across systems
- Improve monitoring and support
- Make integration change easier to understand

The Muzulu View

Middleware is most valuable when it creates clarity.

It should never become an undocumented black box between IFS and the rest of the business.

The Regulated Organisation Issue

For regulated organisations, IFS integrations are never simply technical connections. They may also become part of an entire validated operational process.

If a validated manufacturing, quality or laboratory process depends on data moving between IFS and another system, a change to the integration may require evidence that the wider end-to-end process still behaves as expected.

This is also where integration design has a direct effect on testing effort.

If the integration is heavily customised, difficult to understand or tightly connected to several processes, even a small change can force a wider testing exercise.

Low-code/No-code tools for integration can therefore be a reliable way of relieving the burden of custom development where validation is required.

Heavily Customised Integration	Low-Code/ No-Code Integration
Larger regression testing surface area	More focused regression testing
More bespoke code to assess	More visible integration logic
Harder impact analysis	Clearer change boundaries
Greater documentation burden	More repeatable documentation structure
Longer end-to-end testing cycles	More targeted testing effort
Higher upgrade uncertainty	Better upgrade resilience

The Muzulu View

The goal is not to avoid testing.

It is to make testing more focused, repeatable and proportionate to the change.

An Example for Life Sciences

A Life Sciences manufacturer employs a validated process where data moves between IFS and a laboratory system.

If that integration changes, the impact may not be limited to the interface itself. The organisation may need to show that:

- The right data still moves between systems
- IFS and the connected laboratory system still work as normal
- Exceptions are handled correctly
- The wider business process remains controlled

This is exactly where integration design and approach matters.

In a heavily customised or bespoke setup, even a small change can push teams towards broader end-to-end retesting. That can easily increase the effort required for validation, documentation, testing and regression analysis. It can also make upgrades more complex, because teams need to understand whether the change affects the wider validated process.

Low-code or no-code integration approaches may therefore demonstrate the most value for this scenario.

By reducing the amount of bespoke development involved, they can help make integration logic more visible, repeatable and easier to manage. That can support clearer documentation, more focused testing and better control when integrations need to change.

The important questions here are not just concerned with whether IFS can exchange data with another system. They need to know that integration can be understood, evidenced, tested and controlled over time.

- Can the integration be validated?
- Can the behaviour be documented?
- Can the process be tested repeatedly?
- Can changes be controlled?
- Can regression testing be focused on the areas genuinely affected?
- Can the integration withstand upgrades and patches?

With a more governed integration model, the boundaries of change are clearer. All of which can make it easier to assess impact, focus regression testing and maintain confidence that the wider process still works as intended.

The Muzulu View

The most resilient integration in these situations is often the one that makes change easier to understand, test, document and control.

Common IFS integration Mistakes to Avoid

The same practical problems often appear in IFS integration projects. The most important aspect in avoiding them is to make them a part of the IFS infrastructure for the business at the design stage. This allows integrations to be a more holistic process and not treated as a technical bolt-on at a later stage.

Common Mistake	Why it causes problems
Assuming APIs are always best	Some IFS scenarios are better suited to files, middleware or batch processing
Building too much custom code	This can increase testing, documentation & support effort
Ignoring monitoring	Failures may not be spotted until users report operational issues
Weak ownership	No one is clear who investigates or fixes problems
Poor documentation	Harder to support, validate or upgrade later
Treating validation as an afterthought	This can create delays in regulated environments
Letting middleware become a black box	Orchestration becomes harder to understand and govern

IFS Integration Planning checklist

Our practical checklist is a great way to kick start any IFS integration project. This can be shared across key stakeholders and decision-makers to aid their understanding of the process, technology used and reasons for overall integration approach.

IFS Process

- Has the IFS process or object been identified and documented?
- Is the integration inbound or outbound?
- Is this transactional or batch activity?

Connected System

- Can the external system use REST APIs?
- Does it require SOAP, XML, CSV or file transfer?
- Is middleware available?

Integration Route

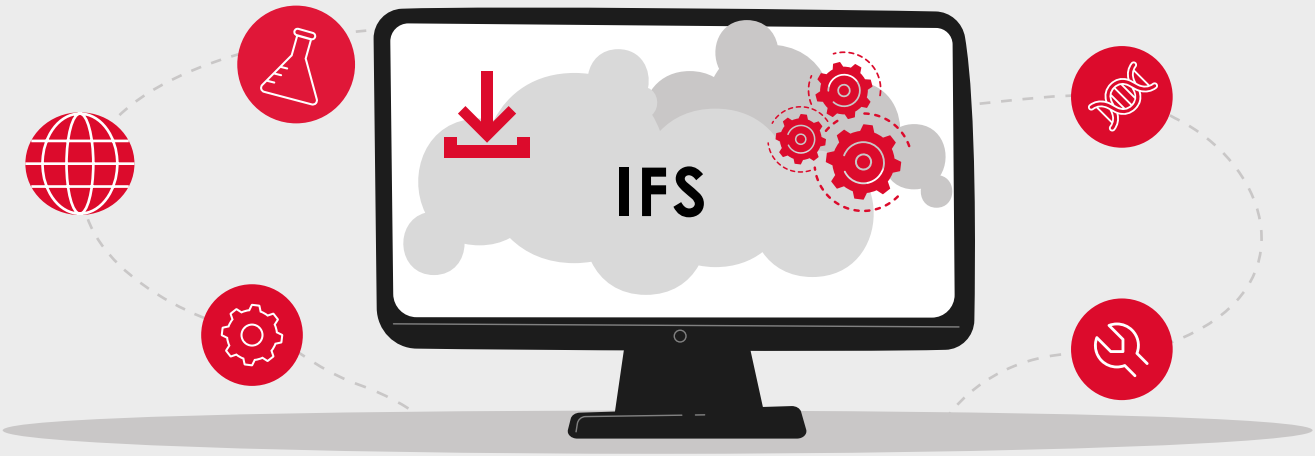
- Is a standard IFS REST API suitable?
- Are Event Actions appropriate?
- Is IFS Connect required?
- Would Migration Jobs or External Files be a better fit?

Governance & Support

- Who owns the integration after go-live?
- How will failures be monitored?
- How will the integration be documented?
- How will this behave during upgrades?

Validation

- Is this part of a validated process?
- What evidence will be required?
- What regression testing will be needed?
- Can testing be focused on affected areas?



The Takeaway

Every decision for IFS integration approaches should be practical and methodical.

REST APIs, middleware, file-based methods, Event Actions, IFS Connect, Migration Jobs, External Files and low-code approaches can all have a role. The right choice depends on the process, the data, the connected system, timing needs, governance and long-term support.

Ultimately, the most successful integrations are those that work clearly, reliably and remain manageable and well supported as the business grows and develops.

The Muzulu View

Muzulu helps organisations choose the right IFS integration approach to fit:

- The process
- The system landscape
- The governance requirements
- The business goals

Talk to our team today to get started.