DRAFT - IT Project

Design: API Integration Platform Considerations

Version:

0.1

## Purpose

To outline key considerations regarding platforms on which to develop integrations.

## Synopsis

<TODO>

## Contents

[Purpose 1](#_Toc194296932)

[Synopsis 1](#_Toc194296933)

[Contents 2](#_Toc194296934)

[Domains 3](#_Toc194296935)

[Domain & Elements 5](#_Toc194296936)

[Demonstration of Use Cases 8](#_Toc194296937)

[Model Query Examples 13](#_Toc194296938)

[Background 16](#_Toc194296939)

[Outcomes [Objectives] 16](#_Toc194296940)

[Options [Considered & Selected] 16](#_Toc194296941)

[Constraints 16](#_Toc194296942)

[Assumptions 16](#_Toc194296943)

[Dependencies 16](#_Toc194296944)

[Decisions 16](#_Toc194296945)

[Deliverables/Outputs 16](#_Toc194296946)

[Heading Level 3 16](#_Toc194296947)

[Heading Level 4 17](#_Toc194296948)

[Appendices 18](#_Toc194296949)

[Appendix A - Document Information 18](#_Toc194296950)

[Versions 18](#_Toc194296951)

[Images 18](#_Toc194296952)

[Tables 18](#_Toc194296953)

[References 18](#_Toc194296954)

[Review Distribution 18](#_Toc194296955)

[Audience 18](#_Toc194296956)

[Structure 18](#_Toc194296957)

[Diagrams 19](#_Toc194296958)

[Acronyms 19](#_Toc194296959)

[API 19](#_Toc194296960)

[GUI 19](#_Toc194296961)

[ICT 19](#_Toc194296962)

[IT 19](#_Toc194296963)

[UI 19](#_Toc194296964)

[Terms 19](#_Toc194296965)

[Application Programming Interface 19](#_Toc194296966)

[User 19](#_Toc194296967)

[User Interface 19](#_Toc194296968)

# Introduction

…

## Background

…

For example, Salesforce is inherently designed around customer relationship management, structuring a specific interaction between a person and a company. This makes it ill-suited for managing people in a broader sense, such as internal staff, external stakeholders, or individuals who interact with multiple organisations in different capacities. Using platform developers to force-fit an organisation’s needs into a CRM system designed for sales pipelines often results in significant compromises, leading to data model rigidity, costly workarounds, and functional misalignment. A similar issue arises when Microsoft Dynamics is chosen as a knowledge base platform. Dynamics is fundamentally a business applications suite, and while it can be extended to store documentation or articles, it lacks the fluid categorisation, semantic search, and version control features required for a true knowledge management system. Organisations that attempt to retrofit a structured, transactional system into an information-rich, evolving knowledge base will often find themselves fighting against its permissions model, UI constraints, and rigid data structures, leading to poor user experience, inefficiencies, and operational friction.

Data hub products and ETL platforms are also frequently misapplied as system solutions. Many ETL platforms were designed for historical, batch-oriented architectures, making them unsuitable for real-time, event-driven interactions. Attempting to repurpose an ETL tool as a dynamic integration hub often leads to inefficient, high-maintenance systems. A data hub is designed for real-time data exchange, with APIs enabling direct system interaction, while ETL platforms are designed for batch-oriented data migration. When data architects rather than system architects lead system design, they often default to building a data store rather than a dynamic data hub, resulting in stagnant, brittle integrations that fail to support real-time operations. The risk worsens when both flawed approaches are combined—buying an ETL-centric platform such as Informatica under the assumption that it will function as a data hub, then later attempting to wrap it with an interface. While Informatica is a powerful data integration tool, it is not designed as an API-first, event-driven data hub. Retrofitting an ETL pipeline into a real-time integration platform creates fundamental inefficiencies, introducing unnecessary latency, limiting system responsiveness due to the lack of real-time event handling, and bypassing validation, business logic, and security layers that would otherwise be enforced via APIs. When systems require dynamic, cross-organisational data exchange, an API-driven data hub—not an ETL pipeline masquerading as one—is the correct architectural choice. Organisations that fail to make this distinction end up with brittle, inefficient integrations that are expensive to maintain and difficult to scale.

Using platform developers to extend prebuilt systems may offer short-term gains but often locks organisations into rigid structures that constrain long-term adaptability. The risk is not just in choosing the wrong developer type but in assuming that all systems should conform to prebuilt solutions, rather than structuring solutions to fit the organisation’s actual current and evolving needs. An organisation should evaluate not just the feature set of a platform but its alignment with the organisation’s operational model.

Poor platform choices frequently result in overly rigid data models that require expensive workarounds, misaligned workflows that force users to adapt to the software rather than the other way around, technical debt from excessive customisation leading to high maintenance costs, and limited interoperability with external systems due to platform-imposed constraints.

Appendices

Appendix A - Document Information

Authors & Collaborators

* Sky Sigal, Solution Architect

### Versions

* 1. Initial Draft

### Images

[Figure 1: TODO Image 2](#_Toc144995112)

### Tables

[Table 1: TODO Table 3](#_Toc145048484)

[Table 2: TODO Table 2 3](#_Toc145048485)

### References

**There are no sources in the current document.**

### Review Distribution

The document was distributed for review as below:

|  |  |
| --- | --- |
| Identity | Notes |
|  |  |
|  |  |
|  |  |

### Audience

The document is technical in nature, but parts are expected to be read and/or validated by a non-technical audience.

### Structure

Where possible, the document structure is guided by either ISO-\* standards or best practice.

### Diagrams

Diagrams are developed for a wide audience. Unless specifically for a technical audience, where the use of industry standard diagram types (ArchiMate, UML, C4), is appropriate, diagrams are developed as simple “box & line” monochrome diagrams.

### Acronyms

API

: [Application Programming Interface](#Term_ApplicationProgrammingInterface).

GUI

: [Graphical User Interface](#Term_ApplicationProgrammingInterface). A form of [UI](#Acronym_UI).

ICT

: acronym for Information & Communication Technology, the domain of defining Information elements and using technology to automate their communication between entities. [IT](#Acronym_IT) is a subset of ICT.

IT

: acronym for Information, using Technology to automate and facilitate its management.

UI

: User Interface. Contrast with [API](#Acronym_API).

### Terms

Refer to the project’s Glossary.

Application Programming Interface

: an Interface provided for other systems to invoke (as opposed to User Interfaces).

User

: a human user of a system via its UIs.

User Interface

: a system interface intended for use by system users. Most computer system UIs are Graphics User Interfaces ([GUI](#Acronym_GUI)) or Text/Console User Interfaces (TUI).