May 2019

**Enterprise Integration Architecture & Data Services**

Standards & Shared Practices Document

**Service Offering:** Technical Services for Developers 🡪 Integration Services 🡪 Data Services and API’s

Document Control Information

Document Information

|  |  |
| --- | --- |
| Document Identification | n/a |
| Document Name | Enterprise Integration Architecture and Data Services Standards |
| Project Name | N/A |
| Client | Yale University |
| Document Author | Darrell Cook |
| Document Version | V1.7 |
| Document Status | Review State |
| Date Released |  |

Document Edit History

| Version | Date | Additions/Modifications | Prepared/Revised by |
| --- | --- | --- | --- |
| 1 | October 2018 | Initial Draft | Darrell Cook |
| 1.1 | November 2018 | Revision based on Technology Architecture Standards Team Feedback | Darrell Cook |
| 1.2 | January 2019 | Substantial updates, deletions, additions across the document | Darrell Cook |
| 1.3 | February 2019 | Additional content added as well as incorporating feedback received from TAST (Lisa Sawin / Louis King) | Darrell Cook |
| 1.4 | March 2019 | Additional updates added and incorporated from feedback and modifications from Christine Viele | Darrell Cook / Christine Viele |
| 1.5 | March 2019 | Modifications and additions based on feedback from several members in the Distribution of Final Document Group | Darrell Cook |
| 1.6 | April 2019 | Modifications based on feedback from Final Document Group and others. Incorporated new sections such as 4.4.1, 5.6 and reference to other Talend Standards. Incorporated feedback from Gartner analyst after 4/11 meeting | Darrell Cook |
| 1.7 | May 2019 | Modifications to Integration Principles from Louis King | Darrell Cook |

Document Review/Approval History

| Date | Name | Organization/Title | Comments |
| --- | --- | --- | --- |
| 10/11/2018 & 2/2019 | Shane Anderson | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Darrell Cook | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Lou Tiseo | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Louis King | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Lisa Sawin | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Andy Newman | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Morrow Long | Yale University | Initial Draft Review |
| 10/11/2018 & 2/2019 | Camden Fisher | Yale University | Initial Draft Review |
| 03/21/2019 | All Individuals listed in “Distribution of Final Document Below” | Yale University | Working copy review |
| 4/11/2019 | Massimo Pezzini | Gartner Research Engagement Services | Working copy review |
| May 2019 | TAC Ratification | Yale University | Final Document |

Distribution of Final Document

The following people have been identified as resources who are involved or play a role in this space. They will have the ability to review and provide changes and / or modifications to these standards and shared practices before being one of the many recipients of the final version of this document.

| Name | Organization/Title |
| --- | --- |
| Oren Kanner | CCT – UX & Digital Strategy |
| Tom Undercuffler | BSG - RE |
| Vidhya Narayanan | BSG - YAA |
| Marissa Kitz | BSG – Data Services |
| Kazi Ahmed | BSG – Data Services |
| Paul Gruhn | BSG – Data Services |
| Christine Viele | BSG – Data Services |
| Vijayshree Erodula | CCT– Academic Administration |
| Tim Hinckley | CCT – Academic Administration |
| Franz Hartl | CCT – Web Technologies |
| Suresh Pandian | Platform and System Administration |
| Vincent LaBonia | Platform Services |

Table of Contents

[1 Introduction and Enterprise Integration and Data Service Standards Overview 5](#_Toc5701062)

[1.1 Governance & Compliance 5](#_Toc5701063)

[1.2 Inputs 6](#_Toc5701064)

[2 Integration and Architecture Scope 7](#_Toc5701065)

[2.1 Scope 7](#_Toc5701066)

[2.2 Engagement 7](#_Toc5701067)

[2.3 Assumptions 7](#_Toc5701068)

[2.4 Constraints and Limitations 7](#_Toc5701069)

[2.5 Dependencies 8](#_Toc5701070)

[2.6 Risks 9](#_Toc5701071)

[3 Guiding Principles 10](#_Toc5701072)

[3.1 Enterprise Specific Guiding Principles 10](#_Toc5701073)

[3.2 Integration Specific Guiding Principles 10](#_Toc5701074)

[3.3 ETL Specific Guiding Principles 13](#_Toc5701078)

[4 Integration Platforms (ESB) and Tools 15](#_Toc5701079)

[4.1 webMethods 15](#_Toc5701080)

[4.2 Workato 16](#_Toc5701082)

[4.3 Talend 17](#_Toc5701088)

[4.4 API Management 18](#_Toc5701089)

[4.4.1 When to use the API Gateway 19](#_Toc5701090)

[4.5 Web Service (XML) Editors 20](#_Toc5701091)

[4.6 Managed File Transfer 20](#_Toc5701092)

[5 SOA Services (Service Development) 21](#_Toc5701096)

[5.1 Overview 21](#_Toc5701097)

[5.2 Service Contract 21](#_Toc5701098)

[5.3 Service Consumer Usage Plans 21](#_Toc5701099)

[5.4 Web Service Security Overview 22](#_Toc5701100)

[5.5 Service Versioning Strategy Overview 23](#_Toc5701107)

[5.6 Error Handling 24](#_Toc5701115)

[5.7 Service Discoverability 24](#_Toc5701117)

[6 ETL 26](#_Toc5701120)

[6.1 Overview 26](#_Toc5701121)

[7 Integration Architecture Decision Trees 27](#_Toc5701122)

[7.1 Overview 27](#_Toc5701123)

[7.2 ESB Integration Platform Decision Tree 27](#_Toc5701124)

[7.3 Web Service Communication Protocols Decision Tree 28](#_Toc5701125)8

# Introduction and Enterprise Integration and Data Service Standards Overview

This document supports Yale’s Operational Excellence initiative by establishing technology standards to support consistent, repeatable, secure, and reliable service delivery across ITS. This document specifically has a three-fold purpose:

1. To help solution designers, application developers, and project teams easily engage with the ITS Integration and Data Services team and to better understand the rigor and structure that defines its standards and practices in this space. This includes platforms and tools, decisions trees guiding principles, strategies employed, and the governance processes and structures that exist.
2. To help ITS management and technology archites better understand the structures and standards associated with Enterprise Integration Architecture (i.e. ESB, ETL, integration tools, etc) and data services. It will provide the guiding principles, frameworks, and **best practices** needed to make informed decisions and to help support any intended future use, in regard to development and use of enterprise integrations and data services.
3. To help application developers and project teams that wish to leverage their own resources in the integration and data services space. The intention is that this document will help to provide some of the principles, standards, and best/shared practices that should be considered in their development and implementation of solutions. Consideration should be given to also look at leveraging the tool sets and strategic platforms available, as well as the use of any decision trees that may help in the overall architecture of their solutions.

This document will be maintained over time, to keep up with the ever changing technology landscape and to ensure the latest and greatest information is included and available. In addition, improvements recommendations by readers and users of these standards and best practices will be evaluated and incorporated into future updates. Please send comments to [ITS.Data.Services@Yale.edu](mailto:ITS.Data.Services@Yale.edu).

## Governance & Compliance

The awareness of these standards and shared practices along with the adherence and adoption of these standards will be governed and administered through a variety of channels.

**TAC:** The Technology Architecture Committee (TAC) will help to ensure that there is awareness and visibility of these standards for any future projects and initiatives that come through the ITS Gating Process. They will assist in establishing the connection between the projects and groups like the ICC (see below). Additional information can be found on the confluence site: <https://yaleits.atlassian.net/wiki/spaces/STAN/pages/803668001/TAC>

**ICC:** The Integration Competency Center (ICC) is the central governing body for data services and integration platforms and tools. The ICC will also help to ensure visibility and adoption of these standards. The ICC is available for enablement and support for application teams. They will work with all areas to either support or help take on development and support activities (if able). Additional info can be found on Yale’s developer portal: <https://developers.yale.edu/> or <https://yaleits.atlassian.net/wiki/spaces/BASP/overview>

**PMO:** The Project Management Office (PMO) will also have an active role in understanding what projects may need engagement during the project portfolio process. They can in turn help to provide awareness and use of services in addition to how they should be engaging with the appropriate governing bodies through the ITS Gating process.

**ISO:** Yale’s ITS Information Security Office (ISO) will continue to be another mechanism to help ensure adherence to predefined standards through processes such as the SDR.

**TAST:** The Technology Architecture Standards Team (TAST) is a committee that has been initiated and formed under the current CIO to look across our technology landscape and to help in identifying areas for standards and shared practices. This document, along with many others, has been discussed and reviewed with this group. Additional information can be found on the confluence site:

<https://yaleits.atlassian.net/wiki/spaces/STAN/pages/803700826/TAST>

## Inputs

Throughout this document there are references, links, or attachments included to many other artifacts, templates and repositories created prior to the development of this document. The artifacts that are noted in this section will not be found or referenced anywhere else throughout the document, but may provide useful information if needed.

* **Workday@Yale Technology Workstream Integration Strategy**

This document was created during the Vision/Plan phases of the Workday project with focus on system integration across Yale’s application ecosystem and established the principles, frameworks and best practices in the Workday integrations space.

* **ICC Strategy Story**

This document provides an overview into the ICC. Specifically, it covers what the ICC is, what the benefits and advantages of an ICC are, who comprises the ICC, what services the ICC offers and what role the ICC can play in helping other areas and teams.

* **Data Services Confluence Site**

|  |  |
| --- | --- |
| Document | Document Location |
| Workday@Yale Technology Workstream Integration Strategy | [Workday @Yale](https://yaleedu.sharepoint.com/sites/pmo/ERP01/Project%20Documents/03.%20Technology/Strategy/Workday%40Yale%20Integration%20Strategy.docx?d=wb60f2b2049d24b6d8bb22ed46f681933) |
| ICC Strategy Story |  |
| BSG Application Services Public Confluence Site | <https://yaleits.atlassian.net/wiki/spaces/BASP/overview> |
| Enterprise Guiding Principles (Confluence Site) | <https://yaleits.atlassian.net/wiki/spaces/STAN/pages/781156666/Architecture+Principles+DRAFT> |

This site provides a great deal of information relating to the platforms and their respective standards. In addition, information on enterprise services such as service contracts, how to test services and things like service version notes can also be found here. Information on the process and required forms for requesting access to data in the PeopleHub and COA (Chart of Accounts) are also available here.

* **Enterprise Guiding Principles**

Several guiding principles are noted in this document and are specific to the integration and ETL space. There are however, many guiding principles that span across ITS and are not specific to any one area or service. These enterprise guiding principles can be accessed through the link in the table below.

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs team.*

# Integration and Architecture Scope

## Scope

This document includes inputs from many other sources and will be referenced accordingly. The scope of this document includes all areas of Enterprise Integration Architecture and Data Services. More specifically, this document covers areas such as the integration platforms and tools that are in use here at Yale. In addition, this document references information and other locations for ETL standards and shared practices. There is also an emphasis around the use of enterprise data services (SOA (Service Oriented Architecture) Service Design) as well as information specific to the application developer portal for consumption and development of APIs. Some of the deliverables that are included in this document are:

* Current state infrastructure / landscape diagrams
* Decision trees
* References to other informational resources (i.e. Confluence, SharePoint, ETL, etc.)

## Engagement

Links and attachments are referenced throughout this document. The ICC / Data Services and APIs Team can and should be engaged to support any of the following:

* To simply help talking through any questions or concerns
* To offer platform / service support and development guidance
* To help design, build or test services (w/ potential enterprise applicability)
* To initiate access and document consumer SLAs in consuming existing services or APIs
* To assist in any future integration and platform tool analysis and evaluations

Engaging with the ICC / Data Services and APIs Team is simple and easy. Tickets should be submitted to the Data Services and APIs team through the ServiceNow portal using the following service offering path: Technical Services for Developers 🡪 Integration Services🡪 Data Services and API’s. Additionally, any questions or support needs can also be directed to the [ITS.Data.Services@Yale.Edu](mailto:ITS.Data.Services@Yale.Edu) Outlook group.

## Assumptions

The Enterprise Integration Architecture and Data Services makes the following assumptions:

* Existing strategic platforms are to be leveraged where and whenever possible. Refer to section 3 Guiding Principles in this document (example: Integration Principle #6). New integration platforms or tools not specified in this document will be considered for inclusion in the standard where appropriate business rationale exists, as service introduction planning is complete and when necessary approvals (i.e. TAC, ICC, ISO, etc.) are secured.
* Decision trees have been provided at the end of this document. To help provide guidance in making informed enterprise integration and data service decisions. The data services team is available to assist further (see section 2.2 Engagement).
* The service seeks to reuse components where possible and leverage existing tools, skills and experience in this space in order to optimize the business value to Yale

## Constraints and Limitations

There may be scenarios or use cases that may fall outside of the current standard. In those situations, teams should still collaborate with the service owners to identify the best possible solution(s). Evaluation of exceptions are made on a case-by-case basis in consultation with the ICC and/or the TAC for approval and documentation. Decisions regarding integration choices may be constrained by the following factors:

* Application limitations (i.e. inability to support web services)
* Service limitations (i.e. volumes and performance requirements)
* Platform or tool dependencies

## Dependencies

Much of the work outlined in this document is specific to the following service / service offering:

* Integration Services > DATA SERVICES & APIS

There are a number of additional services / service offerings that support the service mentioned above. They include:

* PLATFORM SERVICES > ENTERPRISE PLATFORM ADMINISTRATION
* INSTITUITIONAL REPORTING AND ANALYTICS > BUSINESS INTELLIGENCE > DATA MANAGEMENT

The Data Services and APIs service offering has dependencies on several other service offerings and other applications. Application / System areas leveraging the work of this technology space include many of the following:

|  |  |
| --- | --- |
| Yale College Website | Egencia for Business (Expedia) |
| Infor (ILM) | eHisto |
| Teaching Assignment Management System (TAMS) | HAUS |
| Papercut ID Sync | BMS Common Look up tables/ Functions (Com\_People - YSM) |
| Yale Alert | Voyager |
| Canvas | Clinical Effort Reporting |
| System Center Configuration Manager (SCCM) | ITG Hub (T32 Grant Proposal System) |
| GSPS | EHS Integrator |
| Spinup | Pinnacle |
| GetRoles Service | ServiceNow |
| OrgSync (Campus Labs) | Banner General |
| Access Control - Symmetry Security System (from AMAG) | Facilities (FET, FWR, FAMIS) |
| Yale Affiliated Hospital Physicians Lecture Scheduling (YAHP) | Message III |
| Reservations | CardSmith |
| Laboratory Order tracking & Instrument Scheduling (LOTIS) | Council of Heads of College (COH) formally known as Yale College Council of Masters (Fellows Database) |
| TapRide (Formerly Trapeze) | Yale College Standing Committees (YCSC) |
| MSCLAV Media Services Classroom AV | YUAG Photo Gallery |

## Risks

* Application areas that implement non-standard integration and data solutions inhibit operating at scale and add to Yale’s technical debt while also incurring additional costs. Therefore, exceptions are considered under the IT to Business Alignment Principle to ensure the best outcome for Yale.
* Lack of awareness regarding the standards and shared practices as well as not employing governance procedures (or adherence to standards) or shared practices for integration development.

# Guiding Principles

The following guiding principles are used to inform the decision-making process and management of Enterprise Integration Architecture and Data Services. These principles are meant to guide Yale in using best practices for implementation as well as ensure the integration methodology aligns with the strategic objectives. They aid in determining the appropriate integration type during the analysis phase as well as influence design choices during the build phase. There may be use cases that could bring some guiding principles into conflict. If so, the integrations team will attempt to choose the solution that provides the most overall value to the project. This value could be realized in efficiencies in any of the areas of time, quality, cost or Total Cost of Ownership (TCO). This may also lead to valid exceptions to these guidelines. Any exceptions will be documented and will trigger a review of the decision framework. Please refer to section 2.4 Constraints and Limitations.

## Enterprise Specific Guiding Principles

ITS has ratified a number of enterprise guiding principles. These principles are referenced throughout many technology standards being developed here at Yale and apply to all ITS standards and technology architectures.

* ITS to Business Alignment
* Business Continuity
* Compliance with Policies and Standards
* Interoperability
* Information is Secure
* Information is Shared

|  |  |
| --- | --- |
| Document | Document Location |
| Enterprise Guiding Principles (Confluence Site) | <https://yaleits.atlassian.net/wiki/spaces/STAN/pages/781156666/Architecture+Principles+DRAFT> |

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs Team.*

## Integration Specific Guiding Principles

## Integration Principle 1: Integrations are aligned to business requirement needs

**Description:** Integrations must be based on business requirements in order to ensure business alignment.

**Rationale**: Integration and data services maintain a high number of integrations. Each integration has a set of requirements in regard to change management, quality, and performance. Ensuring alignment to business requirements makes sure that integration and data services runs efficiently and that investments in the service are valued by the business.

**Implications:** Integration and data services must continuously change to meet the information needs of the business. Therefore, services must be identified through collaboration with the business and verified that they satisfy specific business requirements and goals.

**Integration Principle 2: Integrations are designed to be simple and reusable**

**Description**: Integration and data services will be most effective if they follow Yale’s enterprise architecture approach of building services with loosely-coupled, modular, and reusable components. Design choices should favor both simplicity and potential for re-use while simultaneously meeting all business and University requirements.

**Rationale**: Keeping integration design simple reduces both development time and ongoing maintenance costs. Utilizing reusable components reduces the overall costs for implementing and maintaining integrations. Modular components increase the systems' ability to adapt to different evolution needs by isolating the impact to fewer affected modules.

**Integration Principle 3 – Integrations are secure**

**Description:** It is vital that the University’s information is adequately protected. Sensitive data ***must*** be handled securely at all times. For more information concerning Yale’s Security and Confidentiality requirements go to: <https://your.yale.edu/policies-procedures>

**Rationale:** The use of appropriate and production-grade security methods will ensure that Yale is simulating true production conditions and is therefore both testing the efficacy of security measures as well as extending those measures to the test environment. Integrations should only have access to the data and web services required to perform their function.

**Implications:**  Sensitive files deposited on file systems for file transfer should be encrypted at rest. In addition, minimally, integrations that perform file transfers, whether test or production, should use SFTP with PGP encryption.

**Integration Principle 4: Integrations are loosely-coupled  
Description**: Interfaces have low coupling, are self-described and offer low impact in case of changes.

**Rationale**: Low-coupling interfaces are preferable, because when interfaces between independent applications are highly coupled, they are less generic and more susceptible to causing unwanted, secondary effects when they are changed.

**Implications:** Low coupling means that the services (APIs, for example) are conceived with no affinity to a certain service consumer. Therefore, the service is completely uncoupled to a service consumer. However, the service consumer is dependent of the service (that is, contains references for service interfaces). The service is also responsible for exception treatment. The result is a low-coupling architecture.

**Integration Principle 5: Integrations are standards-based**

**Description:** Open standards and methodologies should be adopted in preference to proprietary solutions for integration.

**Rationale:** Data delivered through interfaces is crucial to the success of the University. The use of standards extends further than interoperability. The use of open standards shields against supplier dependencies and is an important component in protecting IT investments. A move to greater reuse and componentization relies on standardization.

For example, the use of XSLT for transformations will allow Yale to develop the transforms independently of the integration’s tools, and will promote platform independence, reuse and componentization. Workday has standardized on XSLT, and while they have fielded capabilities that leverage XSLT through the use of proprietary tools, straight XSLT, where practicable, will be preferred over proprietary tools.

**Integration Principle 6: Service-Oriented Architecture is preferred**

**Description:** **Where feasible, web services should be used and consumed to meet project data integrations requirements**; core functionality in all systems should be exposed through services and accessed through a single common integration platform.**Rationale:** Yale is committed to a Service-oriented Architecture and has several foundational services available today. To continue to move in that direction, a bias towards use of SOA should be encouraged. Services require a service to consume and a consumer. Where target systems do not have a native ability to use services, every attempt should be made to use one of Yale’s Strategic Integration Platform components to consume data and transport it to the application in whatever manner is required. Where a service is not already available, a discussion should be had with the Integration Competency Center (ICC) to determine if the project data integration requirements can be met in the project timeline through the creation of an appropriate service.

**Integration Principle 7: Integrations utilized packaged solutions and pre-built components wherever possible  
Rationale:** Vendor-packaged integrations allow for efficient cloud-to-cloud and cloud-to-off-premises integrations that are written, housed, supported, maintained and enhanced by the vendor. These processes represent best practices and standardized business processes.

**Implications:** A packaged integration should be used when available and meets Yale requirements. Where third-party vendors do not currently have packaged integrations, Yale should consult with the vendor to determine whether a packaged solution or connector is on the roadmap.

Yale has declared certain platforms as strategic platforms when combined with other platforms. In the case of Workday, Force.com has been declared a strategic platform. This means that not only should Yale consider using this platform when additional functionality needs to be built, but that the platform tools are also available for use.

**Integration Principle 8: Integrations must be built with performance considerations in mind along with logging and monitoring**

**Rationale~~:~~** It is costly to add scalability as an afterthought. Integrations systems need to maintain efficiency and service levels regardless of demand. Most integrations will grow in data volume over time. Architectures must support increasing numbers of users, source/target systems, transaction volumes and data capacity. Integrations that today are simple may become complex in the future based on this growth. Growth and performance must therefore be considered when designing interfaces. Quality of Service (QoS) should be managed using centralized tools provided by the integration platform for monitoring. Logging should also be performed. Where possible, proactive monitoring along with appropriate alerts and notifications should be established and put in place. Metrics and traceability around integrations should be implemented that provides the necessary visibility on who is calling the integrations and the volumes for which they are calling them.

**Implications:** Constraints and considerations must be observed when designing integrations and must be taken into account when deciding which technology to use. If a file today is near a threshold limit, and a larger file would require the use of a more advanced technology, it is likely that this consideration should drive that level of adoption.

**Integration Principle 9: Design choices should take into account constraints or limitations of the technologies**

**Description:** **The best technology given the probability of growth should be selected from the onset.Rationale:** It does not serve Yale’s best interests to use tools and techniques in a manner that is inconsistent with ongoing supportability and best practices.

For example, Yale should not write custom Java code within the Workday Studio tool. While this is possible it is discouraged by Workday and it will result in a model wherein Yale must take more responsibility for the support of the solution platform.

Integration Principle 10: Integrations are managed and should be published and discoverable (Discoverability)

**Description:** Integrations *should* be managed through project deliverables and templates (may require addition to various templates and deliverables outlined by the PMO. Integrations *must* be documented. Integrations should be entered in an enterprise repository and should be published in a manner that allows users and consumers to find and identify any and all integrations that they can potentially reuse or from which at a minimum they can pattern their integration.

**Rationale:** The benefits of this repository will be visibility of all integrations, an ability to perform better, more precise impact analysis and the ability to be more agile when changes need to occur. Overall this will positively impact operational excellence.

**Implications:** A process for regular updates will need to be provided or the data will go out of date. A suggested proposal is that developers simply update records when integrations are touched, modified, deprecated or removed.

Integration Principle 11: Integrations are tested (Testing)

**Description:** All systems integrating with an application should maintain separate test environments for integrations testing.

**Rationale:** Isolated test environments for integrations testing will greatly reduce the time spent tracking potential data problems. This principle is subject to feasibility and the availability of development/test instances and should not be understood as requiring additional infrastructure.

Integration Principle 12: Integrations should support flexibility and adaptability (Agility)

**Description:**

**Rationale:** Solutions must be flexible and be designed to absorb changes in requirements, processes and organizations. An important facet of flexibility is modularity and reuse.

## ETL Specific Guiding Principles

**ETL Principle 1: Data should be sourced from system(s) of record (Authoritative Systems)**

**Description:** Where the integrity of the data is vital, a data element must either be linked to, or extracted directly from its system of record.

**Rationale:** This principle is intended as guidance. There may be situations where for valid technology and/or business reasons, the data cannot be sourced from a system of record. It should be noted that the fact that an existing integration may source data from a system that is not strictly a system of record should not be cause to require that the existing integration be rewritten. In the event that such a case is isolated, all competing considerations should be weighed. If an exception is merited, it is permissible. In the event that the exception represents a significant departure, it will be documented and reviewed by the appropriate governance body.

**ETL Principle 2: Process data close to the application**

**Description:** Data can be processed and transformed either on the source or on the target side of the equation. The latter is called ELT; the former is rarer, but can be done in scenarios where a large amount of data will be generated. An example of this might be a delta process that determines the data set to be transmitted prior to the ETL/ELT tool performing the extract.

**Rationale:** Delta processing: There is too much data to move every time the need to blend or transform arise. The logic is simple -- place agents where the data lives and process it locally. Carefully coordinated instructions should be sent to the agent, and the work done on the host platform before any data is moved. By taking the processing to where the data lives, you eliminate the bottleneck of the ETL server and decrease the movement of data across the network.

**ETL Principle 3: Performance considerations**

**Description:** In almost all cases, the prime concern of any ETL implementation is to migrate the data from source to target as quickly as possible. There is usually a load “window” specified as part of the non-functional requirements; a duration of time that is available to complete the process. The constraints are based on either the availability of the source system data, the need for the business to have access to the information, or a combination of both.

**ETL Principle 4: Simplicity and repeatability**

**Description:** As with all programming, a premium is placed on simplicity of design. This is in the interests of productivity of development time, consideration of ongoing maintenance, and a likely improvement in performance. The fewer steps involved, the less chance of mistakes being made, or places for things to go wrong. When changes need to be made, or fixes applied, the fewer touch points, the better. During the life of the processes, ownership will likely change hands. The system’s simplicity will aid clarity for those who need to take it on. One needs to be able to re-run jobs to achieve consistent and predictable results each time. This means it needs to be applicable to all relevant incoming sources, and in no way dependent on specific time parameters. If sources change, the process needs to handle those changes gracefully and consistently.

**ETL Principle 5: Extensibility**

**Description:** Rather than “bring everything” from a given source when a data migration process is first built, it should be possible to include only that which is identified as valuable to the business in the context of a given project or release cycle. Over time, additional data elements from the sources can be added to the ETL jobs, with potentially, new targets. The ETL job should take this iterative approach into account.

**ETL Principle 6: Subject-orientation**

**Description:** Workloads are to be divided into units based on business subject areas rather than source-system groupings or strictly target table structures. This recognizes that a given source table may contain information about more than one subject area (e.g., Customers and Accounts). In addition, a given subject area may be composed of multiple source tables, which may populate multiple targets. Simply limiting the ETL jobs to a single source and a single target may compromise the other principles, particularly performance and simplicity. Similarly, orienting the ETL jobs to either the source or target layouts may degrade the efficiency of the design.

# Integration Platforms (ESB) and Tools

There are a number of integration platforms and tools available and in use here at Yale, many of which have been around for some time and others that are fairly new. For those newer integration platform tools, extensive discussions and proof of concepts were conducted to ensure that the necessary evaluations and analysis was done in making the most appropriate selection. The following section highlights the platforms and tools being used in support of our enterprise integration architecture. There are also several references to where additional content and documentation can be found (Note: Some of the references may have access restrictions).

The platforms and tools that follow are what is being used and leveraged here at Yale today, and considered to be the strategic direction. They are explained in more detail in subsequent sections. These integration/service platforms and tools can be categorized in the following manner:

|  |  |
| --- | --- |
| **Platform / Tool** | **Capability** |
| webMethods | Complex & highly technical Integration / Orchestration Platforms |
| Talend | Complex & highly technical Integration / Orchestration Platforms |
| Workato | Non-Technical Integration Cloud Platform. To be used for fast, adaptive integration development. Works well with simple, point to point type integrations |
| Talend | ETL Tool(s) |
| Talend Runtime | ESB Runtime Tools(s |
| webMethods | ESB Runtime Tools(s |
| CA (Layer 7) API Management SaaS Portal | API Developer Portal |
| Oxygen | XML Tool / Editor(s) |
| Axway MFT (Managed File Transfer) | File Transfer Tools |

## webMethods

**webMethods:** (Software AG), is an ESB platform and an eclipse-based integration tool that is focused on application integration, business process integration and B2B partner integration. It is used primarily as a transport between internal (departmental) and external (vendor) data sources. For example, webMethods is often used to pick up files containing batches of transactions, and to deliver those files either to a staging file directory or to read and deliver the data into staging tables. The data is then often processed into the applications using the native technologies of PL/SQL and open interface tables and concurrent programs. webMethods is also used for outbound file transfers to internal/external target systems, and may extract data from EBS using JDBC connections or simply pick up and deliver existing files.

It has a strong Messaging feature - publish/subscribe – Universal Messaging/JMS - which is used in several integrations to de-couple dependency on backend systems, databases and processing. It is used in Procurement, Financial, Banking and Yale Health integrations where payload is required to be encrypted and signed/verified. It is also used to sweep large sets of transactions files from MFT and process these transactions into databases and/or cloud apps, such as, Workday, Salesforce, QuickBase etc via their web services APIs. It is also used in SOAP/REST web services, such as, COA Validator and as listener services for receiving and processing documents in real time via email. Integrations are built based on a lego block type, drag and drop style of declarative development.



**Figure 4.1A**: webMethods Landscape Architectural Diagram: Describes the use of webMethods here at Yale from consuming/invoking applications to back-end databases.

|  |  |
| --- | --- |
| Document | Document Location |
| webMethods Overview | [https://yaleits.atlassian.net/wiki/spaces/BSG/pages/755204129/Overview+of+webMethods](https://nam05.safelinks.protection.outlook.com/?url=https%3A%2F%2Fyaleits.atlassian.net%2Fwiki%2Fspaces%2FBSG%2Fpages%2F755204129%2FOverview%2Bof%2BwebMethods&data=02%7C01%7Cdarrell.cook%40yale.edu%7C8915340c8fcc4219953508d6a3412f8f%7Cdd8cbebb21394df8b4114e3e87abeb5c%7C0%7C0%7C636875897691460325&sdata=W72FXxGYH1G5tMVf3CNgB%2FOxWfSexC1j6CfAmiNtKz0%3D&reserved=0) |
| webMethods Developer’s Guidebook | <https://yaleedu.sharepoint.com/sites/departments-integrationservices/Shared%20Documents/webMethods/Training/Yale%20ESB%20webMethods%20Developer's%20Guidebook.docx?d=w688eb99d4b6b4132a9dc040bad12467d> |
| webMethods (Confluence Site) | <https://yaleits.atlassian.net/wiki/spaces/BSG/pages/52606391/WebMethods>  <https://yaleits.atlassian.net/wiki/spaces/BSG/pages/769130545/Integration> |

## webMethods Documentation Locations

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs team.*

## 

## Workato

Workato is a cloud-based solution (Integration Platform as a Service (iPaaS)) that helps businesses link their web applications easily using a very simple interface with drag and drop functionality. Usable by business users and other developers who are not “integration specialists” (i.e. ad-hoc integrators) Workato helps automate and integrate cloud apps and on-premise apps. There is no client tool to download and install to build the integrations. All that is needed is a browser. At present, Yale has purchased the Business Plus Plan which allows for 250,000 transactions per month and an unlimited number of recipes. Workato has also dozens of built-in connectors available. If you are interested in building integrations (recipes) or you would like to use Workato with a particular app or connector, then please engage the Data Services and APIs Team for further information.

**Workato Standards and Best Practices**

With Workato being a relatively new tool at Yale, the Data Services and APIs team has established some standards (structures and best practices) to which Workato developers here at Yale should look to adhere to. This will help in understanding it’s use and also help in supporting the volume associated with multiple use cases and areas using the tool. Some of the areas where standards and shared practices have been documented and being applied include the following:

**Folder Structures:** Workato does not offer any separate production and non-production instances to separate Workato artifacts, such as, connections, recipes, etc. Hence, Yale has developed some self-imposed separation of artifacts by using folders as follows:

### **Recipe Development and Naming:** In Workato, recipes are the executable objects that perform the integration steps and logic for a given integration. Recipes can be scheduled and/or run on-demand.  Development best practices have been established for areas such as:

1. **Use of comments -** Clear and detailed comments should be utilized with each and every recipe (logic) you build.
2. **Error Handling –**Error handling conditions should be incorporated in your recipes to ensure proper routing or retry actions are taken on failures.
3. **Notifications –**Use of notifications during error handling should also be used to ensure visibility to issues. E-mail distribution groups should be used over individual e-mail addresses where possible (ease of future maintenance).

In order to uniquely identify a recipe, naming conventions are highly suggested and can be found in the Workato standards and best practices document (see Workato Confluence site).

### **Version Management:** Every time a recipe is saved, a version of the recipe is saved. Previous versions of the recipe can be restored at any time. Recipe versions can be viewed in the versions tab and are denoted by their version number.

### **Code Review:** It is always a good idea to have another set of eyes to review the work prior to a production implementation. As a best practice, before a recipe is deemed ready for production migration, it should undergo a design / code review for compliance to standards and best practices.

### **Migration/Deployment:** With Yale's Workato Business Plan, there is no formal deployment process. Users/Integrators can move recipes from one instance to the other. Hence, Yale has developed a self-disciplined approach of using ServiceNow tickets for deployments.

### 

### **Workato Documentation Locations**

There is a great deal of additional information and helpful documentation about Workato available. Detailed information on the Workato standards and best practices, including information on the REST API’s available, can be accessed in the locations referenced below.

|  |  |
| --- | --- |
| Document | Document Location |
| Rest API’s | <https://www.workato.com/apidoc> |
| Tutorials, Solution Articles and Community Forums | <http://docs.workato.com/> |
| Support Documentation | <https://support.workato.com/support/home> |
| Workato Demo | <https://yale.box.com/s/rz9hv9v48dhz6h4rs1xfc1a48vf74qkd> |
| Yale Workato documentation (Confluence Site) | <https://yaleits.atlassian.net/wiki/spaces/IN/pages/578682994/Workato+iPaaS> |

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs team.*

## Talend

Talend is an eclipse-based integration tool for ETL and ESB development. Use of Talend is limited to a small number of resources within the University, primarily supporting ETL and service development (Data Services and APIs Team). A very high-level overview of Talend’s use here at Yale is highlighted in the landscape diagram noted in Figure 4.4A.

Separate Talend Infrastructure exists to support our ETL data loads to (data) sources such as People Hub, COA hub, etc., as well as for our enterprise service delivery. The Talend Runtime infrastructure supports over a dozen of Yale’s enterprise services, a handful of specific application services, as well as several portal based API’s.

A Talend Enterprise Development Standards has also been created to help in assisting from installation to deployments. Included are guidelines as well as naming standards that should be considered and applied when naming things such as:

* Naming of our Talend project
* Name and description for your Talend job
* Naming of your metadata

Additional information and links to helpful documents and resources can be located in the table below.

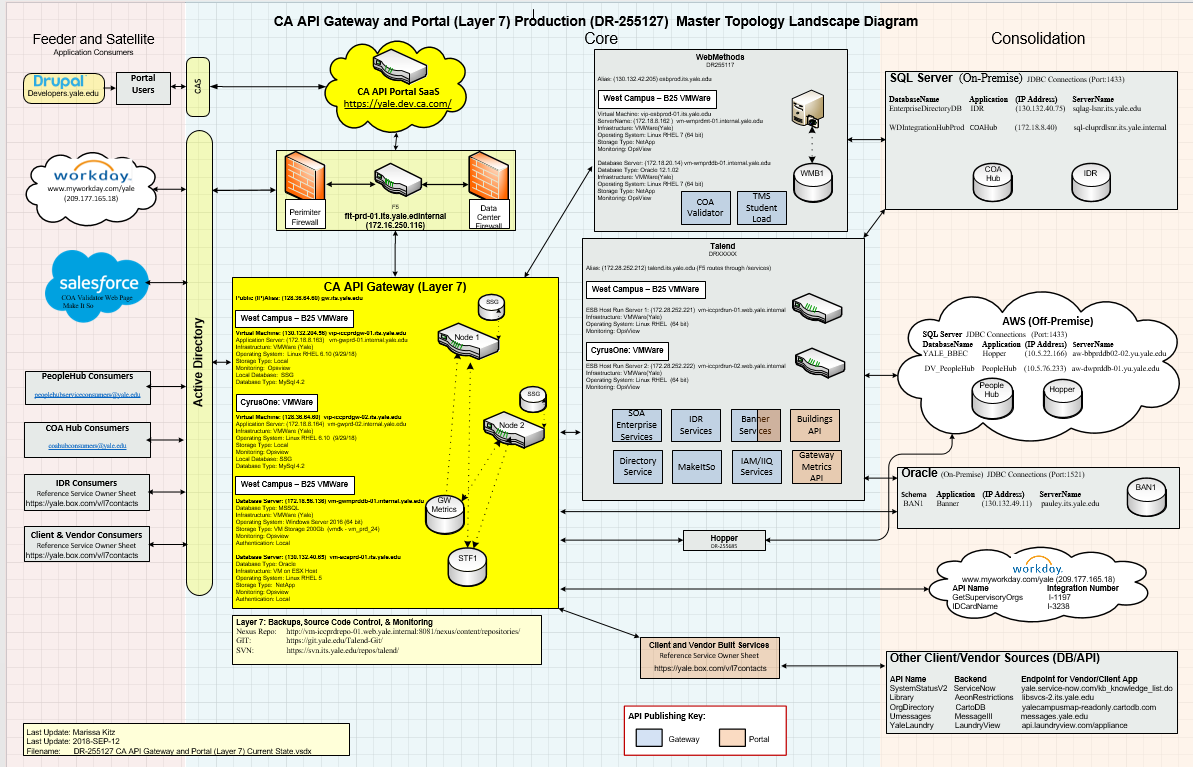
|  |  |
| --- | --- |
| Document | Document Location |
| Talend Overview | <https://yaleits.atlassian.net/wiki/spaces/BSG/pages/754778145/Overview+of+Talend+Data+Services+Platform> |
| Talend Standards | <https://yaleits.atlassian.net/wiki/spaces/TAL/pages/816349929/Talend+7.1.1+Development+Guides+Standards+Best+Practices> |
| Talend (Confluence Site) | <https://yaleits.atlassian.net/wiki/spaces/TAL/overview> |

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs Team.*

## API Management

The CA API Gateway is an XML firewall and service gateway that controls how web services are exposed to and accessed by both internal and external client applications. The Gateway provides runtime control over service-level authentication, authorization, key management, credentialing, integrity, confidentiality, schema validation, content inspection, data transformation, threat protection (including integration with external virus scanners for SOAP attachment scanning), routing, protocol switching, SLA enforcement, logging and other functions.

The Data Services and APIs Team is responsible for the CA API Gateway and for the development and support of ‘wrapping’ and exposing services as well as for creating and maintaining policies. The strategic direction here at Yale is to utilize the CA API Gateway for building new services or exposing of existing services. This ultimately allows for visibility into what applications are consuming services as well as frequency and performance. This information becomes extremely valuable, as consumers may need to be notified of service or data source outages.



**Figure 4.4A**: CA API Gateway (and Talend usage) Landscape Architectural Diagram: Describes the use of the CA API Gateway here at Yale from consuming/invoking applications to back-end data sources.

## When to use the API Gateway

The CA API Gateway should be leveraged as we look to implement any service (ESB, listener, or any other process with an endpoint) that will need to be exposed to allow access into our Yale environment.  This includes services that are called from external vendors or Yale applications hosted in the cloud that need to initiate processes behind the firewall. Placing a gateway policy on top of an existing backend is simple and provides a lot of benefits.

Some of the key benefits and advantages this provides includes:

* Avoids requests for new firewall access – The gateway is already exposed to the internet in Test and Production
* Mitigates unnecessary risks and security vulnerabilities –
  + It limits invalid hits / requests against any of Yale’s backend infrastructure
  + helps to identify attacks by using rate limiting
  + applies standard web service vulnerability and protection (SSL, cross-site scripting protection)
  + provides access control via Yale AD and/or API keys
  + Protects backend infrastructure from being exposed to the internet
* Provides important metrics and monitoring capabilities, these metrics are made available via an API for developers to troubleshoot their services and have access to the data -
  + Who is calling the service
  + Number of calls
  + Latency of service requests (both by the gateway and backend service)
  + What parameters/request was sent
  + Audit logging (captures revisions to the service, any errors, etc.)
* Allows for a consistent point of entry for consumers, even if your backend location changes, the endpoint a consumer hits will remain the same.
* Provides an easy way to translate data into additional formats (XML, JSON, CSV, etc) if the backend endpoint cannot support it
* Decouples consumer from the backend technology, allows both consumer/backend technology to change more easily without significant impacts.
* Can consolidate requests that hit microservices into 1 endpoint for consumers

## Web Service (XML) Editors

Oxygen XML Editor provides a comprehensive suite of XML authoring and development tools (XSLT / XQuery debugger and profiler). It is designed to accommodate a large number of users, ranging from beginners to XML experts. It is a [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) application so it is available on multiple platforms, all major operating systems, and as a standalone application or an Eclipse plug-in. You can use Oxygen XML Editor in conjunction with all XML-based technologies and it includes a large variety of powerful tools for creating, editing and publishing XML documents.

Oxygen also offers a WSDL Editor that supports both editing and online testing. You can use a wide range of searching and refactoring actions and a powerful WSDL documentation tool integrated with XML Schema documentation.

Several XML editing tools are available and have many different advantages and disadvantages. At Yale we have made great strides to standardize in using the Oxygen tool. In order to obtain access to Oxygen, a licensing key is required. Several license keys have been purchased and are in use here at Yale. If a license is needed, please open a ServiceNow ticket to the Data Services and APIs Team to inquire if there are available licenses that can be leveraged.

## Managed File Transfer

Axway (Secure Transport) is used here at Yale for performing certain kinds of file transfers required to interface with systems like Workday and many vendor applications. Specifically, a large number of Journal Staging files are processed using MFT using integration platforms like webMethods (for both picking up and putting files on) as part of integration execution. A standards document around principles (for things like File-based/Batch integrations and Encryption of Files), uses and naming conventions can be found in the table below.

Yale has designated the use of Managed File Server (MFT) as the place where files will be “landed”. Naming standards and other best practices (originally done and created during the Workday project) can be referenced in the attached document below and are included for areas such as:

### Delivery/Retrieval Service Naming Standards

### Reusing Delivery/Retrieval Services

### Security and Access (Encryption of Files and Accounts)

|  |  |
| --- | --- |
| Document | Document Location |
| Managed File Transfer Strategy, Standards and Design Document |  |

# SOA Services (Service Development)

## Overview

Service-oriented architecture (SOA) is a software development model for distributed application components that incorporates discovery, access control, data mapping and security features. The basic principles of service-oriented architecture are independent of vendors, products and technologies. There are 9 design principles to keep in mind when designing a SOA service: Standardized Service Contract, Loose Coupling, Abstraction, Reusability, Autonomy, Statelessness, Discoverability, Composability and Interoperability.

## Service Contract

**What is a Service Contract?**

Services express their purpose and capabilities via a service contract. The basic function of the service contract is to formally specify how the service provider and the service consumer will interact. Specifically, this would be created prior to any service development and serves as the binding contract between the two parties. The service contract should describe functional requirements that are expected from the service, what functionality the provider provides (i.e. formats, validations, etc.) and what the service response will return to any consumer.

**Why is the Service Contract important?**

Within service-oriented solutions, a service contract represents a foundational artifact. Likewise, service contracts represent a cornerstone principle in service-orientation and therefore support or even enable other principles such as loose-coupling, service abstraction, composability, reusability and discoverability. Providing a service contract enables a successful communication and interaction between the service provider and service consumer, but hides the technology and details and only exposes the service interface to the consumer.

**What should the Service Contract contain?**

Service Name

Service Endpoint

Data Source (optional)

Service Description

Author (optional)

Service Version

Input and Output Supported (request parameters/response)

Error Handling Information

Release Notes

|  |  |
| --- | --- |
| Helpful Documents | Document Location |
| Service Contract Template | [Service Contract Template](https://yaleits.atlassian.net/wiki/download/attachments/50134493/Service%20Contract%20Template.xlsx) |
| BSG Application Service Team (Confluence Site) | [Confluence-BSG Application Services](https://yaleits.atlassian.net/wiki/spaces/BASP/overview) |

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs Team.*

## Service Consumer Usage Plans

**What is the Consumer Usage Plan?**

The consumer usage plan is another foundational artifact that is necessary in ensuring a successful relationship between the service provider and the service consumer. The consumer usage plan solicits a consumer’s expected call volume, peak periods, minimum required response latency and impact in the event of service unavailability. With this information collected, meaningful service level agreements (SLAs) can be established and key performance indicators (KPIs) defined to measure whether or not the SLAs are being met. Setting these expectations upfront allows both parties to start the relationship knowing what is expected of them and what they can reasonably expect from the other party.

|  |  |
| --- | --- |
| Helpful Documents | Document Location |
| Consumer Usage Plan Template | [Consumer Usage Plan Template](https://yaleits.atlassian.net/wiki/download/attachments/50134493/Consumer%20Usage%20Plan%20Template.xlsx) |

## Web Service Security Overview

A web service security document was written outlining Yale’s security strategy and standards. This document was written to help provide the specifics on how web service security is implemented and governed while working to help drive and ensure consistency in our SOA approach (see Service Security Strategy and Standards document attached in the documents section below).

In a Service Oriented Architecture, there are many approved patterns for service consumer and service provider interactions. In the ESB pattern, a service consuming application is responsible for generating a token containing a valid user ID and password for a service. The service provider is responsible for accepting the token and validating the user ID and password before execution.

Active Directory (AD) IDs will be used to implement the security token. The service consumer will generate a token containing the ID and password that is associated with that consuming application service provider. The service provider will validate that ID and password.

Additional information and details on the web service security strategy and standards can be found in the attached document reference in the table below.

|  |  |
| --- | --- |
| Documents | Document Location |
| Service Security Strategy and Standards Document |  |

### **Web Service Security Prerequisite**

### All systems projects (and application requests) are required to go thought the data governance process to obtain access to the data attributes/populations that will be needed by that integration. Applications will either need an existing Service Account, or one will be created for them once approved, following the naming standards in place. Data stewards will be identified for each of the source systems who will review and approve/reject requests. Once approved (and depending on the source system), data attribute/populations level security can also be applied (this is specific to People Hub at this time).

### **Naming Conventions and Tracking**

### The IDs that are created for the web service security will be associated to the consuming application. To adhere to the directory naming convention and to limit the size of the IDs, IDs will be in the following format: For Example the YSM Com People the ID will: s\_ph\_complp\_ysm

**Naming standard:**

|  |  |
| --- | --- |
| **s\_** | This designates the account as a Service Account in the Active Directory |
| **ph\_** | Represents that the account is to be used by the Source System (i.e. PeopleHub) |
| **complp\_** | This is a 3-6 character value that is made up to represent your application |
| **ysm** | The Organization Unit for the consuming application (i.e. ysm, law, som, its) |

### **Tracking IDs and Consumer Names**

### The Web Security ID DB (QuickBase) is being used to track all IDs and consumer names. Additional information for what is being tracked can be found in the web service security document referenced in the table above.

## Service Versioning Strategy Overview

With the creation of SOA services, reusability is a key principle. As more and more service consumers are engaged, new requirements necessitate new service versions. Services will continuously need to evolve and procedures must be in place to ensure existing consumers are not impacted. The Service Versioning Strategy document attached below, describes what types of changes constitute major version changes versus minor version changes as well as guidelines for how Yale can best manage version deprecation to ensure that services are and can be properly retired. This is important so that service maintenance activities do not become unmanageable.

|  |  |
| --- | --- |
| Documents | Document Location |
| Service Versioning Strategy Document |  |

A well-documented and understood version strategy will help in dealing with changes to services, now and in the future. Versioning will allow for several variations of the same service to exist simultaneously. While ensuring that existing service consumers can continue to use older versions, it will enable newer consumers to move to any newer and more recent services available. The following illustration shows how two service consumers can consume different major service versions (and also choose whether to consume enhancements related to minor version changes as well).

Service Version 1.0

Service Consumer 1

Can migrate easily to minor version with none to minimal impact

Service Version 1.1

Service Version 2

Service Consumer 2

### **Service Versioning Rules**

### Rules / guidelines for changes that will ultimately determine whether a service will require either minor version changes or major version changes. Major versions will be noted in the subsequent document as M (M=Major version) and minor versions will be noted as m (m=minor versions). Areas for consideration include:

### Schema Changes – there are areas where service schema changes will either constitute a minor version change or a Major version change.

### Production Fixes - Production fixes are also another influencing factor and important aspect to service versioning. It is expected that a production fix would not result in a major version. It would be considered a fix to the current version (M.m) of the service that was previously deployed in production. Once deployed, a production fix to an existing service would increment the minor version by 1 (i.e. M.m+1).

### Other - There may be other influencing factors or decisions that could also alter service versioning. Changes to the overall service implementation, back-out plans, etc. are other possible examples. Regardless of the change, there should be proper analysis and agreement on whether the changes being made would constitute a major or minor version change. If agreement is not reachable, then it is always safest to initiate a major (M.) version change as well as proper communication to service consumers.

It is important to note that regardless of a minor version change or a major version change, existing consumers should be notified, and proper testing should be performed to ensure that there are no impacts.

## Service Deprecation

## The service deprecation (or sunsetting) strategy is an important aspect in order to support the delivery of new versions, while at the same time, maintaining existing versions. The service sunsetting strategy will primarily apply to major service versions. Minor service versions should not constitute service consumer changes and impacts, therefore will be implemented in production without impacts to the gateway URLs. The goal should be to support no more than three service versions (for a given service). Please reference the Service Versioning Strategy document for additional clarification.

## Error Handling

## Another important aspect of web service delivery is to ensure descriptive and helpful error handling of service failures. On a service failure it is important to help ensure applications and / or end users are provided with detailed error codes and messages in order for them to know and understand the nature of the error so that the necessary and appropriate actions can be quickly taken to resolve.

The data services team has established some consistent error codes and error handling mechanisms that we have referred to as invocation outcomes. Within the service contract the applicable outcome codes, outcome types, and outcome messages have been documented. The service response formatting on a failed service call will follow a consistent approach such as:

<People>

<Error>ERROR CODE GOES HERE</Error>

<ErrorMessage>ERROR RESPONSE MESSAGE</ErrorMessage>

</People>

|  |  |
| --- | --- |
| Helpful Documents | Document Location |
| People Hub Web Service Response Codes | <https://yaleits.atlassian.net/wiki/spaces/BASP/pages/50134453/PeopleHub+Services+Troubleshooting> |

*\* Some locations may have restricted access. To request access, open a ServiceNow ticket for the Data Services and APIs Team.*

## Service Discoverability

## API Developer Portal

The Data Services and APIs Team supports a self-service API solution comprised of developers.yale.edu (YaleSites). This site constitutes a content management system (CMS) and third-party software (CA API SaaS Portal), which simplifies API discovery for developers and provides them with access to enterprise data.

Within [developers.yale.edu](https://developers.yale.edu/), developers can find:

* API Documentation
* How-To Info
* Links to developer community resources
* Link to login to the Yale Developer Portal

Developers can use the [**Yale Developer Portal**](https://yale.dev.ca.com/admin/) to access several internal APIs. Developers can use the Yale Developer Portal to:

* Get an API Portal Account
* Browse APIs in the API Catalog
* Get API keys for Applications
* Try out APIs in the API Explorer
* View real-time analytics for APIs and Applications

Access to the Developer Portal is limited to individuals with a valid Yale NetID.

#### Service Wiki – Confluence Site

An additional place to find information on available enterprise services can be found on the BSG Application Service Public Confluence site. The site contains all information on services from service contracts and consumer usage plans to guides on how to test web services.

|  |  |
| --- | --- |
| Documents | Document Location |
| Yale API Portal (Drupal Site) | <https://developers.yale.edu/> |
| BSG Application Service Public (Confluence site) | <https://yaleits.atlassian.net/wiki/spaces/BASP/overview> |

# ETL

## Overview

Information regarding ETL can be found in the ETL standards and shared practices documentation.

Link and other reference information will be forthcoming as the documentation for these standards and guidelines are still in progress.

# Integration Architecture Decision Tree

## Overview

In many cases, decision trees can help provide guidance and direction in making consistent and accurate decisions. Decision trees, like those detailed below, will help force the consideration of all possible outcomes of a decision and traces each path to a conclusion. The following decision trees were created based on criteria used here at Yale and for the purpose of helping guide or answer questions in the enterprise integration space.

## ESB Integration Platform Decision Tree

The following diagram depicts the decision path for selecting the most appropriate ESB integration platform.



**Figure 7.2A**: Enterprise Integration Platform Decision Tree

## Web Service Communication Protocols Decision Tree

The following diagram highlights the decision path for selecting one of the two most widely used web 

**Figure 7.3A**: Rest vs SOAP Protocol