**VA Salesforce  
Center of Excellence  
Platform Standards & Guidelines**

**Version 1.0**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Description |
| 1.0 | 12/08 | COE Team | Initial version |
|  |  |  |  |
|  |  |  |  |

Table of Contents

Introduction 4

Coding Standards and Development Guidelines 4

Intended Audience 4

High Level Overview 4

Standards and Guidelines 5

Introduction 5

Readability Guidelines 5

Naming Conventions for Salesforce Components 6

Use of Descriptive Text 6

Guidelines for Use of Common or Standard Objects 6

Guidelines 7

Regression Testing and Technical Debt 8

Coding Standards 8

Code Structure and Comments 8

Guidelines for Test Classes 9

Apex Guidelines 10

Visualforce Guidelines 11

Development Process, Code Reviews and Exception Requests 12

VA Salesforce Project CCB 12

VA Salesforce Design/Development CCB 13

Development Process and COE interaction 13

Code Review Checklist 15

Componentized Architecture Standards and Processes 19

Intended Audience 19

No Code Technical Approach 20

Code Exception Approval Process 20

On Premise to Cloud Integration Standards 20

Intended Audience 21

Use Case 1: Time & Attendance System 21

Use Case 2: Payroll System 21

Use Case 3: HRIS 22

Use Case 4: Talent Management System 23

Over all Recommendation for Secure On Premise Integration Standards 23

Application Modernization Standards and Guidelines 26

Intended Audience 26

Objective 26

VA CIO Modrnization Goals 26

Platform Decisioning: Value Capability Map (VCM) Metrics 26

Prioritization Using ROI Methodology 27

High Level Frame Work for Prioritization 28

Introduction

The purpose of this document is for documenting the processes for different functions that the Salesforce Administrators at The Department of Veterans Affairs perform. The document is divided into the following sections:

* Coding Standards and Development Guidelines
* Application Modernization Standards and Guidelines
* Componentized Architecture Standards and Processes
* On Premise to Cloud Integration Standards

Coding Standards and Development Guidelines

# Intended Audience

The primary audience for this section are members of the project teams responsible for developing the applications providing business capabilities on The Department of Veterans Affairs (VA) Salesforce platform. Project Managers should make this document available to all team members ensuring they are familiar with coding standards, configuration, and development standards described in this document and enforced by the VA Salesforce Center of Excellence (COE).

The secondary audience for this section is the VA Salesforce Support Team and the VA Salesforce Configuration Management Team. These teams need a working knowledge of the standards established in this document ensuring compliance across the VA Salesforce platform.

# High Level Overview

The VA Salesforce platform is a shared resource that enables projects across VA to quickly develop robust, flexible and scalable applications in a secure cloud environment. The VA COE provides structure and guidance necessary to assist project teams maximize VA’s investment in the platform, with the objective of helping project teams deliver quality solutions. Specifically, the coding standards and development guidelines set forth in this document are intended to:

* **Prevent Conflicts**: Standard objects and other Salesforce components are used for different purposes in the various VA Salesforce applications. A key goal of these standards is to help project teams design and implement their changes in a manner so that they do not conflict or negatively impact other projects.
* **Enhance the Supportability of the Platform:** Adherence to design, configuration, and coding standards helps new projects understand and leverage what has been previously developed and enable operations and maintenance (O&M) teams to better support existing applications.
* **Promote Best Practices**: The Salesforce platform is constantly evolving with new features. The guidance set forth in this document will evolve with the platform to ensure consistency with leading practices and configuration techniques.
* **Help Organization Stay within Governor Limits:** Salesforce is a multi-tenant platform and enforces limits to ensure processes do not monopolize shared resources. The reviews and guidelines described in this document help project teams ensure their applications stay within these limits.

Please note, these guidelines are not meant to restrict developer creativity or provide an undue overhead on VA Salesforce projects. Instead, the intent is to enable project teams to quickly get up to speed with Salesforce best practices, avoid common coding mistakes, and accelerate the application development process with a “configure first” approach that leverages out-of-the-box functionality as much as possible, and to maximize the return on investment for the VA Salesforce platform.

# Standards and Guidelines

## Introduction

The guidelines discussed in this document are organized in the following sections:

* **Readability Guidelines**: This section describes the naming conventions and requirements for use for comments and descriptions to improve the readability of VA Salesforce configuration and code.
* **Guidelines for Use of Common Objects:** This section describes items to consider when using standard objects and other components created by or in use by other project teams.
* **Coding Standards:** This section describes when Apex classes, Apex Triggers, Visualforce pages and components, and other components considered “code” are authorized on the VA Salesforce Platform, how to obtain approval to use code, and guidelines for using code when approved.
* **Development Process, Code Reviews, and Exception Requests:** This section describes how project teams interact with the VA Salesforce COE, how the code reviews will be conducted, and how exceptions to create new profiles, use Apex / Visualforce code, and other deviations from these standards will be processed.
* **Code Review Checklist:** This section provides a checklist for project teams to evaluate their code. The items on this checklist will be verified during code reviews.

## Readability Guidelines

It is important that teams using the VA Salesforce Platform understand how other teams are using the platform to prevent conflicts, increase supportability, and promote reuse. The guidelines in this section describe naming conventions and standards for use of comments and descriptive text to improve the readability of components configured on the platform.

### Naming Conventions for Salesforce Components

The naming convention for Salesforce components along with examples are provided below. The following Salesforce Components are subject to this convention:

|  |  |  |
| --- | --- | --- |
| Profiles | Permission Sets | Queues |
| Page Layouts | Apps | Workflow Rules |
| Approval Processes | Visual Workflow Flows | Process Builder Processes |
| Tasks | Email Alerts | Outbound Messages |
| Custom Objects | Apex Classes | Apex Triggers |
| Visualforce Pages | Visualforce Components |  |

**Component Name / Label:** New Salesforce components are named in accordance with the following convention:

**[Organization Acronym]-[Application Acronym]-[Meaningful Component Name]:**

***Example***: The custom object name, VBA-CS-Schools indicates the organization this object supports is the Veterans Benefits Administration, it was created to support the Compliance Schedule (Compliance & Liaison) application, and it contains school data.

Note, labels for custom fields are displayed to users of the application and must be user friendly. Field labels are not subject to this requirement.

API Names: API names use Salesforce default naming convention consistently. In this convention API names mirror the object or component name; all words have the first letter capitalized and spaces are replaced by the underscore character.

***Example:*** The API name for the VBA-CS-School object is VBA\_CS\_Schools\_\_c

### Use of Descriptive Text

The Description attribute must always be populated. The description provides a brief but concise description of how the component is used.

***Example:*** The description for the VBA-SSD-GSA Trip Tracking object: “Track all the trips made in GSA Vehicles. Tracks the miles driven and for what purpose.”

## Guidelines for Use of Common or Standard Objects

VA Salesforce project teams should consider using standard Salesforce objects wherever practical to take advantage of Salesforce out-of-the-box functionality. This is especially true when there is close alignment between project requirements and the standard Salesforce use case for the object. However, it is critical that project teams research the existing object, identify if and how other teams are using the existing object, and take precautions to ensure that any modifications to the object do not negatively impact current functionality.

### Guidelines

The guidelines described below describe tasks that project teams must complete when using standard objects to prevent conflicts with other applications that are currently using the object, or that may do so in the future.

| Guideline | Rational |
| --- | --- |
| Consider using a new record type, and/or other mechanisms to uniquely identify your project’s records | Allows your project’s data to be distinguished from other data stored in the common object. ***Note***: this is important even if your project is the only application using the object, as other teams may have use for the object in the future. |
| Review existing sharing settings for the object | Organization-wide defaults and other sharing settings already established may impact how your project utilizes the common object |
| Review object permissions for existing profiles and permission sets | Provides insight into the how the common object is currently utilized, the types of users that have access to the common object, and any restrictions to field visibility that may be established. |
| Review existing Workflow Rules, Visual Flows, Process Builder Flows, Assignment Rules, Escalation Rules, and Support / Sales processes for potential conflicts | Helps ensure that updates to the project’s data does not trigger another project’s workflows unintentionally. Identifies changes to existing components that may be required to exclude data from the project. |
| Review existing Apex Classes and Triggers | Helps ensure that updates to the project’s data do not trigger another project’s code unintentionally with unexpected results. Helps identify changes to existing code that may be required to exclude data from the project. |
| Review existing Reports and Dashboards using the object | Ensure that new data is not inappropriately included on existing reports and dashboards |
| Review existing List Views | Ensure that new data is appropriate on existing list views |

### Regression Testing and Technical Debt

**Regression Testing:** It is important that ***ALL*** application functionality supported by the object is completely tested prior to release**.** This includes functional testing of the new capabilities supported by the object and complete regression testing of pre-existing features previously supported by the object. Regression testing is necessary to ensure that changes do not negatively impact existing functionality.

Ideally, regression tests should be performed by the project team that originally developed and tested the capability. If this is not possible, the COE, product owners, and/or other persons familiar with the original functionality should be engaged to ensure that each affected component is adequately regression tested prior to release.

**Technical Debt:** It is likely that existing components will need to be refactored when objects are reused. The project team will provide the COE a list of components that require refactoring for the reused object. Refactoring occurs prior to regression testing and ideally should be performed by the project team that developed the original capability.

## Coding Standards

The VA Salesforce COE considers Apex Classes, Apex Triggers, Visualforce Components, Visualforce Pages, and other components created using a programming language (Apex, Visualforce, HTML, Javascript, etc.) to be “code”, as opposed to declarative development.

* An exception to VA Salesforce Platform COE standards is required to use code. The exception process is described in the “Code Reviews and Exception Requests” section of this document.
* If declarative techniques can address requirements, use of code on the VA Salesforce Platform is to be avoided.
* The VA Salesforce COE will work with project teams during design to identify declarative alternatives to code.
* If code is the best or only option to meet a requirement, an exception to use code may be granted by the VA Salesforce Design/Development Configuration Control Board (CCB) as described in the Development Process, Code Reviews and Exception Requests section of this document.
* The standards and guidelines described in this section must be followed if an exception to use code is approved.

### Code Structure and Comments

Apex code structure and comments should focus on readability. Classes and methods use standard and meaningful naming conventions. Comments are used to explain the purpose of classes, methods, and code.

**Naming Conventions:** Apex classes use a similar convention to those described in the Readability Guidelines above, however CamelCase notation is used instead of underscores.

|  |  |  |
| --- | --- | --- |
| Type | Convention | Rationale /Example |
| Apex Controllers | [Organization Acronym][Application Acronym] [Meaningful Class Name][Controller] | Appending Controller to the end of name indicates that the code is an Apex Controller  Example: VBACSSchoolsController |
| Test Classes | [TestedClassName][Test] | Appending Test to the end of the name indicates the code is a Test class  Example: VBACSSchoolsControllerTest |
| Triggers | [ObjectName][Trigger] | Prepending the object name in front of the trigger clearly indicates the object on which the trigger is acting  Example: VBACSSchoolsTrigger |
| Methods, Variables | Meaningful name in camelCase with initial letter lower case | Using camelCase provides a visual distinction from the API names of custom objects and fields (which use the standard Salesforce notation e.g. VBA\_CS\_School\_\_c) |

**Comments**: Classes and methods contain an information block that briefly describes its purpose, creation date, and a change log. Within methods, more complex sections contain brief comments that describe the code.

Information Block Format:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Name: [Class Name]

\* Description: [Brief description of the class and what it does]

\* Change Log:

\* -----------------------------------------------------------------------------------------

\* Change ID     |   Date     | Author             | Description

\* -----------------------------------------------------------------------------------------

\* 0001       | [Date]     | [Developer Name]    | [Brief description of change]

\* -----------------------------------------------------------------------------------------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

### Guidelines for Test Classes

The table below describes guidelines for creation of test classes and methods on the VA Salesforce Platform.

| Guideline | Rational |
| --- | --- |
| Design and develop test classes and methods at the same time the Apex code to be tested is written | Creating test classes and methods simultaneously with development of the tested code helps to ensure effective testing of expected conditions, helps enforce inclusion of error handling within Apex code, and helps prevent the creation of last minute classes solely for the purpose of meeting code coverage requirements. |
| Create test methods that simulate business functionality, use cases, and error conditions | Test methods should not be created solely for the purpose of meeting code coverage requirements. They should be designed to test the use case that is addressed by the Apex code, as well as to validate error handling code. |
| Create test data within test classes and methods | Developers should not expect that data within the Salesforce organization will be available for testing purposes. Also, they should not anticipate that organization data will have the necessary values to test all conditions and code branches. All test data should be created with test classes to ensure expected results coincide with conditions in test data. |
| Use assert statements logically and correctly | Assert statements are used to validate that code works as intended. Without assert statements, code is not truly tested. Meaningless assert statements such as system.assert(true=true) should not be used. |
| Don’t focus solely on code coverage | Project teams keep the Salesforce 85% code coverage requirement in mind; however, they also focus on ensuring all use cases and possible conditions are tested. |
| Do not use @isTest(SeeAllData=true) | This could allow someone to remove or modify data that they should not have access to. |

### Apex Guidelines

Salesforce provides a number of excellent guidelines for developing Apex code. The intent of this section is not to rewrite existing guidance, but to provide VA Salesforce project teams with links to existing guidance and best practices and highlight areas where the Salesforce COE will focus during design and code reviews.

#### Salesforce Apex Resources:

See the links below for Salesforce guidance and best practices for the use of Apex:

* [Apex Best Practices:](https://developer.salesforce.com/page/Apex_Code_Best_Practices) Describes core Apex best practices for writing efficient, scalable code
* [15 Apex Commandments:](https://developer.salesforce.com/blogs/developer-relations/2015/01/apex-best-practices-15-apex-commandments.html) 15 additional guidelines for efficient Apex code
* [Secure Coding Guidelines:](https://developer.salesforce.com/page/Secure_Coding_Guideline) Describes common security issues Salesforce has identified while auditing applications built on or integrated with Force.com
* [Apex Developer's Guide](https://developer.salesforce.com/docs/atlas.en-us.apexcode.meta/apexcode/apex_intro.htm): Provides a comprehensive guide to Apex code and syntax

#### Focus Areas:

The following list highlights areas that project teams focus on during design and development of Apex code. These will be areas of focus during code reviews.

* **Performance and Governor Limits:** Loops, queries, DML statements, recursion, and other code constructs that are likely to have a negative impact on performance and cause governor limits to be exceeded should be designed carefully. Code reviews will focus on ensuring that SOQL and DML are not executed within loops and that queries are efficient and use collections where possible.
* **Bulkification:** Triggers can be invoked via batch updates that impact multiple records at a time. Project teams should ensure that triggers and trigger helper classes are written to handle multiple records once. Code reviews review triggers to ensure they handle bulk update situations appropriately.
* **Triggers:** The sequence of trigger execution cannot be controlled in Salesforce and having multiple triggers on an object reduces code manageability. Project teams should ensure that only one trigger per object is created. Additionally, triggers can be constructed so they invoke themselves. Project teams should validate their code takes appropriate measures to prevent the unwanted, recursive execution of triggers. Code reviews will ensure one trigger is created per object and check recursion.
* **Exception Handling:** Apex code should have Try-Catch-Finally statements in place to handle and recover from exceptions wherever possible. Project teams validate that all Apex code includes appropriate exception handling. Proper use of exception handling is checked during code reviews.
* **Redundant Code:** Blocks of redundant / repeated code make maintenance more difficult. Project teams identify blocks of code that are repeated and are candidates to be condensed into a single class or method.
* **Secure Coding Guidelines:** Project teams review the Secure Coding Guidelines and ensure the Apex code conforms to the described practices.

### Visualforce Guidelines

Similar to Apex, Salesforce also provides guidelines and best practices for developing Visualforce. The intent of this section is not to rewrite existing guidance, but to provide VA Salesforce project teams with links to existing guidance and best practices and highlight areas where the Salesforce COE will focus during design and code reviews.

#### Visualforce References:

* [Visualforce Developer’s Guide](https://developer.salesforce.com/docs/atlas.en-us.204.0.pages.meta/pages/pages_intro.htm): Provides a comprehensive guide to Visualforce code and syntax.
* [Visualforce Performance Best Practices](https://resources.docs.salesforce.com/sfdc/pdf/salesforce_visualforce_best_practices.pdf): Provides best practices for optimizing the performance of Visualforce pages.

#### Focus Areas:

The following list highlights areas that project teams focus on during design and development of Visualforce code. These are the areas of focus during code reviews.

* Ensure Javascript and CSS are included as static resources to promote consistency and reuse.
* Ensure that custom Visualforce Components are created and used appropriately to promote consistency and reuse.
* Validate that Javascript is not used for functionality that can be accomplished using native Visualforce features.
* Employ techniques to optimize Visualforce performance, including:
  + Using the cache attribute with the <apex:page> component to take advantage CDN caching (when appropriate).
  + Marking controller variables as “transient” (if not needed between server calls) to enable faster page loads.
  + Using <apex:repeat> to iterate over large collections.
  + Use lazy initialization in Visualforce controllers to improve page performance.
  + Validate that the Visualforce page addresses 508 compliance. For example, ensure that all tables have summary text and graphics have ALT text set appropriately.

## Development Process, Code Reviews and Exception Requests

This section describes the various touch-points that project teams have with the VA Salesforce Platform COE during the development process, how the COE conducts code reviews, and how exceptions to create new profiles, use Apex / Visualforce code, and other deviations from the standards described in this document will be processed.

The *VA Salesforce Configuration and Change Management Plan* (July 2016) establishes and documents the VA Salesforce Configuration Management Team and the VA Salesforce Configuration Management Boards (CCBs). These organizations are responsible for the integrity of the Salesforce platform. It is through the CCBs that the development process is managed, code is reviewed, and exception requests are processes. Each CCB is described in more detail below:

### VA Salesforce Project CCB

The VA Salesforce Project CCB is responsible for monitoring and managing existing Salesforce projects and reviewing and approving new projects and future enhancements. Project teams will interact with the VA Salesforce Project CCB as needed for issues regarding Salesforce licenses, sandbox management, and configuration management.

### VA Salesforce Design/Development CCB

The VA Salesforce Design / Development CCB helps maintain the integrity of the Salesforce platform by reviewing and approving project designs, code and configurations. This CCB also provides a forum to assist with design decisions, and answer questions regarding technology and the Salesforce product. Project teams interact with the VA Salesforce Design/Development CCB weekly.

**Exception Requests:** The VA Salesforce Design and Development CCB is responsible for approving exceptions to allow the use of code, and to create new profiles. Exception requests are submitted by a form in Salesforce (currently under development), reviewed by the CCB, and approved during the design review.

**Design Reviews:** Project teams must have their designs reviewed and approved prior to the start of development. Designs will be reviewed at the weekly CCB meeting.

**Code Reviews:** Prior to deployment to the System Integration Test (SIT) or User Acceptance Testing (UAT) environments, all code must be reviewed by the CCB. Prior to deployment to production, all issues noted in code reviews must be resolved.

### Development Process and COE interaction

The timeline described in the table below highlights the development process and interactions between the VA Salesforce COE and project teams. Note, this timeline is based on a two-week Sprint, with deployment the following week. The dates in the timeline may be adjusted to support individual projects.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Week | Due Date | Task / Interaction | Mechanism / Board | Comments |
| 1 | Day 1 (Monday) | User Story / Requirements Review | VA Salesforce Project CCB | Product Owner approved user stories are submitted to and reviewed by the CCB. Design and development cannot begin until approved user stories are reviewed by the CCB. |
| 1 | Day 3 (Wednesday) | Exception Requests submitted to COE | VA Salesforce Exception Request form | Requests to create a new profile, develop code to meet requirements, or other deviations from VA Salesforce Platform standards must be submitted using the exception request form, at least three days prior to the Design Review. This form is currently under development and will be accessible in Salesforce. |
| 1 | Day 5 (Friday) | Design Review / Exception Request Approval | VA Salesforce Design / Development CCB | Project Team presents their design to the VA Salesforce Design / Development CCB for review and approval. |
| 2 | Day 10 (Friday) | Code Review | VA Salesforce Design / Development CCB | The project team conducts a walkthrough of configuration and code with the VA Salesforce Design / Development CCB. |
| 3 | Day 12 (or Prior to SIT/UAT Deployment) | Resolve outstanding issues from code review | VA Salesforce Design / Development CCB | All issues identified in the Code Review must be resolved before the VA Salesforce Administration team deploys code to higher environments. |
| 3 | Day 12 (or Prior to SIT Deployment) | Finalize Deployment Plan | VA Salesforce Design / Development CCB | The deployment plan is developed and tested by the project team, prior to the execution of SIT/UAT deployment by the VA Salesforce Administration Team. See the “*VA Code Deployment Management Plan”* for more details. |

## Code Review Checklist

This section provides a checklist for evaluating project teams’ code. This checklists augments standards described elsewhere in this document, and the references provided. Items on this checklist are verified during code reviews.

Code Review Checklist

DATE: [DATE COMPLETED]

APPLICATION REVIEWED: [APPLICATION NAME]

REVIEWER: [REVIEWER NAME]

| Item # | Description |
| --- | --- |
| 1 | Is all code well-structured, consistent in style, and consistently formatted (Is TABS used in formatting)? |
| 2 | Are there any uncalled/unneeded classes or triggers? |
| 3 | Are there any blocks of repeated code that can be condensed into a  single class? |
| 4 | Does the code implement naming and other standards as defined in these coding standards? |
| 5 | Are there any redundant or unused variables? |
| 6 | Do all test classes and methods run/execute successfully? |
| 7 | Is the code clearly and adequately commented and is the code consistent with the comments? |
| 8 | Are all variables properly initialized? |
| 9 | Are triggers appropriately bulkified? |
| 10 | Are collections and loops used appropriately to process records? |
| 11 | Are SOQL and SOSL queries avoided inside of loops? |
| 12 | Are DML statements avoided inside of loops? |
| 13 | Is there no more than one trigger per object? |
| 14 | Are unwanted trigger recursions avoided? |
| 15 | Are any record IDs hard coded? |
| 16 | Does all code include appropriate exception handling? |
| 17 | Are static queries, bind variables and escapeSingleQuotes method used to protect against SOQL and SOSL injection attacks |
| 18 | Are any classes excessively complex and should be restructured or split into multiple routines? |
| 19 | Do triggers contain complex logic that is better maintained in a separate Apex class? |
| 20 | Are controllers used to define picklist values? (instead of including them in the Visualforce pages) |
| 21 | Are controller variables or wrapper classes used to hold data referenced in Visualforce pages? (this will avoid “insufficient privilege” error if user does not have access to the data) |
| 22 | Is “With Sharing” used in Apex Classes to honor sharing settings and rules that prevent unauthorized access to data? |
| 23 | Are JavaScript and CSS included as static resources instead of on Visualforce pages? |
| 24 | Are CSS references located at the top of, and JavaScript references located at the bottom of Visualforce pages to provide faster page load? |
| 25 | Is Javascript used that can be replaced by Visualforce conventions? |
| 26 | Have Visualforce pages accounted for 508 compliance? (e.g. Do all graphic components have an alt text? |
| 27 | Do test methods simulate business functionality and use cases? |
| 28 | Are assert statements used logically, and correctly? |
| 29 | Is test data created appropriately in test methods and classes? |

Componentized Architecture Standards and Processes

# Intended Audience

The primary audience for this section are members of the project teams responsible for developing the applications providing business capabilities on The Department of Veterans Affairs (VA) Salesforce platform. Project Managers should make this document available to all team members ensuring they are familiar with coding standards, configuration, and development standards described in this document and enforced by the VA Salesforce Center of Excellence (COE).

The secondary audience for this section is the VA Salesforce Support Team and the VA Salesforce Configuration Management Team. These teams need a working knowledge of the standards established in this document ensuring compliance across the VA Salesforce platform.

## No Code Technical Approach

Regardless of your role, you'll need to able to make wise decisions when it comes to delivering functionality for VA Salesforce Organization. Inevitably, you will encounter scenarios in which a solution can be delivered either with or without of the user of code.

There are unique pros and cons in every situation, but do consider a few key points that may support the decision to avoid using code:

* The need to consider Salesforce.com limits and parameters is significantly reduced or, at times, eliminated completely when building solutions using declarative means.
* Modifications are often more straightforward, as they may only require a change to a configuration setting, not to a line of code.
* No code coverage for testing is required to deploy(Apex test classes, in contrast, must be written for custom Apex Code)
* Knowledge transfer burdens are reduced, since an understanding of the particular feature or function is typically sufficient to quickly determine what a specific application is intended to do
* Future maintenance is simplified, If for example, an individual who built custom applications for you leaves abruptly, picking up the pieces is much simpler if the work was done declaratively.

There are numerous scenarios that may not able to achieve business requirement with declarative method, which need to do development with code. However, developing with code in Salesforce.com is often done based on need; Many times want to extend the platform to support functionality that simply does not exist "Out of the box". Make sure you have a solid justification for developing with code if your business requirement cannot be met through configuration.

## Code Exception Approval Process

DTC Exception Request custom link is available under VA Help Desk Links on Homepage to submit any Exception request for review to get approval for any new profile change creation, Code Exception,Update existing Profile. You can collaborate any questions/concerns on anything in VA System Integrator chatter public group, anyone can join to this public group.

On Premise to Cloud Integration Standards

# Intended Audience

The primary audience for this section are members of the project teams responsible for developing the applications providing business capabilities on The Department of Veterans Affairs (VA) Salesforce platform. Project Managers,System Adminstrators ,Configuration Management Teams.

Below are the different Use Cases & Busniess Scenarios at The Department of Veterans Affairs (VA) that needs to follow the On Premise /Secure Cloud Integration Standards Process.

## Use Case 1: Time & Attendance System

**Business Scenario & Characteristics**

* Web based Time & Attendance app used to enter vacation requests by employee
* Vacation data to be viewable by users via Salesforce & also used by Salesforce applications for calculating performance analysis & reporting
* Inbound data integration to Salesforce
* ~15K employees total; Volume of requests per day : Approximately low 1000s
* Frequency of updates - No need for real-time updates. 1-2 times a day update in SFDC will be sufficient
* Every user will have access to Salesforce & the Time & Attendance app. Single-sign on not available.

**Recommendations & Best Practice Considerations**

* Implement scheduled batch jobs using **SOAP or REST API** for uploading data into Salesforce
* Fine grain interaction is sufficient for this (no rules; just sequence of the records)
* Factor Parent child dependencies in design;
* Use external IDs to simplify parent child relationships (e.g. employee ID)
* Consider Insert vs Upsert operations to handle both new records vs updated records
* Plan for retention strategy for historical data

## Use Case 2: Payroll System

**Business Scenario & Characteristics**

* Employee data to be synchronized between Payroll system and Salesforce for performance analysis & reporting within Salesforce
* Inbound data integration to Salesforce
* ~15K employees total; Volume of requests - 15K requests every 2 weeks approximately
* Contact object with custom fields used to represent employee record
* Frequency of updates - Bi-weekly
* Employee info stored as Contact records and likely to be related to either an Account representing the office, or a single Account for the agency

**Recommendations & Best Practice Considerations**

* Implement scheduled batch jobs using **SOAP or REST API** for uploading data into Salesforce
* Fine grain interaction is sufficient for this (no rules; just sequence of the records)
* Factor Parent child dependencies in design;
* Use external IDs to simplify parent child relationships (e.g. employee ID)
* Consider Insert vs Upsert operations to handle both new records vs updated records
* Consider security/visibility requirements in the use of Contact object for this purpose

## Use Case 3: HRIS

**Business Scenario & Characteristics**

* Employee data to be synchronized between HRIS system and Salesforce for determining user profiles based on employee position, as well as other privileges/metric values for other apps (e.g. performance goals)
* Inbound data integration to Salesforce
* ~15K employees total; Volume of requests - 15K requests every 2 weeks approximately
* Contact object with custom fields used to represent employee record
* Frequency of updates - Bi-weekly

**Recommendations & Best Practice Considerations**

* Implement scheduled batch jobs using **SOAP or REST API** for uploading data into Salesforce
* Fine grain interaction is sufficient for this (no rules; just sequence of the records)
* Factor Parent child dependencies in design;
* Use external IDs to simplify parent child relationships (e.g. employee ID)
* Consider Insert vs Upsert operations to handle both new records vs updated records

## Use Case 4: Talent Management System

**Business Scenario & Characteristics**

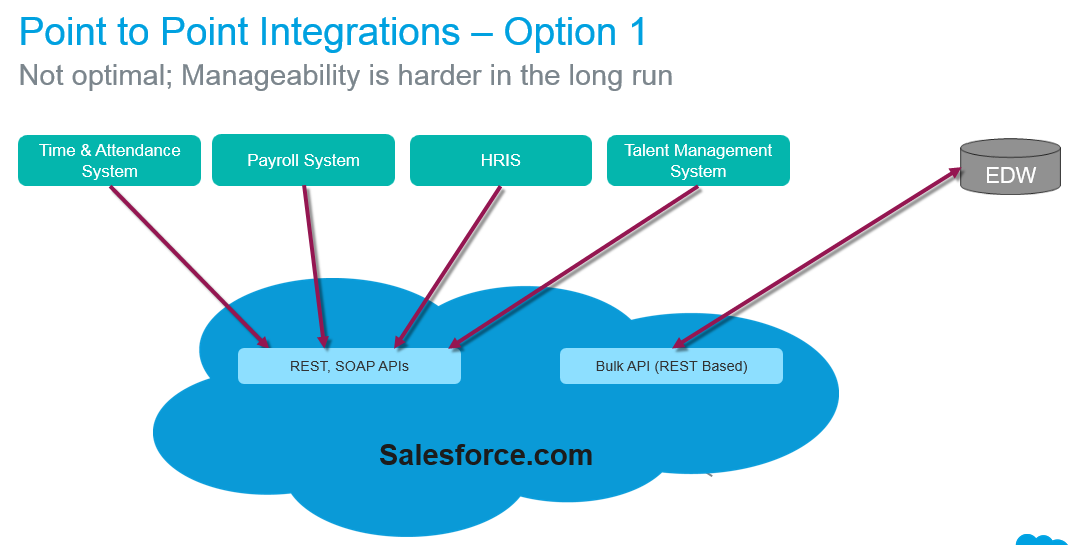
* Training data to be synchronized between Talent Management system and Salesforce for performance analysis and reporting
* Inbound data integration to Salesforce
* ~15K employees total; Volume of requests unknown – approximately handful of records per employee annually
* Historical data not required (only current year)
* Frequency of updates – Nightly

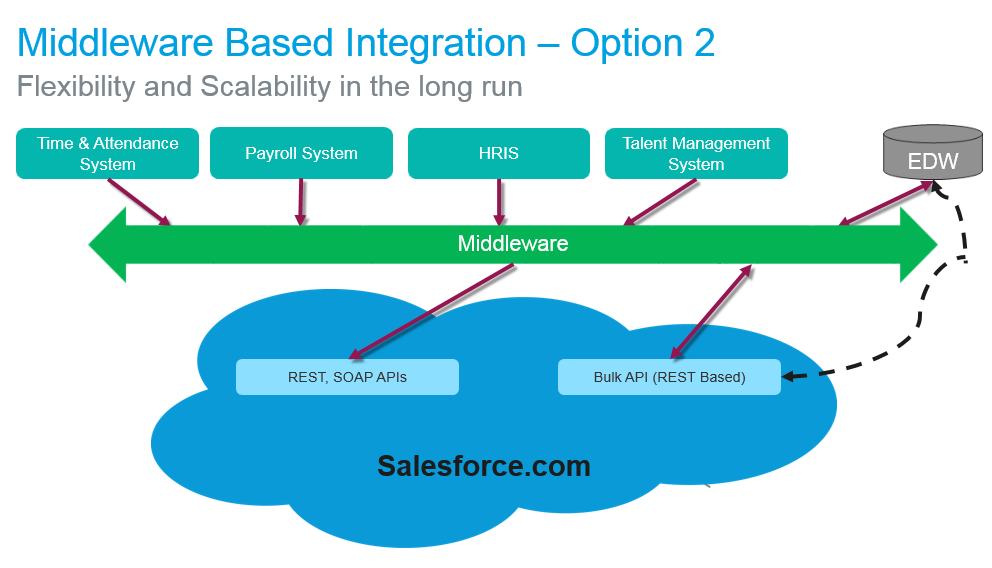
**Recommendations & Best Practice Considerations**

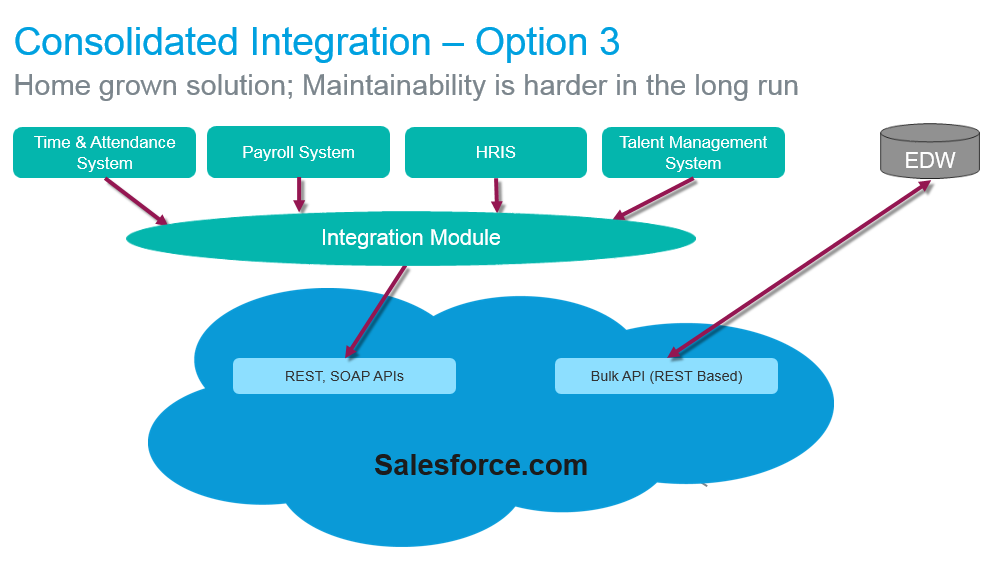
* Implement scheduled batch jobs using **SOAP or REST API** for uploading data into Salesforce
* Fine grain interaction is sufficient for this (no rules; just sequence of the records)
* Factor Parent child dependencies in design;
* Use external IDs to simplify parent child relationships (e.g. employee ID)
* Consider Insert vs Upsert operations to handle both new records vs updated records
* Plan for retention strategy for historical data

## Over all Recommendation for Secure On Premise Integration Standards

* Consider Middleware tool (Informatica, Jitterbit etc) to handle all batch integrations
  + On-premise version as against cloud version options are available, based on security needs
  + Factor in the option to switch over to Bulk API, if required for future growth and needs
* Other option to consider is to consolidate all integrations to common app (e.g. Oracle) instead of separate point to point integrations
* Choose option that supports the ability to add-on ODATA interface, to be able to implement Salesforce Connect technology, if need arises in future







Application Modernization Standards and Guidelines

# Intended Audience

The purpose this document is for helping VA Management/Executives for making decisions to move any legacy VA applications into Salesforce.com in order to align Modernization Goals to VA CIO Strategy and VA Enterprise Strategy.

## Objective

Develop standards and processes to evaluate and prioritize legacy applications and move them to the Salesforce cloud.

High level Application Modernization Standards and Guidelines are :

* **VA CIO Modernization Goals**
* **Salesforce Platform Decisioning**
* **Prioritization**

## VA CIO Modrnization Goals

What are the business drivers/goals behind moving to the cloud?

* Increase Operational Efficiencies
* Avoid Operational Expenses
* Improve VA Veteran demand
* Improve VA Veteran interaction
* Increase agility for System enhancements
* Deliver new capabilities/modernization

## Platform Decisioning: Value Capability Map (VCM) Metrics

**IT Staff Productivity :**

* % decrease in time to develop backlog items.
* % decrease in time to develop admin tasks**.**

**Time to Market (Faster Gain realized):**

* Rate of reduction in time to market i.e,% increase of development projects that meet time-to-market.
* Forecast Time-to-Profit dates to identify early success or failure of a project**.**

**Business Productivity :**

* % decrease in time taking to serve veterans multiply by no of users /cost per user.
* % increase Mission-specific effectiveness metrics (uniqueness and viability).

**Cost Avoidance :**

* Current cost of the system due to licenses, hardware, maintenance staff, data center moves versus upgrades saving from moving legacy application to salesforce.com to decom legacy application.

**System Flexibility**

* Cost savings for continued development for new features to add with ease of use.

## Prioritization Using ROI Methodology

After VA decide to use Salesforce as a platform of choice for legacy application developments efforts based on the different metrics gathered from the decision processing, VA can analyze the benefits and costs which will attribute to use of salesforce. VA can use below three-step method for conducting the ROI analysis to determine best candidate to migrate legacy applications into Salesforce.

* Gather quantitative benefit information during the salesforce platform decisioning process using a before-and-after assessment. Priorize the legacy application which has benefits included higher user and IT staff productivity, higher revenue, and reduced infrastructure-related costs.
* Create a complete investment (five-year total cost analysis) profile based on the users. Investments go beyond the initial and annual costs of using Salesforce and can include additional costs related to IT staff time for deploying and managing the Salesforce platform and additional costs associated with migrating to Salesforce.
* Calculate the ROI and payback period. Conduct a depreciated cash flow analysis of the benefits and investments for use of the Salesforce platform over a five-year period. ROI is the ratio of the net present value (NPV) and the discounted investment. The payback period is the point at which cumulative benefits

## High Level Frame Work for Prioritization

* Use below high level framework for prioritizing legacy application migration into salesforce platform by using simple Business Value(ROI) versus Implementation Complexity.
* Develop a higher value solution inspired by the low value feature that can move up to one of the upper quadrants, or work on small deliverable that will move the ball on one of the larger high value features.

