|  |
| --- |
|  |
| **Integration Draft Strategy**  **Coop DK** |

Role and Reason for Approval

|  |  |
| --- | --- |
| **Role** | **Reason for Approval** |
| Author | The author is signing to confirm that this document has been prepared in accordance with the programme document management process, that relevant input from any contributory authors has been included and that an appropriate review/editing process has been conducted. |
| Transformation Lead or Manager, or Business Process Owner | Transformation Lead, Transformation Manager or Business Process Owner is signing, on behalf of the Workstream, to confirm that this Functional Specification meets the Acceptance Criteria expected of it and assigned to it in the Deliverable Quality Log. |
| TCS Development Lead or Manager | The Development Lead or Manager is signing, on behalf of the Development Team, to confirm that this Functional Specification meets the Acceptance Criteria expected of it and assigned to it in the Deliverable Quality Log. |

*Note. Master copy of this document, with signatories, is held on Solution Manager*

| **Version** | **Date** | **Name** | **Alteration Reason** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Note. The version number of the document will be 1, 2, 3 etc.… (full integer numbers) there will only be a change in version number when updating a previously approved document.

# Referenced Documents

|  |  |  |
| --- | --- | --- |
| **Sl. No.** | **Document Name** | **Document Link** |
|  |  |  |
|  |  |  |

# Document Approvers

|  |  |  |
| --- | --- | --- |
| Name (Position) | Signature | Date |
|  | X | X |
|  | X | X |
|  |  |  |
|  |  |  |

# Document Distribution List

| Name | Title & Area | Action (Information, Review or Sign-off) |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Table of Contents

[1 Context 6](#_Toc506149658)

[1.1 Business Background 6](#_Toc506149659)

[1.2 Scope and Objective 9](#_Toc506149660)

[1.3 Phases of Integration 9](#_Toc506149661)

[1.4 Out of scope 9](#_Toc506149662)

[1.5 Dependencies 10](#_Toc506149663)

[1.6 Assumptions 10](#_Toc506149664)

[1.7 Risks 10](#_Toc506149665)

[1.8 Links 10](#_Toc506149666)

[2 Non Functional Requirements 11](#_Toc506149667)

[3 Integration Tools 12](#_Toc506149668)

[3.1 List of Integration Tools 12](#_Toc506149669)

[3.1.1 Mulesoft 12](#_Toc506149670)

[3.1.2 VL Trader 12](#_Toc506149671)

[3.1.3 SAP Data Services 12](#_Toc506149672)

[3.1.4 Opswise 12](#_Toc506149673)

[3.2 Decision Matrix 13](#_Toc506149674)

[4 Integration Design Guidelines and Principles 14](#_Toc506149675)

[4.1 Favor To-be Design 14](#_Toc506149676)

[4.2 External Contracts 14](#_Toc506149677)

[4.3 Mulesoft for all Routing 14](#_Toc506149678)

[4.4 Canonical Data Models 14](#_Toc506149679)

[4.5 Design Patterns 14](#_Toc506149680)

[4.6 Reference Data Mapping Logic 14](#_Toc506149681)

[4.7 Standard Connectors 15](#_Toc506149682)

[5 Integration Design Considerations 16](#_Toc506149683)

[5.1 Integration Type 16](#_Toc506149684)

[5.2 Processing Time 17](#_Toc506149685)

[5.3 Permanent 17](#_Toc506149686)

[5.4 Ownership 18](#_Toc506149687)

[5.5 Distribution 18](#_Toc506149688)

[5.6 Volume 18](#_Toc506149689)

[5.7 Frequency 19](#_Toc506149690)

[5.8 Security 19](#_Toc506149691)

[5.9 Synchronization 20](#_Toc506149692)

[5.10 Error Handling 21](#_Toc506149693)

[5.11 Volatility 21](#_Toc506149694)

[5.12 Mapping 21](#_Toc506149695)

[6 API Management Strategy 22](#_Toc506149696)

[The use cases of API are listed below 22](#_Toc506149697)

[6.1 API Types 22](#_Toc506149698)

[6.2 Cloudhub and On-premise Mule Instances 23](#_Toc506149699)

[6.3 API Manager and Core Mule Servers 23](#_Toc506149700)

[6.4 Core, Process and Experience APIs 23](#_Toc506149701)

[7 Integration Patterns and Approach 24](#_Toc506149702)

[7.1 List of Patterns 24](#_Toc506149703)

[7.2 Decision Matrix 24](#_Toc506149704)

[8 Continuous Integration and Deployment 25](#_Toc506149705)

[8.1 Workflow 25](#_Toc506149706)

[8.2 Code Versioning 25](#_Toc506149707)

[9 Monitoring 26](#_Toc506149708)

[10 Error Handling 27](#_Toc506149709)

[11 Security Features 28](#_Toc506149710)

[11.1 OAuth 28](#_Toc506149711)

[11.2 GDPR 28](#_Toc506149712)

[12 Documentation 29](#_Toc506149713)

[12.1 Architecture Documents 29](#_Toc506149714)

[12.1.1 Mulesoft 29](#_Toc506149715)

[12.1.2 SAP DS 29](#_Toc506149716)

[12.2 Design Documents 29](#_Toc506149717)

[12.2.1 High Level Design Document 29](#_Toc506149718)

[12.2.2 Pattern Specification Document 29](#_Toc506149719)

[12.2.3 Technical Design Document 29](#_Toc506149720)

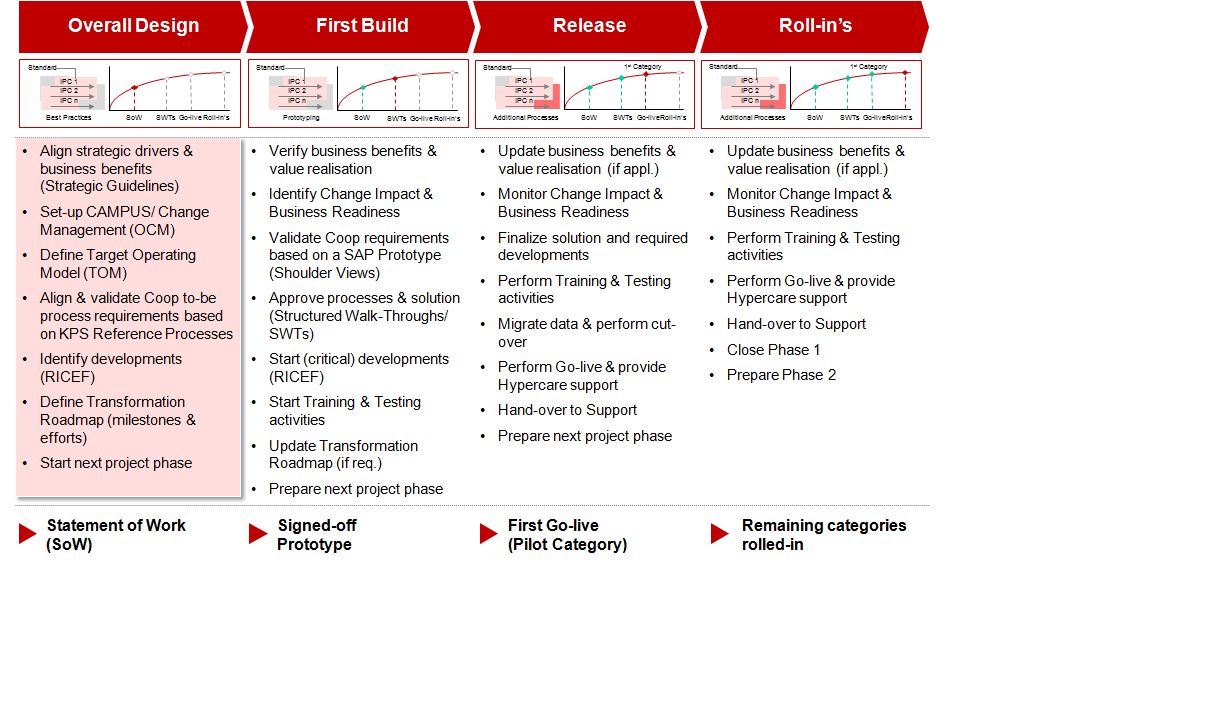
# Context

## Business Background

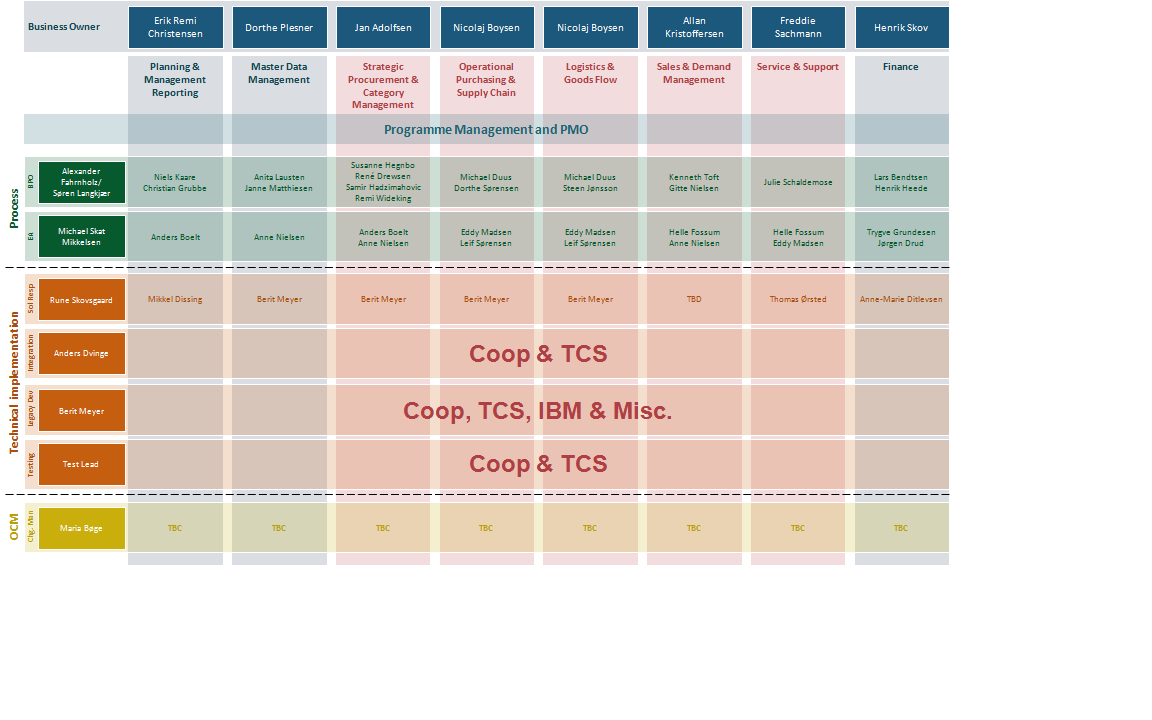
Coop CoreERP Program is a Retail Transformation Program of Coop Denmark which Transform the Business process for COOP Retail Business. CoreERP will be the foundation for Coop`s future strategy. SAP S4 Hana System will be implemented for this Business Process transformation. This program will adopt Standard SAP Business Process. The CoreERP Scope will cover Supplier, Warehouses and Stores with key business process as mentioned in the below diagram.

Phase 1 of the Program is divided into 4 Project Stages. Following diagram outline the Phases, Key deliverables and Milestones. The diagram also details the Core below ERP programme with expected outcomes by phase. The programme will identify all dates and estimates in the “Overall Design” Phase.

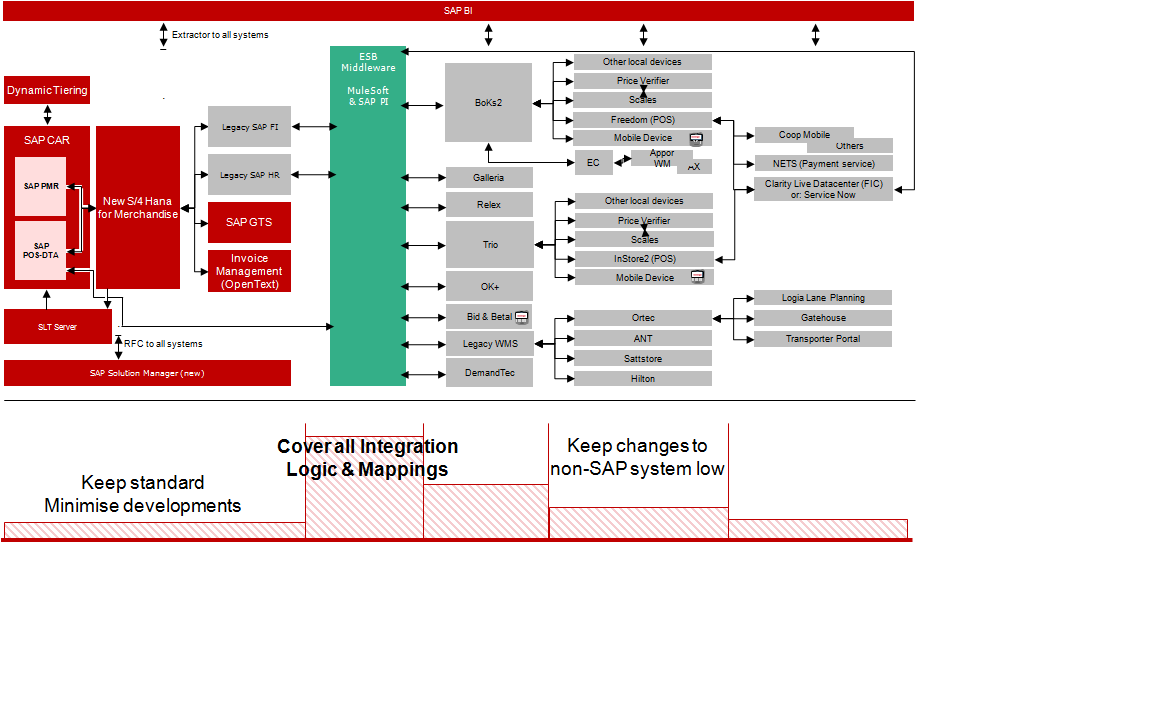
The following diagram details the programme with expected outcomes by phase. The programme will identify all dates and estimates in the “Overall Design” Phase. This document serves as a “Draft Migration Strategy”, which will define the components of the work required to successfully implement the SAP Core ERP with existing systems:



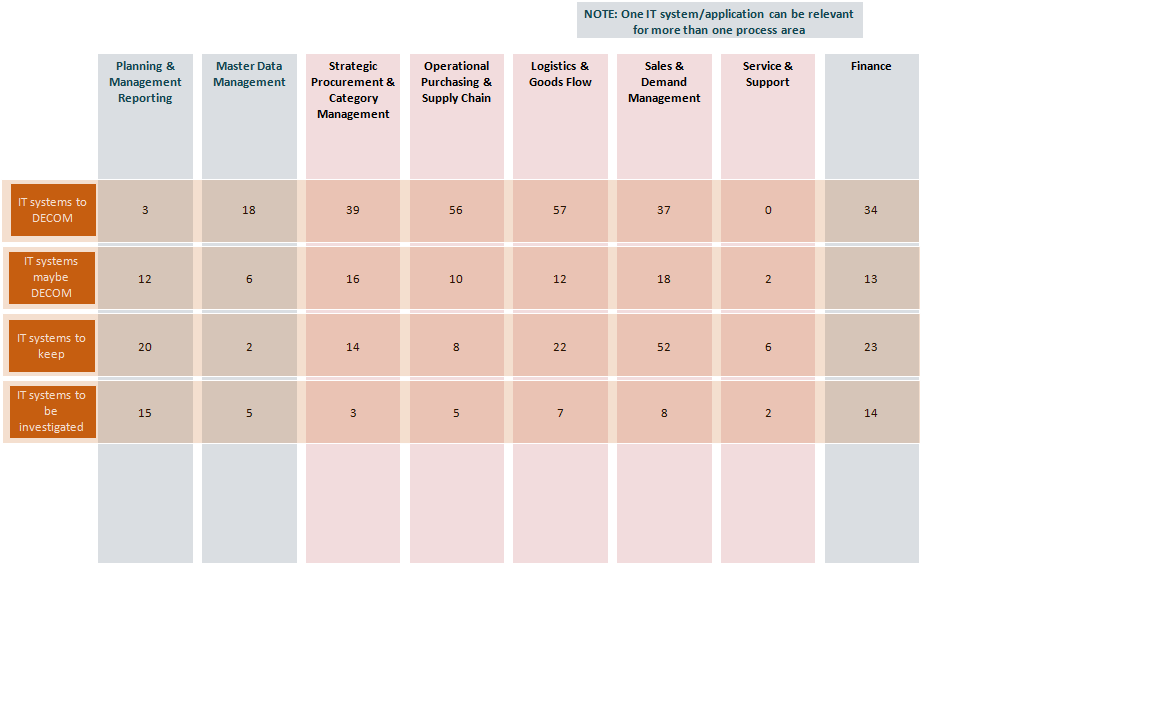
The following diagram details the business and technical team aligned by function to drive the programme:



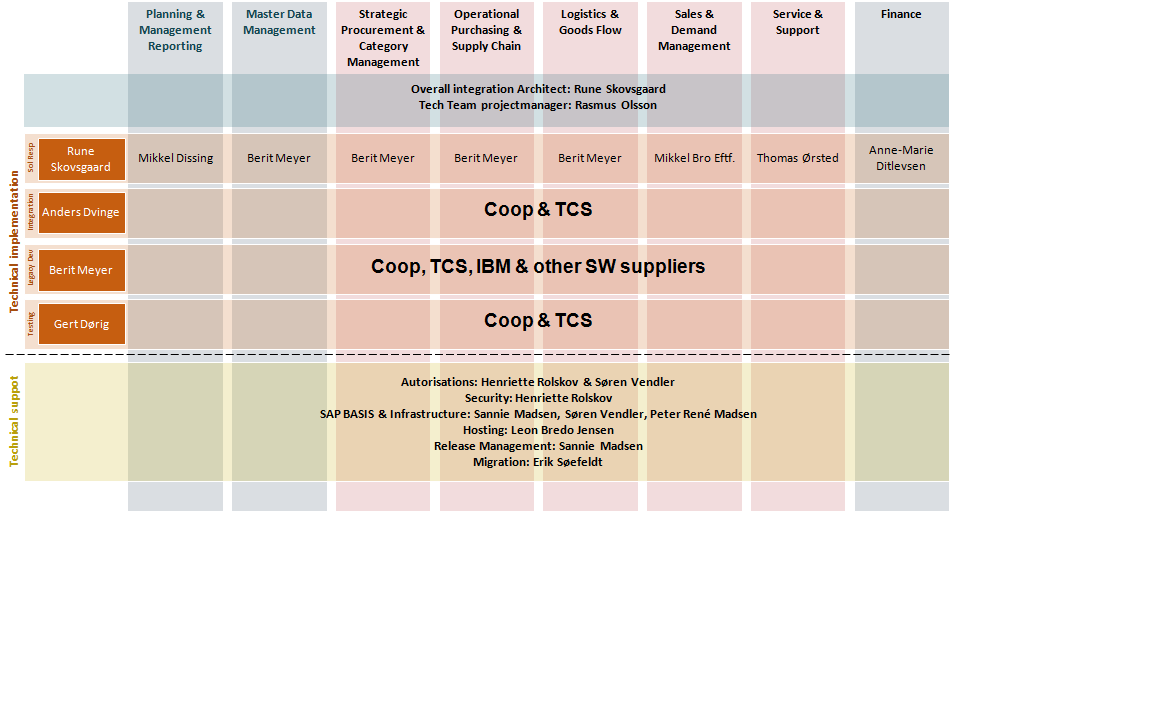
The following Architectural “To Be” diagram indicates all of the existing (on right) and new capability brought in through the SAP implementation. Data flows are shown at a high level in the diagram, which will be further explained in this document:



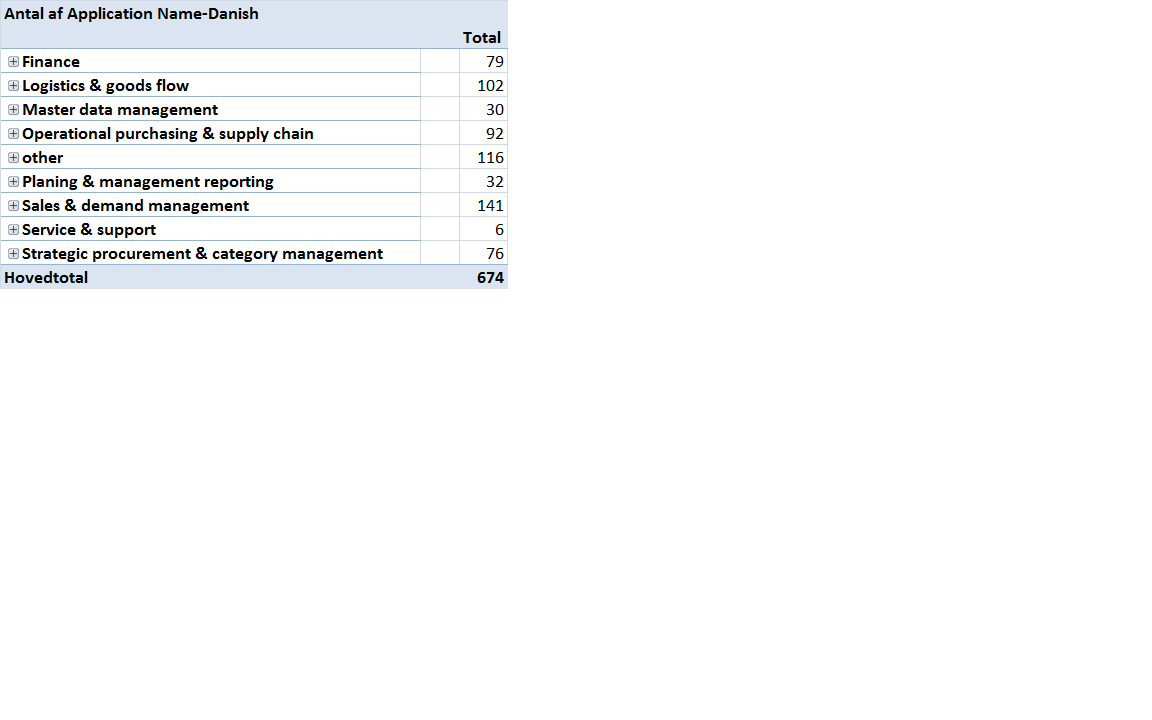
The following diagram indicates known IT Systems across Coop. Through planned workshops, greater understanding will be achieved linking new capability against existing systems. The business will need to make the final determination of which systems will be retained based on the benefits:



The following identifies the technical team and alignment with the eight business functions. Collectively, the parties involved will be required to provide inputs into the estimates for the “Overall Design Phase”. Coop will work with the third party providers in securing the estimates for existing applications that will require enhancements/changes:



The diagram below lists the applications across the enterprise. The Estimation Factors: Total Number of Systems/Interfaces developed – existing systems, and new development for SAP will be taken from assessing the applications for Integrations, Data, Reporting/Forms, and inputs for testing:



## Scope and Objective

The Retail Program is a large transformation of Coops current IT-system and processes, which affects the whole organization.

The current IT-system is old and considered “a burning platform”. Additionally, Coops strategy i.e. simple, local and digital focuses on better digital solutions that can harness benefits that the current IT-system cannot.

Objective of Core ERP programme is to replace IBM host with SAP, which will be implementing majority of the business processes. Overall design phase will be used to identify the capabilities that will be moved to SAP. For each of these capabilities, the RICEF objects will be identified and estimated.

On a high level, the scope of Integration is to

* Build new integrations between SAP and existing Coop applications for the SAP business processes to run
* Modify existing interfaces to intercept the data flow and extract necessary information
* Build temporary interfaces for the coexistence phase
* Build Integrations that impacts decommission of an existing IT application

This will be done in phase wise manner. Each phase involves moving a category from Host to SAP. The purpose of an integration platform is to make it faster, easier and more flexible to get new systems and applications.

## Phases of Integration

## Out of scope

The Integrations listed are considered out of scope:

* Interfaces between applications excluding Host and which are not impacting the overall solution will follow the as is design
* Transformations in SAP PO, if any, will not be taken care by TCS

## Dependencies

* Analysis and design phase requires participation from application owners. Integration design is dependent on the turnaround of both source and target systems
* All certificates used for authentication will be provided by application owner

## Assumptions

* Mule Enterprise Edition and cloudhub are setup already. Three environments are functional i.e. test, QA and production

## Risks

## Links

# Non Functional Requirements

The requirements that help us to judge the operation of a system are termed as non functional requirements. For each of the Integrations, below set of Non functional requirements will be captured during design phase and tested during one of test cycles.

* Volume and Frequency
* Severity to be allotted in case of failures
* Service level agreement
* Target application downtime
* Security & Authorizations

Volume and Frequency: The number of messages that will be generated by source on an average and the peak volume will be captured during design phase. Volume testing or performance testing will cover the system performance when it is handling peak load.

Severity: Based on the criticality of the interface, severity level will be allotted for each of the Integrations. The levels range from 1 to 4, with 1 being the most severe. Support team must respond to the failures in production based on the severity of the incident.

Service level agreement: For every business process, there will be a particular time period before which the messages have to be processed from source to target systems. These agreements will be captured during the design phase and Integration layer has to be designed to meet the SLAs

Target application downtime: During upgrades, an application might be completely down and no communications can take place during its downtime. These scenarios must be captured and tested to ensure there is no data loss in Integration layer. Based on the source communication protocol, Integration layer handles these scenarios differently.

Security & Authorizations:

# Integration Tools

Depending on the non functional requirements, a combination of tools will be used for Integration. The list of Integration tools along with their capabilities are listed below and for any Integration requirement, decision matrix has to be followed to arrive at the correct Integration tool to be used. Exceptions must be discussed with Coop design authority team.

## List of Integration Tools

### Mulesoft

Coop IT landscape does not have an Enterprise Service Bus. All Integrations between Host and the existing applications are primarily through file transfer protocol. Mulesoft is going to be the Enterprise Service Bus in Coop IT landscape. Mule is a lightweight Java-based enterprise service bus (ESB) and [integration platform](https://www.mulesoft.com/platform/enterprise-integration) that features below capabilities

* Connect applications together quickly and easily
* Data transformations
* It enables easy integration of existing systems, regardless of the different technologies that the applications use.
* The [ESB](https://www.mulesoft.com/platform/soa/mule-esb-open-source-esb) can be deployed anywhere, can integrate and orchestrate events in real time or in batch, and has universal connectivity.

Mulesoft will be the first choice for most of the Integration requirements.

### VL Trader

VL Trader is a file transfer tool that can connect to applications running on various platforms like Windows, Unix, Linux, etc. Currently VL trader is one of the heavily used tools in Coop IT landscape to transfer files from one system to another. VL trader can also connect to applications outside of Coop network.

### SAP Data Services

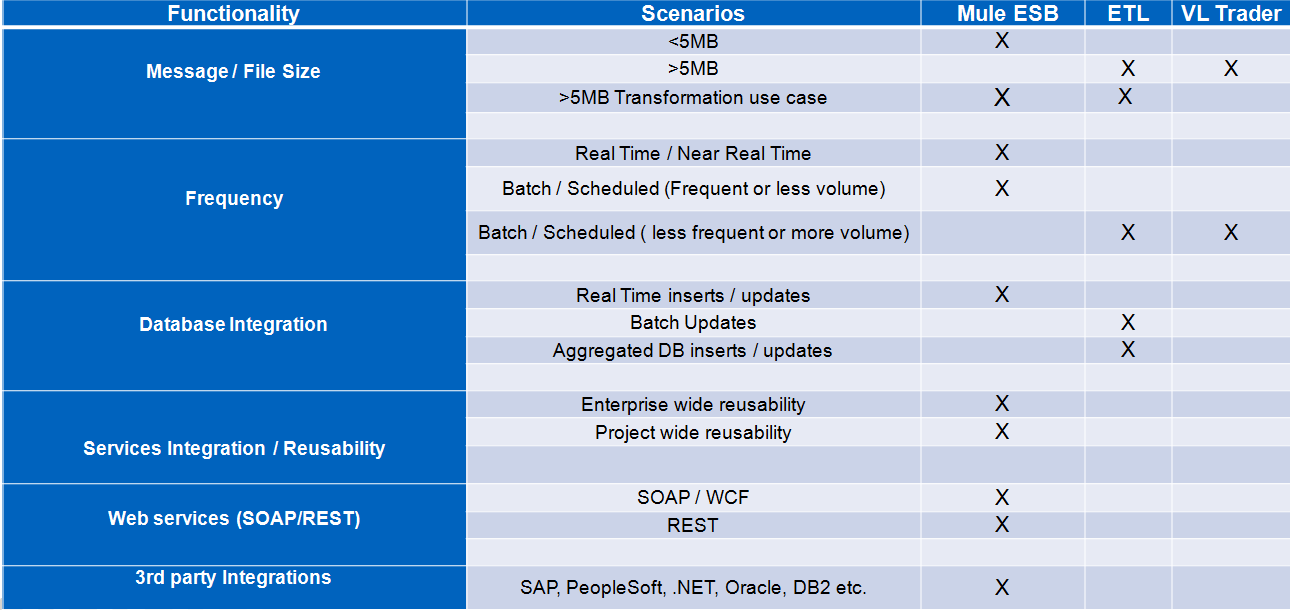
SAP Data Services is a data integration and transformation software application. It allows users to develop and execute workflows that take data from predefined sources called [data stores](http://whatis.techtarget.com/definition/data-store) (applications, Web services, flat-files, databases, etc.) and then allows the user to combine, transform, and refine that data, and then output the results back to the same or different data stores.

### Opswise

Opswise Universal Agent is used for enterprise job scheduling. Opswise Controller provides time-and-date scheduling of an enterprise job scheduler, as well as being a workload automation broker capable of meeting the needs of real-time enterprises.

## Decision Matrix

The tools listed above will be used based on the Integration requirements and design. Broadly, below decision matrix can be referred for choosing the right tool for Integrations. For some requirements, a combination of tools can be used for effective throughput.



This matrix is created based on functionalities like Message size, frequency, communication protocol and the level of reusability. If any particular requirement does not fit into these attributes or a new tool is introduced into the landscape, matrix has to be updated.

Opswise is not listed in the matrix as it is used only for scheduling the jobs and does not play forefront role in Integrations.

# Integration Design Guidelines and Principles

In the initial phase of the project, SAP will implement standard vanilla solution. Integration layer has to be designed for the standard SAP solution to be catered to the existing Coop applications by following the best fit among the principles mentioned below. In many cases, there could be conflicting principles and hence the best fit must be chosen based on the requirements and keeping in mind about the future Coop IT landscape. Any deviation from these principles would need to be approved by Design Authority.

## Favor To-be Design

As the project is going to be rolled out in phases, where each phase involves a new category into SAP, the Integration design must favor the end state of the project and not just the co-existing phase where both Host and SAP are active. Integration layer must be able to handle the movement of categories from Host to SAP with configurable changes and without needing to re-develop the code.

## External Contracts

As much as possible, the design has to have minimum impact on the existing applications. The data contracts of existing applications must remain the same unless there is a necessity of a new field and all required transformations to handle SAP Integrations must be done in Integration layer.

## Mulesoft for all Routing

One of the main problems with the current Coop IT model is the lack of transparency in the data flows between two applications. Going forward, all routing must be done using Mulesoft which will log the transactions into Elastic Search-Logstash-Kibana(ELK) stack.

## Canonical Data Models

The idea of building a canonical data model is restricted to only those business objects which need to be distributed to multiple applications. Otherwise, SAP model will be considered as Canonical data model as it is expected that the future applications of Coop will support SAP data model.

## Design Patterns

All Integrations must follow a defined design pattern. If we come across a requirement which does not meet any of the available design patterns, create a new design pattern.

## Reference Data Mapping Logic

Reference data lookups can be done either in SAP or Integration layer. Irrespective of where it is done, lookups on database must follow micro service architecture. Micro services will be developed in Mulesoft and will be made available to all applications for querying the databases.

## Standard Connectors

The preference is to use SAP standard connectors available in any application than to create a custom component.

# Integration Design Considerations

Several factors should be considered when analyzing existing or proposed interfaces before starting any interface development as mentioned on the following table:

|  |  |  |
| --- | --- | --- |
| **Design Factor** | **Variable** | **Remarks** |
| 1 | Integration Type | Message-Based or File-Based |
| 2 | Processing Time | Real Time vs. Batch |
| 3 | Permanent | Temporary vs. Permanent |
| 4 | Ownership | Internal vs. External |
| 5 | Distribution | Single vs. Multiple systems |
| 6 | Volume | How large is the amount of data to be processed |
| 7 | Frequency | How frequently is the interface used |
| 8 | Security | Does every interface need to be secured |
| 9 | Synchronization | How often is it required |
| 10 | Network | Type of Network LAN/WAN being used |
| 11 | Error Handling | What error handling needs to be in place |
| 12 | Volatility | How often source data changed |
| 13 | Mapping | Tranformation of data |
| 14 | Adapter Modules | Modules for adapters |
| 15 | SFTP-PGP Adapter | Adapter |
| 16 | OS-Level Scripts | Scripts |
| 17 | Message Processing | No changing of messages directly in SAP |
| 18 | File Size | Recommended file size |
| 19 | Business data | Business data will not be stored in SAP PO |

Details on every factor are explained at the following sections:

## Integration Type

**Design Factor 1:** Integration of data comes in two main types Message-Based or File Based

For integration there are two main types in which data is represented as

* **File Based:** where a file is formatted or structured to be interpreted by different application programs. These files will normally contain a header record, trailer record and detail records. The formatting of these files will be in delimited format being COMMA, TAB, PIPE delimited format.
* **Message Based:** normally transactional based and represented in delimited format and following web standards e.g. XML, JSON. For messages which are exchanged with external parties this can be represented in an EDI standard format.

## Processing Time

**Design Factor 2:** Interface processing time corresponds to either real-time or batch processing, this can also be seen as asynchronous or synchronous processing.

**Real Time** interfaces should be avoided unless required by the business, as it can consume more processing power and might impact on-line performance (user experience) when the data volume is high. When a real-time interface cannot be avoided, it is necessary to ensure that only small amounts of data will be processed or break down the data into smaller parts for improved processing performance.

**Batch** interfacing can be run in the background and has minimal impact on the end-user. It is recommended to set-up batch interfaces to run on nightly basis during the batch window time, especially when they process a large amount of data and due to there being limited users on the system.

However, if it cannot be avoided due to business requirements, running during the day time is also allowed under a condition that only small amounts of data are processed.

Guaranteed Delivery will be implemented through the use of acknowledgement messages. Acknowledgement messages will be sent from the receiver systems on receipt of messages and these will be reconciled by the sender systems.

The sender of a message uses the attribute Quality of Service (QoS) to determine how a message is delivered. The following types of quality of service are supported:

* **BE (Best Effort):** The message is sent synchronously. The sender waits for a response before it continues processing.
* **EO (Exactly Once):** The message is sent asynchronously. The sender does not wait for a response. The [Integration Engine](http://help.sap.com/saphelp_nw04/helpdata/en/af/7427da882c1845a23aef5148f0ffad/content.htm) and the [Adapter Engine](http://help.sap.com/saphelp_nw04/helpdata/en/6d/9c44b95c168e4cb500bb430ec193cc/content.htm) guarantee that the message is sent and processed exactly once.
* **EOIO (Exactly Once In Order):** Messages are delivered with the same queue names (supplied by the application) in the same sequence that they were sent from the sender system. Message processing is asynchronous in this case.

## Permanent

**Design Factor 3:** Permanent is a variable that considers whether the interface is only temporarily used as a workaround for a period of time or is permanent with respect to the SAP system.

**Temporary** interface means an interface that will only be used temporarily until a new system interface is available.

Things that need to be kept in mind in this case are:

1. Do not establish a specific complex solution if it will get thrown away in a short period of time.
2. Prepare a communication method that is very similar to the permanent one, so that only minimal effort is required to change to the permanent solution.
3. If necessary, utilize manual interfaces to reduce development effort.
4. Limit the changes in the SAP application for the interface, and make changes in the middleware application.

**Permanent** interface means it is a final solution for the project. Therefore, a thorough and extensible design solution should be considered before developing the interface to ensure easy maintenance in the future.

## Ownership

**Design Factor 4: Ownership** is a very important variable to ensure the development can be done successfully in a timely manner.

**Internal (Inside Firewall):** Internal ownership corresponds to any system that is owned, located within the firewall. This type of ownership usually leads to better time management in getting the requirements done.

**External (Outside Firewall):** On the other hand, external ownership corresponds to any system that is located outside the firewall, and is therefore owned and maintained externally by a different company.

This ownership type usually leads to a longer processing time since it requires more communication and service level agreements between parties.

Therefore, it is important to consider any overhead time to each individual interface development to reflect the actual time that may be required to complete the development.

## Distribution

**Design Factor 5: Distribution type of an interface can be distinguished as follows**

**Single System**: Single system means that the communication process is 1 to 1. It requires less time during the design, but extendibility should be taken into account by having a generic data model for future maintenance.

**Multiple Systems**: For multiple systems, the communication process is 1 to many. Multiple systems will automatically require a generic data model to accommodate the multiple system requirements. One important thing is to minimize the amount of data being transferred out (keeping to 1 data type whenever possible).

## Volume

**Design Factor 6: Volume is an important factor as it impacts the processing time**

It is essential to know the expected data volume and processing time requirements of a specific interface early in the design process, so that the interface can be designed and scheduled correctly, e.g. how many parallel jobs should be run to help the processing speed and when is the right slot within the batch window based on its data volume and business criticality. Additionally, it can also play an important role in deciding whether it is necessary to run a full data interface or only delta changes as a data synchronization interface.

If it is a large volume of data, then delta changes are recommended, and full processing should only be run infrequently (weekly, monthly, .). For small amount of data, then a full run can be done more frequently**.**

The volume of data also has an impact on the hardware specifically CPU and Memory. With large message there is an increase on how much data is loaded into memory for transformation activities. If interfaces are processed in parallel to increase performance times then CPU would be impacted.

Use reasonable message sizes to improve performance, to avoid memory overflows and to increase overall system stability.

**The best practice is to keep the average message size in the range of 1 MB to 5 MB, see performance improvement mechanisms below.**

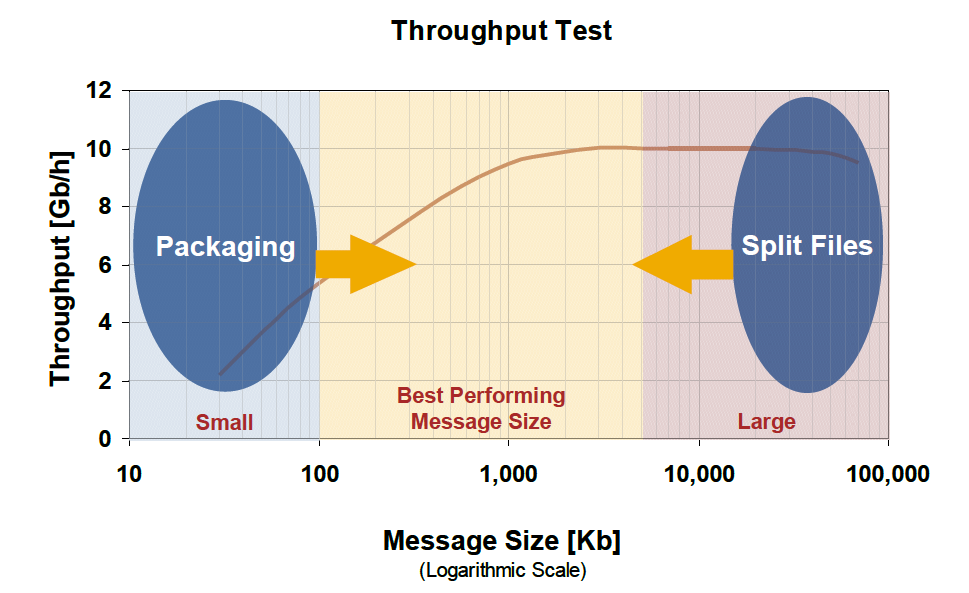


Figure 12 :Throughput Size

## Frequency

**Design Factor 7: Frequency of data exchange will determine how often the interface should be run**

This information is important to help setting up the interface scheduler correctly, e.g. whether it runs at a certain time, is based on any standard & manual triggering event.

## Security

**Design Factor 8: Security is a factor that might be a trade off between functionality and performance**

If an interface is secured (e.g., using HTTPS), it will generally take more processing time, while if it is unsecured, then it will be faster.

Only data deemed sensitive will be made secured. This security of data will also depend if it is being processed internally or externally

There are several factors to enable the security:

* Secured user authentication and authorization. User authentication should be used so that only known and specific types of users can access the system.
* Secured network / communication layer, e.g. https, secure network connection.
* Message level security, e.g. signature, encryption. This type of security can be enabled on all systems that support message based communication.

Anonymous requests should be strictly avoided.For the systems which are within the firewall, account based authentication is recommended.

The communication with 3rd party systems should always use secure protocols. Mulesoft supports secure communication via FTPS, SFTP and HTTPS protocols.

**Authentication and Authorization** should also be considered while interfacing with the 3rd parties which are outside the network. Mule allows following mechanisms to authenticate the connection request –

**Account based authentication** - Users with the sufficient authorization/role can be configured using Open Authentication techniques

For JDBC scenarios jdbc user should have sufficient authorization to read/write to database table as well as call to stored procedure. For SOAP scenarios user should have sufficient authorization to call the web-service hosted on web-server.

**Key based authentication** – This has two flavours. Server Authentication and Client Authentication. It is recommended that the certificates (for example an X.509 certificate) must be used that have been issued by a company-internal Certification Authority or by an external trusted supplier such as Thawte, Verisign, or TC Trustcenter.

**Encryption:** Where it is necessary to encrypt the data, for example the data being transferred to the banks, the encryption process can be built into the individual interfaces. (In most cases, the encryption methods will be specific to the type of data being transferred, so there is no benefit in developing generic services to handle the encryption/decryption). The requirement for encryption should be captured as part of Business Functional Specification.

## Synchronization

**Design Factor 9:** Synchronization is required to keep data consistency

**Synchronization is required to keep data consistency**, especially for an interface that also keeps data on each individual system due to its logic and therefore creates a master slave data relationship, for example SAP holds Customer Master Data and specific systems will refer to that data for their transaction processing.

It is important to determine whether we need to have one or two-way synchronization, since this will impact the logic functionality and its complexity.

This factor is closely related to data volume to determine what changes (full/delta) are required between the two systems.

## Error Handling

**Design Factor 11:** Error Handling

For the interface at different points of processing where the data is transformed and certain conditions or logic is applied it’s important to consider error handling when this transformation process fails, and what the behavior patterns are i.e. blocking retry, exception queue, and return to source of interface to resolve issue.

## Volatility

**Design Factor 12:** Volatility indicates how often the information in the source system changes

It is closely related to synchronization and data volume variables to determine logic complexity and setting up the interface scheduler.

## Mapping

**Standard:** Only Message Mapping to be used for Integration Scenarios. Groovy, XSLT and Java mappings will be used only if the functionality cannot be achieved using graphical mappings.

Mapping messages from one format/structure to another is a fundamental feature of Data Weave component in Mulesoft. It will be used for all standard transformations. Complex transformations involving grouping, aggretation and sorting will be done using groovy/java.

# API Management Strategy

An API is a ‘public persona for an enterprise’; exposing defined assets, data or services for public consumption which can be easily invoked.

API should be considered as an extension to existing integration; SOA services allowing external app developers to easily leverage publicize and/or aggregate a company’s assets for broad-based consumption.

An API management enables the process of publishing, promoting and overseeing application programming interfaces (APIs) in a secure, scalable environment. It also includes the creation of end user support resources that define and document the API.

The goal of API management is to allow an organization that publishes an API to monitor the interface’s lifecycle and make sure the needs of developers and applications using the API are being met.

## The use cases of API are listed below

* Build a broad open web community with simple and free REST APIs
* Support mobile app development with REST APIs
* Facilitate internal use of SOAP APIs and REST APIs for an enterprise services strategy
* Build a B2B community around mission-critical
* Increasing innovation and new business models
* Increasing consumer loyalty

## API Types

* **Private/Internal APIs**

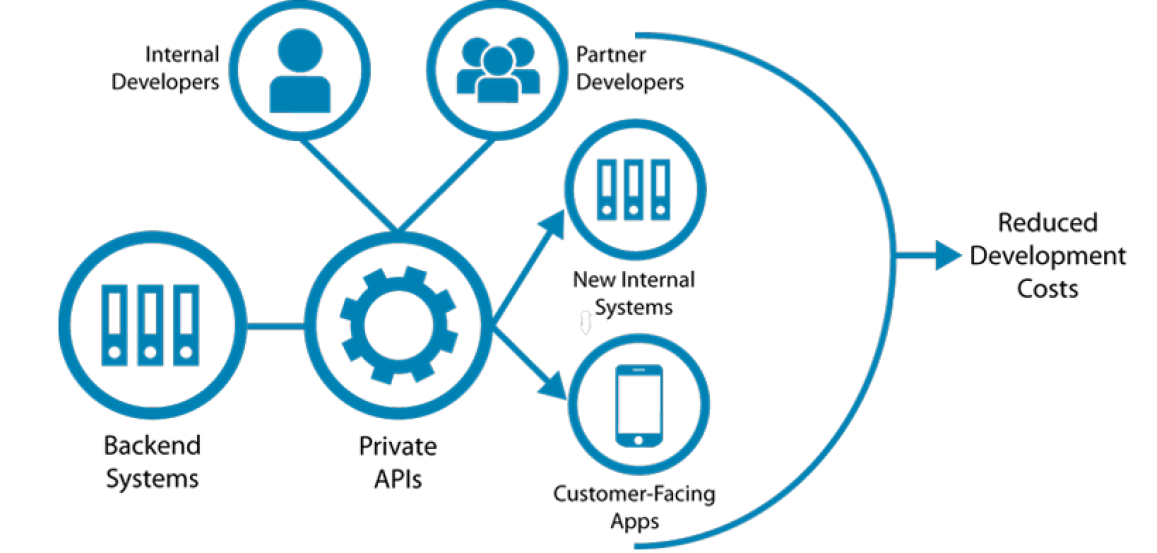
A private API is an interface that opens parts of an organization’s backend data and application functionality for use by developers working within (or contractors working for) that organization.

a. Interfaces are only exposed to internal developers

b. Reduced security risk

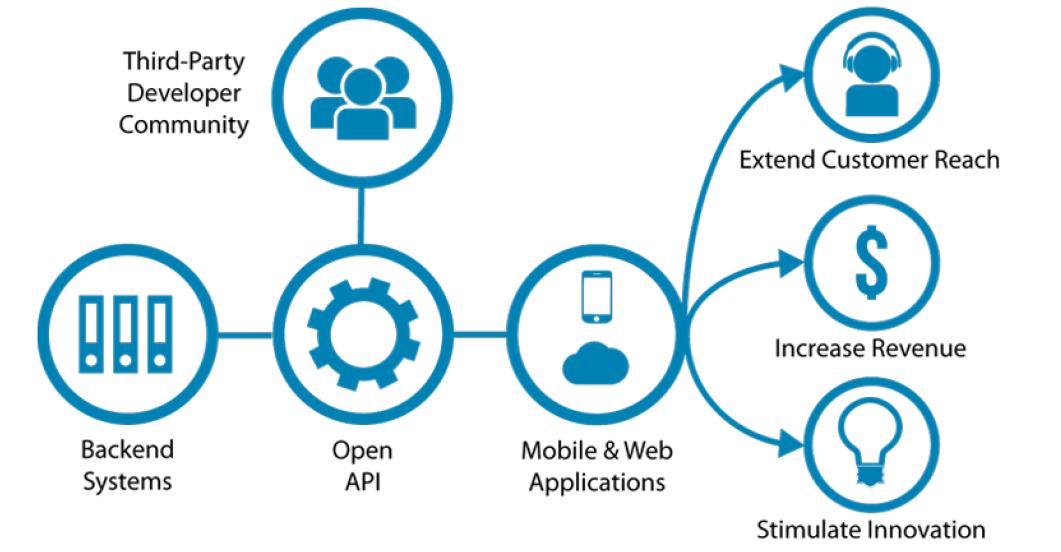
c. API designer have direct access to developers, making it easier to create dev-friendly interfaces

d. Exposing software interface always leads to the security and management challenges



* **External / Presentation APIs**

External APIs are the interfaces that are accessible by web and mobile developers. By opening the APIs to world, developers can take the API in creative and valuable direction of business.



## Cloudhub and On-premise Mule Instances

## API Manager and Core Mule Servers

## Core, Process and Experience APIs

# Integration Patterns and Approach

## List of Patterns

## Decision Matrix

# Continuous Integration and Deployment

## Workflow

## Code Versioning

# Monitoring

# Error Handling

# Security Features

## OAuth

## GDPR

# Documentation

## Architecture Documents

### Mulesoft

### SAP DS

## Design Documents

### High Level Design Document

### Pattern Specification Document

### Technical Design Document