|  |  |
| --- | --- |
|  | Level Set  [**Overview**](#_9ukxg63y6qep)  [What is a Reference Architecture?](#_eh89hmjydu2y)  [Benefits of a standards-based architecture](#_653dd6h5vcpx)  [Architecture Components](#_gzymfgvg6yuj)  [Architecture Principles](#_p7et68f2fh1r)  [Why use a Reference Architecture?](#_3ufocgxtmd6b)  [Who uses a Reference Architecture?](#_3oqr1bz8zad4)  [**TELUS API Reference Architecture**](#_wtqedss0mfl2)  [**Key Definitions**](#_aakhfu4g9ryd)  [Definitions of Architectures – Baseline, Target and Transition Architectures](#_2et92p0)  [**Guiding Principles**](#_2rw3478v1z4r) |

# Overview

## What is a Reference Architecture?

A reference architecture does not define an enterprise’s complete architecture itself but, instead, is a model or template used by internal and external stakeholders to plan and design an architecture.

Just as the City of Calgary’s [Guidebook for Great Communities](https://www.calgary.ca/pda/pd/current-studies-and-ongoing-activities/guidebook-for-great-communities.html) isn’t itself an urban plan, it supports those who do plan cities and neighbourhoods by breaking down the planning steps, and detailing overarching principles and policies.

A reference architecture is usually based on an industry-developed standard, which provides a consistent language so that all stakeholders can more easily understand the architecture. The closer an enterprise architecture is to standards the greater the benefits.

### Benefits of a standards-based architecture

* higher performance
* shorter development cycles
* lower cost of integration
* reduced overall cost of ownership
* creates a market of customers and partners that can easily adopt their products
* supports data accuracy, security, privacy, and traceability

### Architecture Components

A reference architecture consists of a number of components which contribute to the overall vision of the target architecture.

The architecture vision defines how the architecture addresses the needs of the enterprise. Architecture principles govern architecture and define how it operates. Functional architecture lays out loose grouping of components. Definition of the components decomposes the functional groupings and defines their properties.

### Architecture Principles

A reference architecture applies architecture principles. Architecture principles govern the architecture and define how it operates. Principles can be collected into the following groupings:

* Overarching
* Business
* Information and Data
* Application
* Technology

## Why use a Reference Architecture?

Without a Reference Architecture, there is no way to accurately validate that individual architecture plans will meet the enterprise’s needs.

It provides the overall framework and guidelines to ensure that, when architectures are planned, they will include the necessary components given their role in the overall enterprise, and consider enterprise obligations.

By providing practical patterns, it jumpstarts the development of architectures.

## Who uses a Reference Architecture?

* **Enterprise Architecture:** to govern the architecture. Used to assess and measure the solutions against the reference architecture principles and patterns
* **Domain Architects**: to guide the creation and maintenance of their domain and api architecture roadmaps.
* **Executives:** to ensure the strategy and guiding principles align with the goals and desired outcomes of the enterprise. To understand the target architecture and steps required to transition to the target architecture.
* **Development Teams:** to reference and apply the api reference architecture patterns and ensure they are developing APIs which conform to the architecture, security and reliability required by the enterprise.

# TELUS API Reference Architecture

The TELUS API Reference Architecture is based on architecture principles.

This suite of documents describes the TELUS API Reference Architectures. The API Reference Architecture consists of the principles, patterns as well as a target and a number of transitional reference architectures. The reference architectures apply the architecture principles and patterns to provide a blueprint for the transformation of TELUS’ integration architecture from a traditional monolithic architecture towards a modern architecture meeting the demands of a digital economy.

APIs are a key foundation of digital transformation: they enable mobile apps, create integrated digital ecosystems across customers and partners, allow firms to benefit from the innovations of digital disruptors, and drive operational excellence. Done right, APIs create business agility that fosters rapid business reconfiguration necessary to adapting to constant change. Change is driven by the need to revamp customer experience, address regulatory challenges, respond to new and changing competition, and adapt to unpredictable scenarios. API management solutions are central to managing the relationships between API providers and API users; as such, they are business applications that are critical to digital business success. APIs are a way to gain or provide access to some type of business asset.

With the continued acceleration of the digital marketplace, the emergence of the API economy and cloud technologies, Telcos must evolve to be digital service providers and digital service enablers to compete in the new, hyper-connected world. The new generation of digital service providers needs to be able to plug in partners and immediately create products to meet consumer demands and stay competitive.

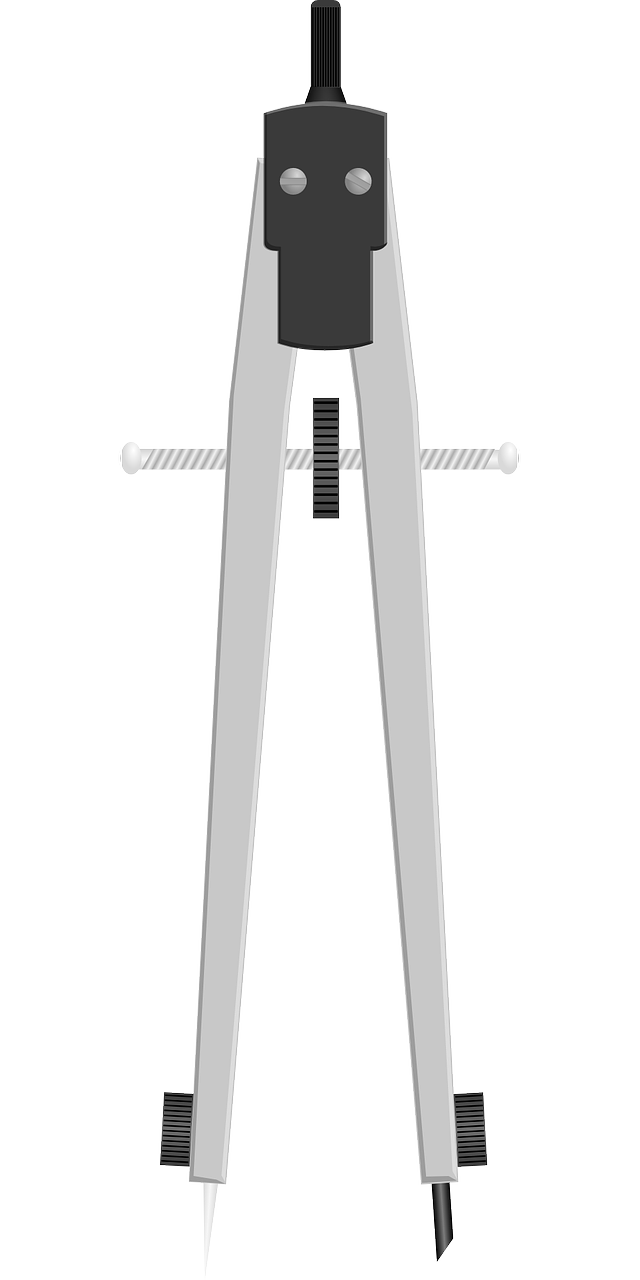
Consumers now expect that there will be services available for everything, including cloud, NFV, IoT and traditional access components. In the future all things physical and digital will be bought and sold through APIs.

The API Reference Architecture provides the blueprint for the target architecture. It is a playbook containing the architecture concepts, architecture principles, patterns, best practices and models to apply when creating transitional and target API architectures. Solution Architects use the reference architecture to guide, assess and measure how each solution architecture contributes towards the realization of the target architecture within the architecture portfolios.

## TELUS Enterprise Outcomes

The API reference architecture document provides a blueprint for an architecture transformation required to meet the business drivers and future vision:

* Customers First
* Factor of 10 x change
* Real Time
* Over The Top
* Daily Innovation
* Reliability / Resiliency
* Interoperability
* Cloud at Scale
* Security By Design
* Business Outcomes; Increased revenue, new revenue streams



# Key Definitionshttps://lh5.googleusercontent.com/8Y10MYdIKDVYA4uyp5oSUAO9IxbWfsrRg3iiKsxgyrVgr73_ipQuVH3Y0xSufIHV4GOrgPeXWCvGLZA8nQMCGHQeeT7gQNRrqWRGVS7AT8x4E4l8quGjigSXtQ3uUrN6ifPDA6bK01gQErhh9Q

## Definitions of Architectures – Baseline, Target and Transition Architectures

The ***baseline*** architecture is the current, as-is, architecture

The ***target*** architecture is the future architecture and relies on the aspect of time to anchor the vision of the target.

***Transition*** architectures represent interim states of the overall architecture as it progresses towards the target.

The API Reference Architecture defines the requirements, principles, and constraints of the target architecture. The architecture roadmap demonstrates the applications of the patterns and principles of the API Reference Architecture presented as a blueprint for transitional architectures. The target architecture is used to govern, assess and measure the architecture of a domain and portfolio.

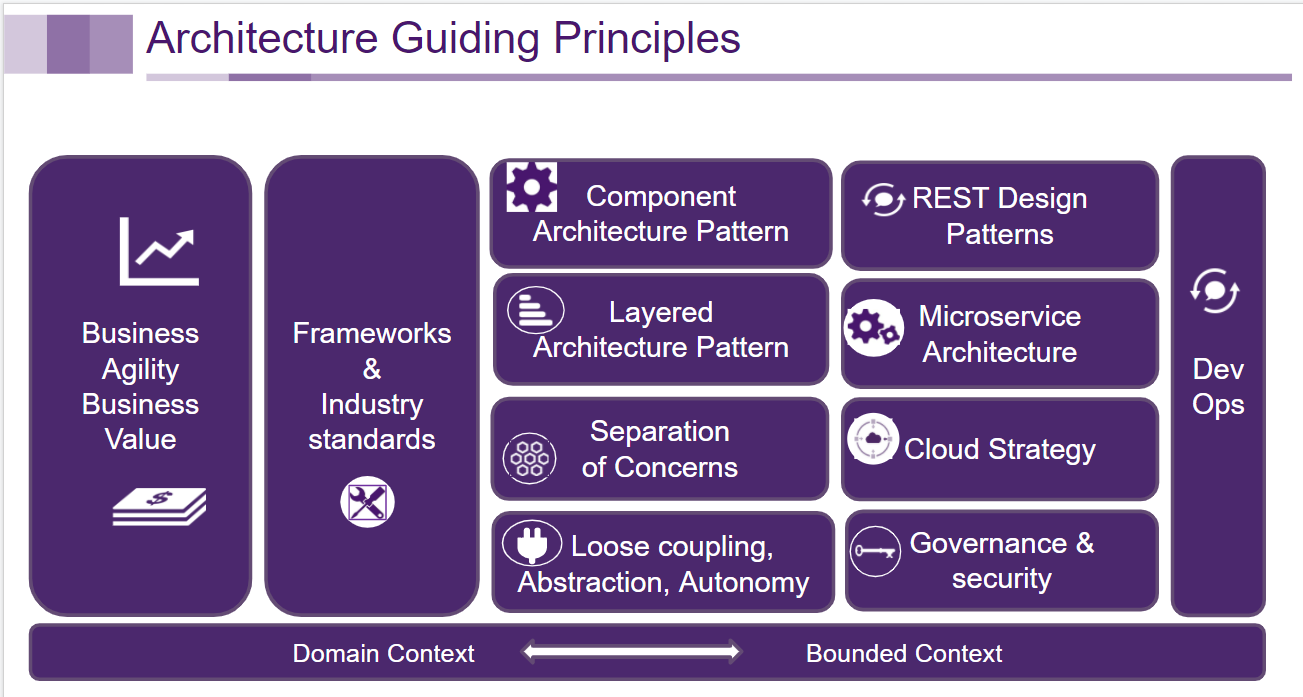
Solution architectures, architecture plateaus which reflect an architecture specific to a project or solution use the API Reference Architecture to govern, assess and measure the solution in relation to the desired target architecture. A solution architecture is a transitional architecture within the domains and portfolios.

Solution Architectures use the architecture reference as a blueprint to realize each transitional architecture within the architecture roadmap and move towards the target architecture.

The model demonstrates the relationship between the Enterprise drivers, desired enterprise outcomes, architecture principles, standards, patterns, and architecture plateaus realized by the solution architectures. The model depicts the API Reference Architecture as supporting the business driven objectives and drivers of the Enterprise.

# Guiding Principles

The API Reference Architecture adheres to a number of guiding principles.



1. **Business Agility** – represents the responsiveness of an enterprise. The goal of the API reference architecture is to increase business agility
2. **Business Value** – One goal of architecture is to increase the value of the enterprise and the return on investment. ROI is the tangible value and cost-savings that something provides compared to the cost of producing and governing it
3. **Business Enablement** - Zero touch IT. Provide capabilities through APIs to enable the business to create innovative solutions.
4. **Domain Context** - Provide a clear and concise context and scope of responsibility for domains. Domains will encompass the systems, data, integrations and resources which have the responsibility of providing the capabilities as described by the Domain Context.
5. **Bounded Context** - each domain context and bounded context will consist of a conceptual model that is described in its own ubiquitous language
6. **System and data abstraction** – hide the logic and implementation of a system and/or data. Enables the option to decompose and move system components to the target architecture. Abstracts and decouples the direct dependency on any specific vendor technologies and products.
7. **Separation of Concerns** – Separates APIs into clearly defined context and contextual layers each with its own concern. Creates autonomous components. Reduces development time and duplication, integration time and increases reuse.
8. **Loose Coupling** – The principle of maintaining a relationship that minimizes dependencies and only requires that they retain an awareness of each other
9. **Layered Architecture** - Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role. The API Architecture defines five categories of APIs; Experience, Process, Entity, System Wrapper and System. The characteristics of each category relates each API to a specific layer of the architecture. The interaction between components within each architecture layer is clearly defined and governed
10. **Stateless** - each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server.
11. **Uniform Interface** – this constraint is fundamental to the design of any REST service. The uniform interface simplifies and decouples the architecture, which enables each part to evolve independently. The four guiding principles of this interface are:

* **Identification of resources** - Individual resources are identified in requests, for example using URIs in web-based REST systems. The resources themselves are conceptually separate from the representations that are returned to the client. For example, the server may send data from its database as HTML, XML or JSON, none of which are the server's internal representation, and it is the same one resource regardless.
* **Manipulation of resources through these representations** - When a client holds a representation of a resource, including any metadata attached, it has enough information to modify or delete the resource.
* **Self-descriptive messages** - Each message includes enough information to describe how to process the message. For example, the parser to invoke may be specified by an Internet media type (previously known as a MIME type). Responses also explicitly indicate their cacheability.
* **Hypermedia as the engine of application state** - Clients make state transitions only through actions that are dynamically identified within hypermedia by the server (e.g.by hyperlinks within hypertext). Except for simple fixed entry points to the application, a client does not assume that any particular action is available for any particular resources beyond those described in representations – Note this principle is *not* mandatory at TELUS.

1. **Abstractions** - The framework and reference architecture endeavor to describe service provider operations in an implementation-agnostic manner. Abstractions are defined utilizing conceptual and logical abstractions.
2. **Adoptions of standards and best practices**- Standards and best practices are utilized and applied as intended, minimizing customization