DTC Standard Operating Procedures for the VA Salesforce Platform

DTC Standard Operating Procedures for the VA Salesforce Platform

Introduction

The purpose of this document is to provide an overview of the different functions performed by the Digital Transformation Center (DTC) at the Department of Veterans Affairs (VA). The document is divided into the following sections:

VA Platform Standard Operating Procedures

- User Account Management
- Data Management

Application Modernization Standards and Guidelines for VA Salesforce Development

- Deployment Management Plan
- Version Control/Source Control

Intended Audience

The primary audience for this document is members of the project teams responsible for developing modules that provide business capabilities on the VA Salesforce platform. Project managers should make this document available to all team members supporting module deployment to ensure they are familiar with the processes described in this document and understand the activities required for a successful deployment.

The secondary audience for this document is members of the DTC team. These teams will have a working knowledge of the processes described in this plan.

User Account Management

Within the VA Salesforce Platform there are several types of user accounts. The table below outlines the various types and their purpose.

Table 1: User Account Management

Account Types	Purpose
User Internal	Tied to an individual who is an internal Salesforce license.
User Community	Tied to an individual who is a Salesforce Community license type. Community users are given limited access to specific functions in Salesforce.
Integration	User account that explicitly is used for integration to Salesforce (Web services, etc.).
User – Sandbox	A user account that gives user access to a test, training or development environment. By default, users with this access have the matched credentials as in Production.
Administrative User	There are a few administrative accounts used to deploy changes to Salesforce Production.
Emergency	An individual user may be created for this purpose, but the Account will be tied to the individual user's credentials.
Guest/Anonymous /Shared	**NOT** allowed in VA Salesforce.

Establishing Accounts

A specific process needs to be followed to establish an account for each user within the system. The following steps must complete before a new account is established.

The user completes a User Permissions Request Form in the DTC Help Desk App, which includes the following:

1. Verification that user is current on signed Rules of Behavior, Cyber Training, and Privacy Training
2. Federation ID
3. Email Address
4. Manager
5. Does the user require colorblind palettes on charts?
6. Any specific requirements or permissions

Note: These items require the direct written approval of the Module Approver in an email or chatter post in addition to their approval of the case request.

The user must submit the request for Account access. A brief description of the modules the user needs access should be included in this request.

Once a user has completed the account request, the system owner for the Module must approve. The VA Helpdesk team will then review the request and verify that the appropriate approval has been received. The user will then be created and/or added to the appropriate security group (s) based on job requirements and enforcing the least privilege concept. In the event a user requires Elevated Permissions, it must be justified and approved by the system owner/ISO or their designated organizational official.

Training requirements:

- Cyber Training
- Privacy Training
- Rules of Behavior Training

Modifying Accounts

If a user is requesting a modification to their account, they must submit an access request (via the VA User Account Request form) along with valid justification to the designated approval authority. Once the approving authority approves the request, the System/Module Administrator is notified, and the modification is made to the account.

Disabling Accounts

For VA users, the VA FIM process is inherited for offboarding. When access to the VA Network and SSO are disabled, the user will no longer be able to access VA Salesforce. Maintenance process by the DTC ensures that accounts that have been inactive for 90 days get disabled.

Deleting Accounts

Should an account need to be removed due to termination or reassignment of responsibilities, the account will be disabled at the same time (or just before) the employee is notified of their dismissal or upon receipt of resignation and the VA FIM offboarding process is inherited. The user will no longer be able to access VA Salesforce when their access to the VA Network and SSO are disabled. The user will be set to Inactive in VA Salesforce with specific security measures put in place, including the removal of permission sets, groups, queues, etc. which provide application access following 90 days of inactivity.

If a user account should need to be reactivated, the user MUST request a user account via the normal new user channels and appropriate access will be given.

Project/Department Transfers

When a user is transferred to a different department or project within the VA Salesforce environment, the current Project Manager is responsible for notifying the Module Administrators. This process is designed to ensure that privilege creep does not occur.

Auditing Accounts

The VA Salesforce Platform will utilize the Salesforce audit capabilities to audit the creation, modification, disabling, and termination actions of users and system/service accounts. All changes that are performed on accounts within The VA Salesforce Platform are tracked in the VA Salesforce Platform module.

Test “Dummy” Users

Project Teams are responsible for the creation of any test “dummy” users in DEV, SIT and INT. If test users are needed in REG, the appropriate setup instructions should be provided in the Deployment Plan for the DTC Release Team. Test “dummy” users are not permitted in environments past REG as those are full copy or Production environments which contain PHI/PII, and individual access must be managed/monitored.

Data Management

VA Data Backups

The VA performs Salesforce Data backups on a regularly scheduled basis to ensure that information is retained and for system recovery purposes.

Data Archiving

In accordance with NARA guidelines, VA exports data directly from Salesforce in a NARA- approved electronic format. The retention periods for data vary based on the type of data. More information can be found in the NARA Records Control Schedule (RCS).

Audit Logs

VA maintains and manages an audit log to track Salesforce record access and user account information. Salesforce maintains the login history of users. However, it does not store all the record changes and activity for each record.

Application Modernization Standards and Guidelines for VA Salesforce Development

The VA Salesforce platform is a shared resource that enables projects across VA to quickly develop robust, flexible, and scalable modules in a secure cloud environment. The DTC Team provides structure and guidance necessary to assist project teams to maximize VA’s investment in the platform, with the objective of helping project teams deliver quality solutions.

This section outlines the processes and technologies used to deploy VA Salesforce modules from development environments to the Systems Integration Testing (SIT), Integration (INT), Regression (REG), Staging or Performance (PERF), and Production (PROD) environments.

As use of the VA Salesforce platform grows, it will become increasingly important to manage changes to the configuration of the platform. This section describes the version control processes currently employed by the VA Salesforce platform to manage VA Salesforce metadata.

Deployment Management Plan

The VA uses a centralized model for Salesforce deployment and release management activities. Responsibility for developing and maintaining deployment management processes lies within the DTC Team. Deployment of solutions to the systems integration testing and production environments are executed by the DTC Team.

Project Teams perform a key role in the deployment processes. They ensure that deployment tasks are documented and tested, prerequisite tasks are completed successfully, and they work collaboratively with the DTC to help ensure successful deployments.

Agile Methodology

Project teams are expected to practice Scrum Agile methodology as prescribed by the Salesforce Agile Working Group (comprised of the VA Agile COE, DTC, and System Integrators). Project teams are advised to establish a Team Agreement and Work Definitions (definition of ready, definition of done) documented based on the templates defined by the Agile Working Group and communicate the documentation to the DTC.

All work/requirements should be defined as user stories and managed in an Agile Lifecycle Management Suite. User Stories must meet their work definitions to move on to various phases of their lifecycle. The Product Owner (Business Unit) involvement, LOE, and expectations can be defined in the team agreement so that all parties understand the expected level of involvement to support a project.

VA Salesforce Environment Management

To understand the deployment process, teams must be aware of the various Salesforce environments at the VA and how they are managed. The table below provides an overview of each type of environment and the supporting sandbox:

Table 2: Environment Management

Environment Type	Purpose	Sandbox Type	Deployments Performed By	Comments
Production	Provides business capabilities to production users	N/A	VA DTC Team	Currently there are five segregated Production Salesforce organizations operating in the VA environment. Administrator access to Production environments is strictly controlled.
Staging /PERF	Final pre-prod Test Environment for performance testing	Full Sandbox	VA DTC Team	Staging sandboxes are treated similarly to the Production environment. Accounts with elevated privileges are held only by the VA DTC Team. Note: These sandboxes contain Production data, and the Production security model applies.
Regression (REG)	Regression testing to ensure functionality matches functionality signed off on in SIT during User Acceptance Testing	Partial Copy Sandbox	VA DTC Team	REG sandboxes are treated similarly to the Production environment. Accounts with elevated privileges are held only by the VA DTC Team. **Note**: These sandboxes contain limited Production data, and the Production security model applies. It does not contain any PII or PHI data.
Integration (INT)	Testing of code integrated from multiple project teams	Developer	Project Teams	INT sandboxes are available to integrate test code created by multiple project teams.
System Integration Test (SIT)	System testing of code from a single project team	Developer	Project Teams	SIT sandboxes created and refreshed by DTC and are used by project teams to conduct system tests and to perform the mock deployments necessary Project Team Owned to develop and confirm their deployment checklists. User Acceptance Testing by the business team should also be completed in this environment.
Development (Dev) - Project Team Owned	Development and Unit Test	Developer	Project Teams	The DTC is responsible for development sandbox creation and refresh. Development sandboxes are not typically targets for migration. Development teams should use a sandbox to complete their initial development and keep a change log in order to migrate their changes to SIT.

The DTC is responsible for the management of all Sandboxes, including:

- **Sandbox Creation:** To request a new sandbox, project teams need to submit a Release Support Ticket (RSR) in VAPM. The DTC team determines if the sandbox supports an approved project, validates that an unused sandbox is available, and creates sandboxes for requests approved by the

***Note**: No sandboxes are distributed unless Gateway Approval has been received for the project from the DTC Product Portfolio team. To begin this process, submit a SaaS (Salesforce) Discovery Ticket in the VA OIT Marketplace.*

- **Sandbox Refresh:** To request a Sandbox Refresh, project teams need to submit a Release Support Ticket (RSR) in VA PM. The DTC team determines if the sandbox is eligible for refresh and confirms that the current use of the sandbox allows for a refresh to occur. Please note a refresh will eliminate all data in the sandbox and any in flight uncommitted development changes.
- **Sandbox Monitoring and Deletion:** The DTC team monitors sandboxes for inactivity. If a sandbox has not been used for three months and/or it is not tied to an active Module Enhancement Request, the administration team contacts the project and determines if the sandbox is a candidate for deletion.

Deployment Process

This section describes how code and configuration are deployed in the VA Salesforce environment. Requests for migrations between environments are maintained through the Module Enhancement Request object. All requests should follow the deadlines set forth in the Deployment Schedule; delays in requests may result in deployment delays or rejection of deployments for the target release month. No deployments will be approved without proper documentation and customer sign offs in the enhancement request. Should an extenuating circumstance arise, please attend a DTC integrator call (held Wednesdays at 4pm EST) or submit a Release Support Request (RSR).

Project Teams are responsible for designing and configuring their modules within their respective team development sandbox. Project teams must migrate code to their Team SIT sandbox and test. Integrators will then test in the shared environment INT by deploying their full package. It is advised that teams deploy all development changes as one package into INT. All development should be completed, and User Acceptance Testing sign off received by the time teams request to go to INT or by the Build Complete Deadline, whichever comes first. All integrator teams are expected to either have recently refreshed sandboxes and/or the latest back deploy from Production via Copado. The DTC team reserves the right to reject enhancement requests if sandboxes do not meet that criteria. When integrator teams have finished testing and have remediated any defects caused by Integration with other teams' code in INT, integrator teams should mark their stories in Copado as ready to promote.

Once integrator teams have marked their stories as 'Ready to Promote,' their enhancement has been updated to indicate that they are 'Ready for REG,' and all of the required documentation has been provided (user setup and pre & post-deployment steps in the Deployment Plan); the VA DTC team creates a single merged package and migrates the deployment to REG during the designated REG Deployment Period.

After the package has been moved to REG, integrators should test again and can't receive sign off from the business owner, or the business owner can delegate this sign off to the SI to ensure the functionality after migration and integration with other teams still matches the UAT functionality that was signed off on in SIT; this should be indicated on the enhancement request.

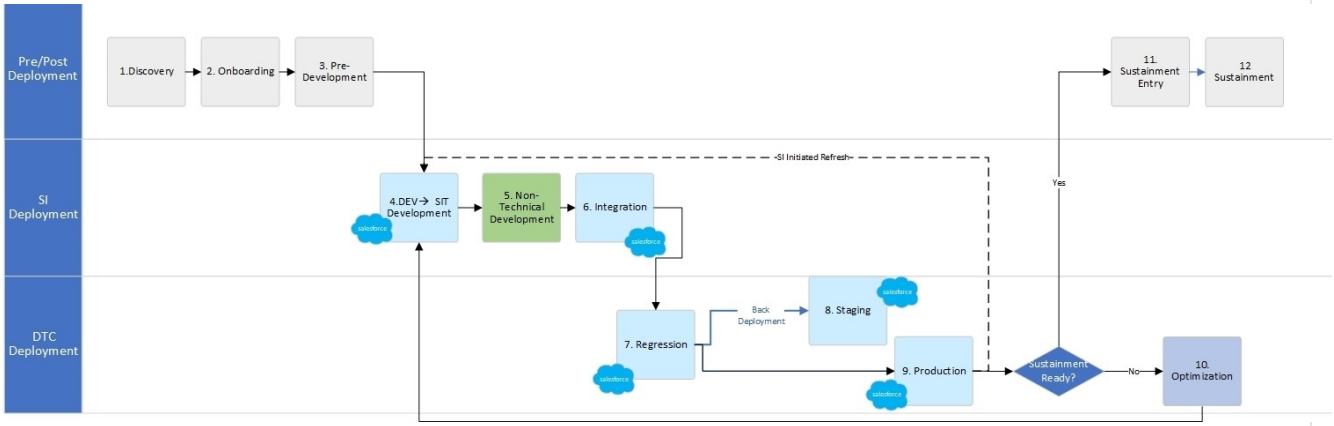
Deployment Schedule for the deadline based on the target release month. In addition to promoting to REG, the DTC team will back promote the user stories to Staging/PERF; should you have testing needs that require additional data to measure performance, you may utilize Staging/PERF as opposed to REG. When integrator teams have finished testing in REG or Staging/PERF, integrator teams should mark their User Stories as Ready to Promote in Copado. The VA DTC team will then migrate the consolidated package to Production. Teams will have a week from the Production Release Date to provide their Production Sign Off on their Enhancement Requests.

For DX and Unlocked Package associated orgs there are similarities to the main VA org's deployment process above. Similar to modules in the main VA org, deployments are driven by the creation of an Enhancement record with a selection of an associated release date based on the bi-weekly DX and Unlocked Packages Release Calendar. This calendar follows the same bi-weekly pattern as the main Salesforce Release Calendar (see below for DX and Unlocked Package calendar reference). The DX and Unlocked Package Deployment Process is referenced below. Once an Enhancement record is created and associated governance activities completed like completing any SECRISKs or architecture reviews, a Solution Integrator Team leverages version control to regularly introduce new functionality which kicks off automation that handles several quality related checks against the codebase before it automatically deploys the Team's Unlocked Package into DTC's INT environment. Completing this allows for immediate feedback early in the release cycle on whether or not a Team's codebase can successfully deploy against their targeted production environment.

DX and Unlocked Packages Deployment Process References:

- VA DX Release Calendar for DX Dependent ORGs
- VA DTC DX Release Process
- Creating an Account for MAX Federal Community (max.gov)

For additional information on deadlines for customer signoffs in enhancement requests, please reference the shared Salesforce Live Calendar called "DTC Deployment Schedule" In VAPM. Additional information can be found in the VA Development and Release Chatter Group in the VAPM org.



The figure above graphically depicts the flow of code for deployments that are conducted on the VA Salesforce Platform. Deployment Plans created by the project teams are executed by the DTC Release Team to migrate code to the consolidated REG, Staging/PERF, and Production environments.

The tasks involved in the deployment process are described in detail below. Please note that not all tasks in this table are performed for all project deployments. Additionally, rows with a gray background are not deployment tasks, but are listed to provide the context of the deployment task within the overall development process.

Table 3: Deployment Tasks

Task	Performed By	Comments
Create Development Sandbox or Scratch Org	DTC Team	When a project is approved, the DTC team creates a sandbox and provides access to project team upon request via RSR. If you require access to create scratch orgs, for a new module, you should also create an RSR to have these permissions added to your VAPM user **Note**: A Gateway Approval must be obtained prior to this step. .
Design and develop solution	Project Team	The project team configures Salesforce to meet requirements as detailed in the approved user stories.
Create and Obtain Approval on Architecture Document	Project Team	All new modules and enhancements to existing modules must create an Architecture Design document based on VA's approved template and obtain approval from the DTC team. This should take place before code is completed and it is recommended to create this as early as possible.

Create and maintain a log of configured items	Project Team	Project teams maintain a configuration log listing all Salesforce components created or updated during development. This will help the Project Team to set up Copado stories as one package when it comes time for their deployments.
Create and maintain a Deployment Plan	Project Team	The Deployment Plan lists the manual pre- and post-steps performed as part of deployment and the Copado User Stories involved in the deployment.
Deploy to Team SIT Sandbox	Project Team	Project teams follow the deployment plan and their Copado User Stories to deploy their configuration to their SIT sandbox. During this deployment, the deployment plan is validated and refined.
Perform System Integration Tests	Project Team	Testing performed in team SIT sandbox.
Remediate defects and deploy to SIT	Project Team	The project team remediates defects in DEV sandbox and migrates to SIT for each successive test cycle. Best practice is for project teams to continuously deploy to their own SIT environment and use the SIT environment for product demonstrations/User Acceptance Testing
Conduct User Acceptance Testing	Project Team	UAT is performed in the team SIT sandbox.
Deploy to the consolidated INT environment	Project Team	The project team uses the Deployment Plan to deploy to the consolidated INT environment.
Conduct Integration Testing	Project Team	Testing is performed in consolidated INT sandbox.
Remediate defects and deploy	Project Team / DTC Team	Defects are remediated in the DEV sandbox and migrated to the consolidated INT by the Project Teams. The DTC Release Team indicates the INT deployment deadline on the Release Calendar for each release month.
Conduct REG Migration	DTC Team	The DTC Team uses the deployment plan to migrate the configuration to the REG environment.
Conduct Staging /PERF Migration	DTC Team	The DTC Team uses the deployment plan to migrate the configuration to the Staging/PERF environment.
Conduct Production Migration	DTC Team	The DTC Team uses the deployment checklists to conduct the Production deployment.
Verify Production Acceptance	Project Team	Verifies Product Owner acceptance of Production Deployment.

Defect Release Management

The developer of the solution is responsible for testing the solution through the appropriate sandboxes. After testing, the developer creates Copado user story or DX package in their test environment or scratch org and validates that it works through all the other environments. If it fails anywhere, the developer is responsible for resolving the issue and returning.

Per the DTC guidelines, Go-Live and Security Package Approval (if required) will be completed prior to any deployment to Staging or Production. In addition, updates to a Technical Deployment Plan will be made and reviewed with the Release Management team.

Resolving Defects

When defects are identified, the original developer is contacted to find a solution if they are available. When the defect is resolved, it is tested through all testing stages from development to UAT. If a defect cannot be resolved, additional resources, including System Administrators, will help find a solution.

Dark Releases for New Modules

DTC supports and recommends dark releases to Production prior to going live in Production for larger or more complex modules that have multi-month development cycles.

What Does This Mean for Integrator Teams?

The module and its users are considered "dark," meaning that they are not actively creating records, data, etc. in Production. All workflow rules, process builder scripts, community, integrations, and other configurations are not visible to an active user and no real data is being created.

By pushing these configurations early and regularly, the current state of the project is integrated with existing configurations, has an interim sign off from the customer, and has met DTC standards. The resulting benefit is that this process gives the project great milestone accomplishments. In addition, the development teams save time because the sandboxes are refreshed, and the configurations do not have to be manually pushed to the sandboxes each time they are adjusted.

This same concept can be applied and utilized via feature toggles for existing applications. This can increase stability for the application in production and allow teams to control their own migration windows based on external integrations that do not occur on the same cadence as the DTC release schedule.

Deployment Technologies

Deployments on the VA Salesforce Platform are currently conducted using the following:

- **Configuration Log**: The configuration log is a complete list of all the Salesforce components created or updated in a release. Configuration logs can be maintained in spreadsheets. Project teams should ensure their configuration log is up to date throughout the development lifecycle.
- **Copado**: Copado is a Salesforce-native platform that provides visibility, tracking, and deployments in all stages of an Org (Sandbox) Development Model's release process.
- **Salesforce DX and Unlocked Packages**: Unlocked packages follow a source-driven development model and represent the source of truth for the application and not what may be in an org. This version-control driven model brings with it all the benefits of modern source-driven development practices such as "shifting-left", automation to support continuous integration/continuous delivery (CI/CD), proactive monitoring, and automated UI tests. These modern source-driven models can lead to higher performing software delivery, leading to higher performing organizations.
- **Deployment Plan**: The deployment plan defines all the activities needed to perform a deployment. It includes the following:
    - **Pre-Deployment tasks**: An ordered list of activities that must be manually performed prior to deployment. Please ensure these are only items that will cause the deployment to fail if they are not performed before the deployment. If they will not cause the deployment to fail if not complete, they can be post deployment tasks.
    - **User Stories**: Code that is committed to a branch in Git, these branches are merged and deployed in a single promotion.
    - **Post-Deployment Tasks**: An ordered list of activities that must be manually performed after deployment.
    - **Verification Steps**: A list of tasks to be executed that confirm the deployment was successful.

Version Control / Source Code Management

No Code Technical Approach

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

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

- The need to consider Salesforce 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 changing a configuration setting, rather than lines of code.
- Code coverage for testing is required to deploy APEX changes (Apex test classes 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 module is intended to do.
- Future maintenance is simplified if, for example, an individual who built custom modules leaves abruptly, picking up the pieces is much simpler if the work was done declaratively.

There are numerous scenarios in which using a declarative method may not sufficiently fulfill a business requirement, and instead may require development with code. Developing with code in Salesforce is often done based on need. For instance, when the platform needs to be extended to support functionality that simply does not exist "out of the box".

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 Workflows
- 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] - [Module 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) module, and it contains school data.

Note: Labels for custom fields are displayed to users of the module 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 states, “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.

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

Table 4: Guidelines for Preventing Potential Conflicts with Dependent Modules

Guideline	Rationale
Use 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 module 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 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

Note on Role Hierarchies: Instead of using Salesforce's out-of-the-box functionality, role hierarchy is mimicked on the VA environment by using nested public groups, called Functional Queues. Each module should design this "hierarchy" separately based on their business' use case and create groups to control read/write access to object records.

Regression Testing and Technical Debt

Regression Testing

It is important that ***ALL*** module 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 DTC, 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

DTC's Technical Debt ownership consists of (3) primary areas within Salesforce: Optimizer, Health Check and Overall System Maintenance.

- **Optimizer**: Salesforce Optimizer gives us detailed data inside org on more than 50 metrics covering everything from storage, insecure sharing settings, fields, custom code, custom layouts for objects, Rule Limits, API Versioning, Static Resource Limits, inactive/unused Validation/Workflow Rules, etc.
- **Security Health Check**: Health Check gives us the visibility into all our org's security settings and allows us to identify and fix vulnerabilities within our own security settings.
- **System Maintenance**: System Maintenance includes but is not limited to: Languishing Users, APEX tests, Object/field utilization reports, Permission Set usage, Profile usage, Role usage, Report & Dashboard usage, and several others depending on SI utilization.

In monitoring these tools on a daily/weekly/monthly/quarterly and annual basis, the DTC team determines the greatest possible risks or allocation of resources and works hand-in-hand with the SI's to determine which items need to be refactored, updated, or purged from the System.

Coding Standards

The VA Salesforce DTC considers Apex Classes, Lightning Web Components, Aura Components, 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.

- If declarative techniques can address requirements, use of code on the VA Salesforce Platform is to be avoided.
- The VA Salesforce DTC 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).
- 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.

Table 5: Naming Conventions

Type	Convention	Rationale /Example
Apex Controllers	[Organization Acronym][Module Acronym][Meaningful Class Name] [Controller]	Appending Controller to the end of name indicates that the code is an Apex Controller **Example**: VBACSSchoolsController
Lightning Web Components	[Module Acronym_Meaningful Name]	Prepending the module in front of the lightning web component clearly indicates which module uses this component. **Example**: VALERI_AcquisitionSummary
Aura components	[Module Acronym_Meaningful Name]	Prepending the module in front of the aura component clearly indicates which module uses this component. **Example**: CARMA_VeteranSearch
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.

Table 6: Test Class Guidelines

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. Respect personas by using runAs method.	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 testing scope but avoid creating test data within a method individually	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 a test utility class to ensure expected results coincide with conditions in test data.
Use assert statements logically and correctly to test for both positive and negative scenarios	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.
Tests should check for bulk scenarios	Validate code will scale with bulk DML operations
Test should use startTest() and stopTest()	Setup your testing scenario before and mark where test is actually beginning and ending

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 as well as highlight areas where the Salesforce DTC 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: Describes core Apex best practices for writing efficient, scalable code
- 15 Apex Commandments: 15 additional guidelines for efficient Apex code
- Secure Coding Guidelines: Describes common security issues Salesforce has identified while auditing modules built on or integrated with com
- Apex Developer's Guide: Provides a comprehensive guide to Apex code and syntax
- Lightning Web Component Guidelines: Provides a list of best practices to optimize the performance of your Lightning Components

Focus Areas

The following list highlights areas that project teams focus on during design and development of Apex code. These will also 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 ensure that triggers 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 is created per object. Additionally, triggers can be constructed so they invoke themselves. Project teams should validate that 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 or repeated code make maintenance more difficult. Project teams identify blocks of code that are repeated and are therefore 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, as well as highlight areas where the Salesforce DTC will focus during design and code reviews.

Visualforce References

- Visualforce Developer's Guide: Provides a comprehensive guide to Visualforce code and syntax.
- Visualforce Performance Best Practices: 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 also 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 of 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 Section 508 compliance. For example, ensure that all tables have summary text and graphics have Alt Text set appropriately.

Lightning Web Components

Lightning Web Components uses core Web Components standards and provides only what's necessary to perform well in browsers supported by Salesforce. Because it's built on code that runs natively in browsers, Lightning Web Components is lightweight and delivers exceptional performance. Most of the code you write is standard JavaScript and HTML.

- Lightning Web Developer Guide
- Lightning Web Component Guidelines: Provides a list of best practices to optimize the performance of your Lightning Components

VA Salesforce CCB, Design Reviews and Code Reviews

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 processed. 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 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.

Design Reviews

Project teams must have their designs reviewed and approved prior to the start of development. Designs will be reviewed at the weekly Architecture Design Review meetings on Thursdays at 10:15 am. Prior to a review, the team must complete an Architecture Document using the template found in the VA System Integrators Chatter group.

Code Reviews

Prior to deployment to the System Integration Test (SIT) or Integration Testing (INT) environments, shared components of a module's code must be reviewed by the CCB. Prior to deployment to production, all issues noted in code reviews must be resolved.

Code Review Checklist

This section provides a framework project teams to evaluate code. Items on this checklist are verified during code reviews.

Item #	Description
1	Is all code well-structured, consistent in style, and consistently formatted (Are 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 an “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 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 / Lightning Web components accounted for 508 compliance? (e.g., Do all graphic components have 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?
30	Is the committed work traceable? If no, add traceability to your commit statement by using comment format "<JIRA Issue #> - <a brief sentence summarizing your commit>"

Revision History

Date	Version	Description	Author
9/22/2021	3.0	Updated to reflect current process	Maura Vitelli
10/02/2020	2.0	Updated to reflect current process.	Karina Cox
02/27/2018	1.0	Published.	Christine Talbot