happiest minds
The Mindful IT Company
Born **Digital** . Born **Agile**

Technical Whitepaper

How to Approach Tool Integrations

Contents

Integration overview

Tool Integration is an effective technique of integrating tools of the same or different classes to build a robust tool framework to support various business operations.

Need for tool integration

There are multiple tools specialized in performing certain day-to-day operations at an enterprise level and deliver the required results. These tools can be anywhere from traditional or legacy applications to modernized applications and support various business processes.

If the tool is self-sufficient/specialized to cater for the business processes and does not require sharing the data/ information across any other tool, in that case, there is no need for integration.

However, there may be a scenario where the data generated/submitted in one tool/application requires an external application to perform extended task owned by multiple groups in an organization. Such scenarios demand the integration of other tools/applications.

The other typical business needs could be

Standardization of processes across multiple groups

Adoption of new process by the organization as part of new initiatives

Process upgradation

External audit compliance requirements

Acquisition/merger of two different sub-units or enterprises

Categories of tools/ integration

The tools involved are broadly categorized into the below

01 SaaS-based tools/applications

02 Open-source tools/applications

03 Homegrown tools/applications

04 Web-based/customer end-user portal applications

05 Modernized applications (server-less)

06 COTS tools/applications

Type of integrations mainly include

01
Uni-directional integration

02
Bi-directional integration

03
Point-to-point integration

04
Middleware based tool integration

Key drivers for integration

- Information needs to flow from one tool/application/system into another environment
- Keeping near real-time sync between the two tools
- The process involves heavy usage of swivel seating and decreased productivity
- Any change getting introduced to the business processes as part of standardization/upgradation of existing processes
- Avoid manual errors while performing repetitive tasks across multiple tools/applications

Key Benefits of integration

- Users will be able to view consistent data across tools/ application at any given point in time
- Seamless information exchange between tools/applications in the real-time
- Enhanced process automation

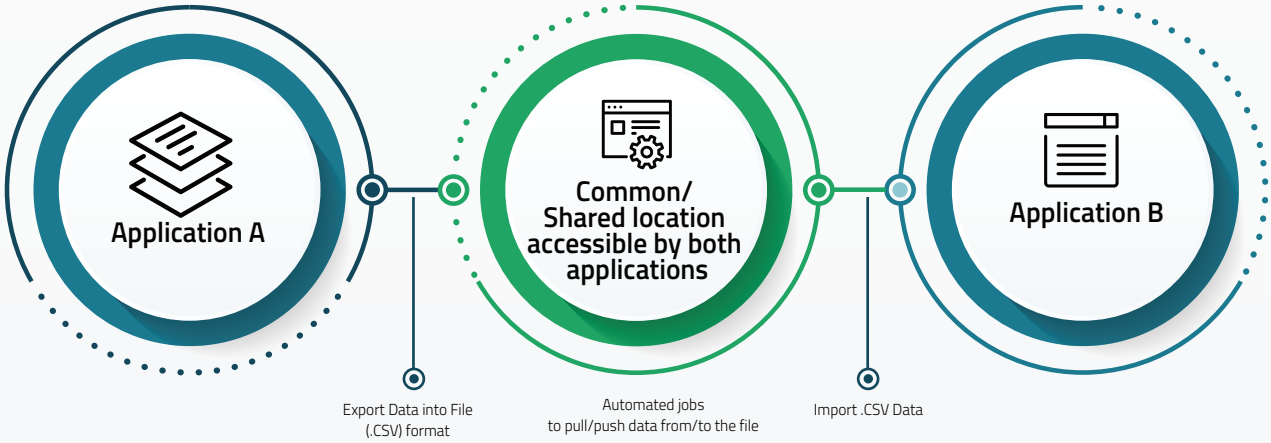
Common techniques to achieve tool integration

There are lots of ways to achieve tool integration. Each of these below listed common approaches have their own pros and cons. The most optimal approach could be taken up for implementation after a thorough evaluation.

File based Integration

This approach is generally taken up with no need for direct connection between the two tools. The integration is achieved by placing the required file in a pre-determined folder location/agreed format at a scheduled time of the day by the tool and is accessed by the other tool

File based integration



Pros

- Less expensive solution with quick time to implement
- No need of APIs and custom development
- CSV format is easier to access, more readable and could be validated by end-user

Cons

- Data sync between the two tools is not real-time
- Lack of Data validation leading to import of corrupt data
- Lesser security as data resides in a flat file without encryption

Email based Integration

Certain tools with limited capabilities may have email-based integration based on the emails received from the external tool. Certain actions could be triggered to perform the required automation and preferred in unidirectional with lesser complexity.

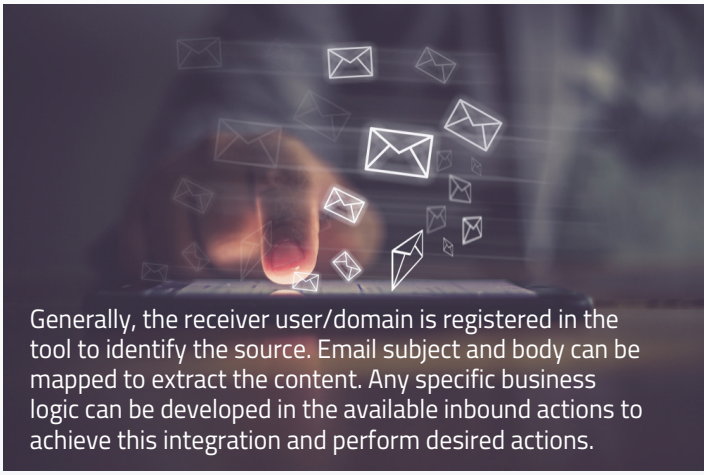
There can be two categories

Inbound

Emails coming into the tool

Outbound

Emails going out of the tool



Generally, the receiver user/domain is registered in the tool to identify the source. Email subject and body can be mapped to extract the content. Any specific business logic can be developed in the available inbound actions to achieve this integration and perform desired actions.

Pros

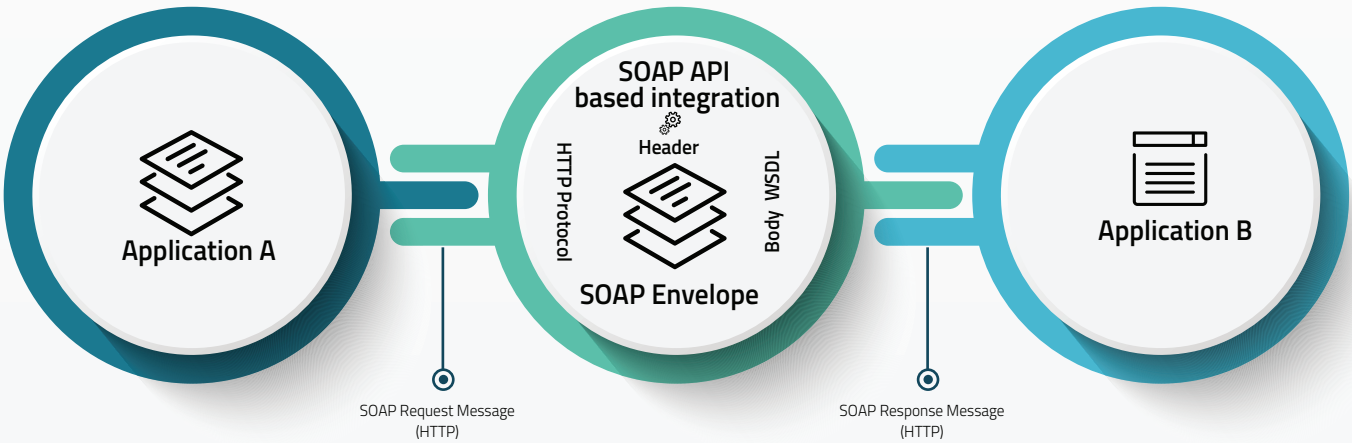
- Ready templates lead to faster implementation
- Most of the tools with basic maturity and features support email integration

Cons

- Delay in receiving emails could be an area of concern for processes that are having very short cycle time
- Handling of special scenarios like email signatures, logos, URLs, and type/count/size of attachments could pose challenges

SOAP (Simple Object Access Protocol) based API integration

SOAP leverages XML and advocates a strong messaging framework. The service operations involved are defined with the appropriate XML request/response structure. There will be input, and output parameters encapsulated in the WSDL (Web Service Description Language). The tool that exposes the API is referred as a publisher, and the one that consumes is called consumer of the web service. SOAP request body contains Query parameters, Path parameters, Form parameters, Header parameters and Body parameters



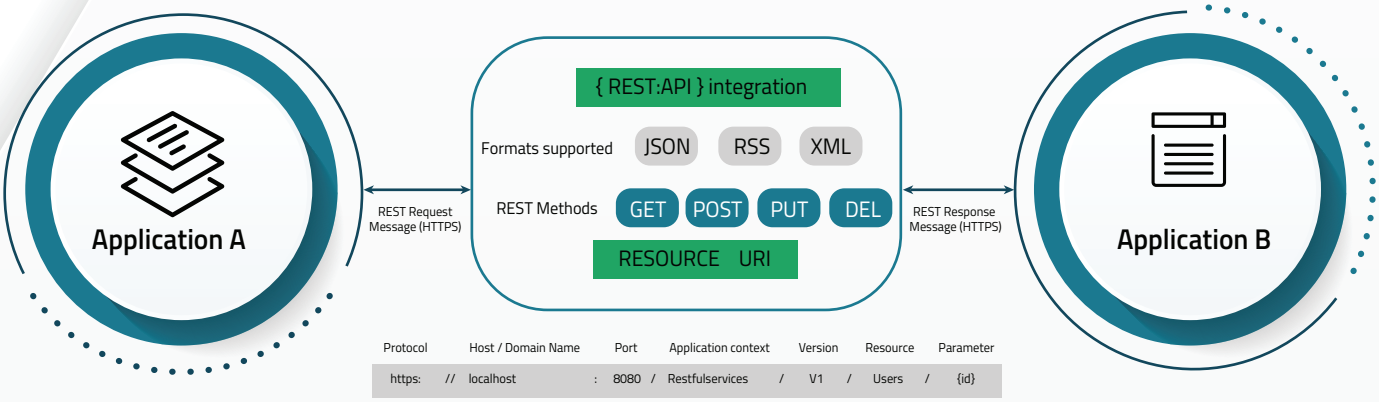
Pros

SOAP can communicate over any transport protocol primarily uses XML

Cons

Even if there is a slight change to the API such as a single parameter, the entire WSDL will undergo changes and thereby resulting in recompilation of changes to the client application

REST (Representational State Transfer) based API integration



Pros

Simple to use and low utilization of resources

Handling of different format of data

Cons

Not able to maintain session

Direct database connection integration

e.g.- JDBC/ODBC (Java Data Base Connectivity/Open Data Base Connectivity)

ODBC (interconnector) act as a middleware between databases and applications. It provides database independence by using the interaction of the driver and the application on the corresponding application platform and the corresponding database to avoid the direct call to the database.

Pros

Common interface supporting most of the popular databases and widely used with built in functions, pre-configured and other capabilities/ connectors to establish the database connection

Cons

Performance issues with larger volume of datasets

Middleware application

e.g.- Talend, JBoss, Mulesoft

The middleware acts as a system that communicates with both the systems involved to perform any translations of data and ensures seamless information and integrity.

Pros

Enables process streamlining and efficiency improvisation

Cons

System performance could be an area of improvement to optimize the processing time and have real time information

Hub-Spoke mechanism

This provides a modular approach when there are multiple endpoints to be integrated and leads to standardization.

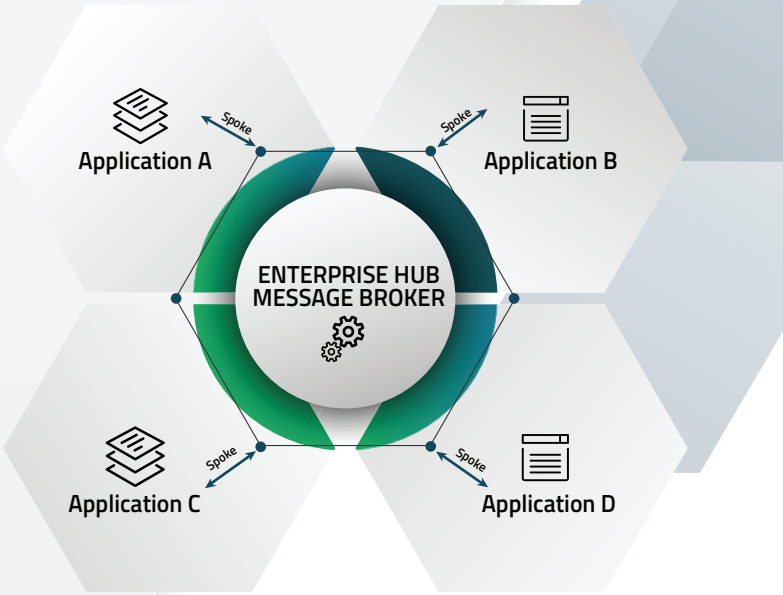
Pros

Ability to handle larger volume of traffic with better architecture provides the ability to easily scale with robust framework

Cons

Any design changes to the hub will lead to complex re-work due to the centralized architecture

Hub - Spoke integration model



Service-Oriented Application Integration (SOAI)

Identified standard services that are being offered from a centralized server reusing the objects and methods

Pros

Provides high reusability with a common business logic and methods leading to shortened development cycles

Cons

Typically, consistent technology and architecture are rarely observed in enterprises due to the various platforms and tools

Orchestration

This is the most advanced level of integration with usage of automation of various business processes using multiple tools

Pros

Provides the ability to stitch the various process across heterogeneous platforms/ toolsets using the workflow

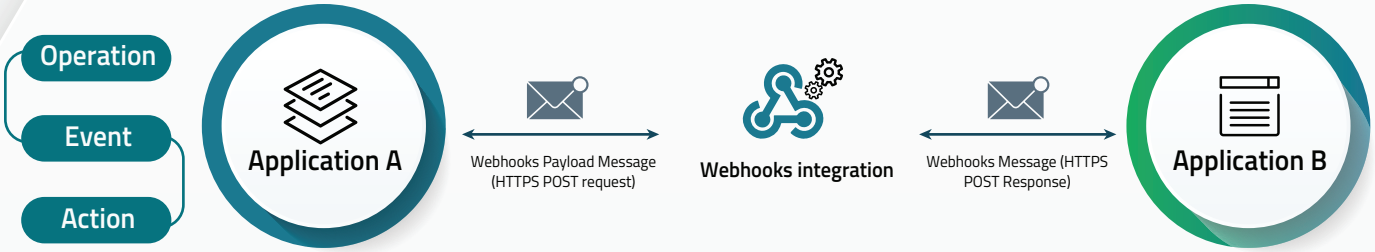
High amount of automation possible including self-healing and other L0/L1 tasks activities

Cons

Generally, the processes are not well documented

Process needs to be completely standardized without any gaps before taking up the automation tasks

Webhooks



Certain web applications may not directly expose or consume APIs and instead encourage the use of webhooks that are event based.

Pros

Instant data is delivered using webhooks almost on a real-time basis with minimum utilization of resources

Compelling features and simple to implement

Cons

While the primary focus of Webhooks is on delivering data in real-time, still there are improvements needed on application error scenarios

Application Integration Challenges

01 Lack of OEM Vendor support for custom integration

02 Lack of enterprise-grade API made available by OEM/product vendor for various actions

03 Lack of development and test environments for testing the integration before production release

04 Lack of extensive/detailed documentation on available APIs from the tool vendor

05 Lack of clarity in business requirements between the parties involved

06 Requirements getting defined turn agile, leading to scalability/-maintainability issues

07 Infinite looping in bi-directional integration due to the absence of defined business rules

Environment overview

It is imperative to understand the business objectives that need to be achieved by involving all the relevant stakeholders/parties from the customer side, including tool and process experts, right at the very outset of the initiation phase. This can be achieved through workshops to understand various components in the current/AS-IS tools architecture in the customer environment, current processes, and workflow. The process in the future state will have to be documented and socialized with the stakeholders prior to tool integration.

This will go a long way in building the required tools/process integrations by taking necessary measures for any course corrections during design considerations in the future state. The typical outcomes of these working sessions will be data flow diagrams in Visio or a similar format.

High-level steps involved in typical integration

01

Gathering the business requirement

02

Tools for validation of integration endpoints

Postman SOAP UI

03

Maintainability - Checklists and SOPs

High-level pre-requisites for enabling integration

Key parameter
Dedicated system/service accounts with credentials and appropriate level of permissions such as Tool admin/API admin for accessing endpoints with an ability to perform Read/Write operations
Identification of lower instance, preferably such as Dev/Staging and leverage it for development/ testing activities
Identification of appropriate authorization and authentication mechanisms of the tools
Configuration of service accounts
Firewall port opening between the applications (if any)
Identify the various actors in the process, triggers, qualifiers, actions to be performed and business logic involved for both the positive and negative use case scenarios
Understanding the limitations of the tools at early stages by going through the product documentation or consulting with the product SMEs
Identification of the required skills that include development exposure and usage of APIs (publishing and consumption), level of scripting involved, validation tools and the cross-training/upskilling of team members getting involved
Both the tools involved must be able to perform API calls as well as parse and store the response received, including error scenarios

Integration Design considerations

General design considerations when defining any integration

Understanding the current volume of transactions and forecasted load that needs to be supported by integration. If a high-volume transaction is foreseen, we could explore the multi-threading features of the application.

Designing/optimizing the To-be process and workflow involving the various support groups is crucial.
Understanding and documentation of the business use cases with relevant events and conditions
Understanding the tool family – If the tools/applications are Homogeneous (i.e., from the same OEM vendor), there may be native integrations in the form of pre-configured APIs with minimal changes.
For heterogeneous tools/applications, there may be built-in or custom plugins available on the source and target application.

Determine the frequency of data sync between the source and the target (e.g., Real-time, Scheduled interval)
Complete flow of data between source and the target application
Feasibility of storing of data in source and the target application
Involvement of any middleware tools/connectors/plugins/third party apps
Any complex non-functional requirements around API transaction that require multiple calls either in sequence or in parallel.
e.g., In certain rare scenarios, output from the first API call may be the input for the subsequent API call(s)

Technical design factors involved while defining integration

01 **Type of integration – Uni-directional/ Bi-directional**

02 **Understanding the integration protocol for communication – (e.g.-: REST/SOAP) supported by source as well as the target application**

03 **Understanding the source and target tool capabilities**

- Authentication mechanism (e.g., - Basic, OAuth, SSL, KeyStore, Mutual Authentication and others)
- Pull and push mechanism for handling data
- Controlled access to APIs to read, write, query, and delete operations
- Simultaneous updates to the same record from the front end-user interface and external API calls will lead to locking the record/table at the database level. By leveraging webhooks or any other similar technologies, locking mechanism can be handled
- Ability to invoke required APIs on the defined user/system events for only refined/ required filter criteria
- Message format supported by source and target (e.g.- XML, JSON)
- APIs made available out of the box and documenting any limitations on the source and target side
- Special fields (e.g., date, notes, comments, attachments, unstructured data formats, email signatures, reference URL links and others)
- System date formats configured on the source and target side

04 **Evaluation of accelerators such as pre-built connectors from the vendor or third party**

05 **Security restrictions, if any, such as firewall rules, VPN and Cloud**

06 **Field mapping between the tools**

07 **Handling of Special fields**

08 **Special scenarios**

09 **Underlying business rules**

10 **Error handling**

11 **Determining/Defining audit mechanism**

12 **Any other design standards/Best practices**

13 **The handiness of test data and test scenarios**

Identification and Communication of Technical Limitations

Based on the detailed assessment/workshops/POC results and by consulting product SMEs or OEM vendors, key technical limitations can be identified/foreseen. In such cases, it is crucial to communicate these limitations to all the stakeholders involved in setting clear expectations before the implementation phase. This would minimize the business impact, project costs, development efforts, and an alternative approach could be postulated.

Integration Development Standards

Naming conventions

Determining the name of user accounts/service accounts - This may have to be consulted with the tool administrators to maintain consistency across the board.

The naming of business rules/events/ triggers will ensure that the code is maintainable and comes in very handy during any troubleshooting.

Integration Validation

01 A dedicated group of system testers with a combination of pilot users

02 A separate UAT environment to catch any issues early before getting released to production

03 Complete list of test cases that cover both positive and negative scenarios

04 Each test case should, to a minimum, capture the various actors initiating the process, required triggers/events, exact qualifiers, and actions to be performed, user notifications

05 Visual representation (in any form such as Visio/Flowchart) capturing the data information flow

06 Stitch the integration test cases into regression test case suites as applicable to cover overall functionality

07 Test data needed for basic validations/ troubleshooting needs to be arrived at based on any past ticket information to a near-real-time validation

Monitoring Integration

Integration Health Check

01 Validate services are up and running

02 Start/Restart the service if it is not in running status

03 Validate CPU and memory utilization Restart-Service if CPU/Memory utilization is more than the configured threshold percentage

04 Validate disk space and ensure the minimum amount of disk space is available If no sufficient space available, clean up any logs and restart services by following Standard Operating Procedure (SOP)

05 Build appropriate reports as necessary to track any issues proactively

Integration Troubleshooting

Troubleshooting Basics

01 Ensure authentication and connectivity

02 Visual representation (in any form such as Visio/Flowchart) capturing the data information flow

03 Listing/Documenting of commonly occurring issues

04 Referring Log files – best practice error message and description

05 Standard Operating Procedure capturing steps

06 Handling timeout issues

Best Practice Guidelines

01 Detailed project plan with clear tasks/activities and milestones

02 Defined RACI (Responsible Accountable Consulted and Informed) matrix for the various teams involved

03 Each developers' artifacts, such as checklists, templates need to be reviewed and certified by the Centre of Excellence

04 Follow development and design standards

05 Detailed review of a communication plan with stakeholders prior to rollout

06 Start with a Proof of concept and gradually expand instead of the big bang approach

07 Inputs from previous implementations as lessons learnt and success stories

AUTHOR BIO

Chethananand N K is a seasoned ITSM professional with diverse experience of leading enterprise-level engagements in ServiceNow, BMC Remedy and other ITSM tools/ solutions and successfully delivered complex integration projects to automate business processes.

He is currently working as Sr. Practice Manager for ITSM/ ITOM toolsets portfolio at Happiest Minds Technologies. He is actively involved in carving out the use cases during the solutioning and planning phase in large infrastructure transformation projects. Chethananand is also responsible for architecting solutions and road map discussions to ensure smooth transitioning of services and steady-state operations.

Chethananand N K
Senior Practice Manager, ITSM

happiest minds
The Mindful IT Company
Born Digital . Born Agile

www.happiestminds.com

Business Contact **business@happiestminds.com**

About Happiest Minds Technologies

Happiest Minds Technologies Limited (NSE: HAPPSTMNDS), a Mindful IT Company, enables digital transformation for enterprises and technology providers by delivering seamless customer experiences, business efficiency and actionable insights. We do this by leveraging a spectrum of disruptive technologies such as: artificial intelligence, blockchain, cloud, digital process automation, internet of things, robotics/drones, security, virtual/augmented reality, etc. Positioned as 'Born Digital . Born Agile', our capabilities span digital solutions, infrastructure, product engineering and security. We deliver these services across industry sectors such as automotive, BFSI, consumer packaged goods, e-commerce, edutech, engineering R&D, hi-tech, manufacturing, retail and travel/transportation/hospitality.

A Great Place to Work-Certified™ company, Happiest Minds is headquartered in Bangalore, India with operations in the U.S., UK, Canada, Australia and Middle East.