DTC Standard Operating Procedures for the VA Salesforce Platform

Purpose

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.

Scope

This SOP is applicable to all staff involved in the Digital Transformation Center (DTC) and work within the Salesforce Platform.

Roles and Responsibilities

Information System Owner

- Ensure adherence to access control policy by key OIT staff
- Provide oversight for personnel with significant responsibility to information systems.
- Update procedures as needed to ensure it meets OIT mission requirements and complies with Federal Laws, Policies, Procedures, and Guidelines.
- Ensure all procedures herein are properly complied with.

Information Systems Security Officer (ISSO)

- Monitor and audit the Access Controls to ensure adherence to VA Knowledge Service.
- Perform monthly New User Access Request reviews.
- Perform quarterly Privileged Account reviews.
- Perform quarterly Separated User reviews.
- Perform monthly Inactive Account Reviews

Account Managers/OIT Support Staff

- Manage the entire lifecycle of accounts (create, enable, modify, disable and remove) for the information systems to ensure compliance with account management.
- Promptly taking appropriate actions on accounts as requested.

Supervisors

- Designate individuals to act as a liaison between OIT Account Managers and the service to authorize and submit access requests.
- Ensure security requirements (training, ROB, and background investigation) have been met prior to submission of access request.
- Promptly report changes to OIT staff on a user’s employment status or change of duties to ensure all accounts are acted on appropriately.

COR

- Function as the designated individual for respective service to submit access requests.
- Promptly report changes to OIT staff on a user’s employment status or change of duties to ensure all accounts are acted on appropriately.
- Ensure all access requests comply with background investigation completion, training and Rules of Behavior (ROB) completion prior to submitted access request.

Platform User Access Model

The DTC platforms will adhere (as closely as possible) to the access model below. Adjustments (by platform) to this User Access Model can be implemented with approval by DTC Leadership.   All platforms will adhere to a least privilege model, applying only the base requisite role and all corresponding access, permission and privileges as necessary

PaaS Platform Roles

System Administrator

Has access to manage the platform environment, applications, and services deployed on the platform. Will have few limits to scope and access. Might have access to User Management, Platform Setup and Configuration, Feature Provisioning, and Monitoring and Logging

**System Administrators could have access to “Read All Data” with an approved EPAS.*

VA Administrator

May have access to view the platform setup and configuration. Role could include application and module support requiring global access and a broad scope. Could have access to User Management and other platform support features.

**VA Administrators could have access to “Read All Data” with an approved EPAS.*

Business Owner

Business owners or module administrators will have global access within the scope of a single application. They will not have access to platform level configuration. Their data access and permissions will provide the greatest capabilities within an application or module but will provide no access to other applications or modules.

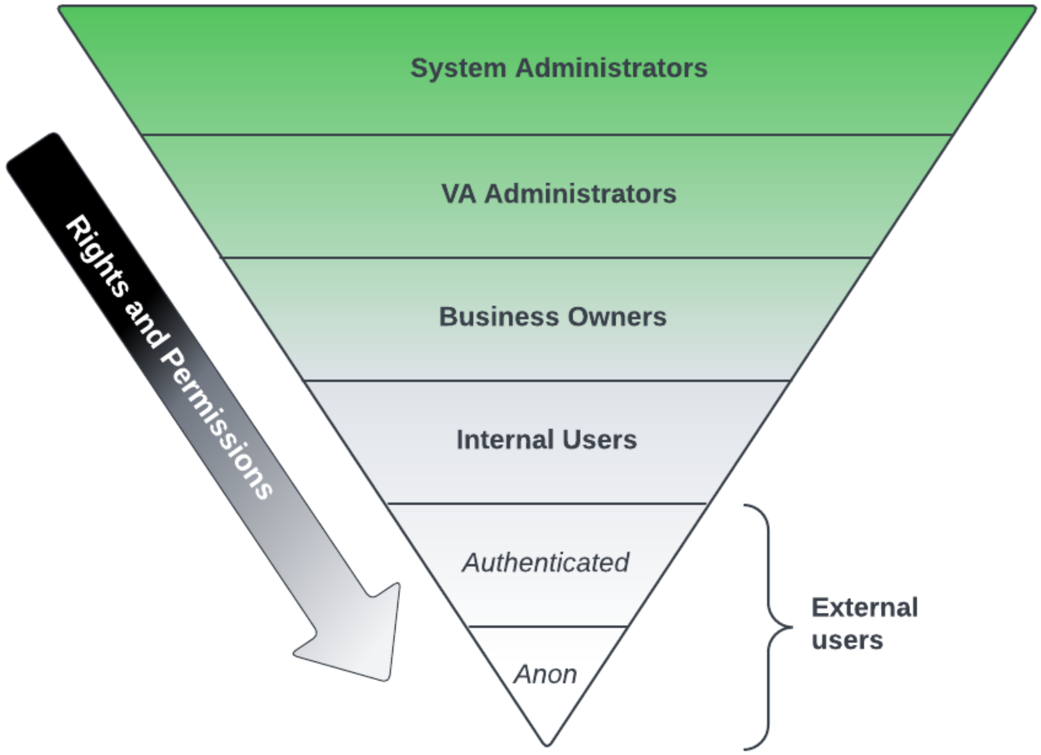
Internal VA Users

Internal User could be aligned to numerous sub-roles that will provide varying levels of access within a given application or module. The application-level roles will be defined by the personas and use cases based on the application requirements. Their access should be limited to only those permissions necessary for their role and will provide no access outside of their given module or application.

External Users

External Users are users who are external to the VA organization and will come in two flavors:

- Authenticated Users – which are external users that access the platform via a VA approved authentication service (MyHealtyVet, ID.. me, etc…). Will have limited access to view any data and will have only have access to the externally exposed features for their applied application persona or role.
- Unauthenticated Users – which are external users that will access the platform anonymously. These users should have no access to view "non-public" data and should be limited to only forms intended to capture data.   Anonymous access should be avoided unless absolutely necessary and should be limited to accessing systems only from the VA network when possible..



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 has an internal Salesforce license.
User Community	Tied to an individual who has 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.
Service User	There are a few service accounts used to support various administrative tasks
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.
Elevated Permissions User	An individual assigned System Administrator or equivalent Profile in a Production org or a full copy sandbox (i.e. with Prod data).

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. Description which includes 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. The specific module the user needs access to 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 have their permissions updated based on the user setup instructions for their requested role while 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. The following section outlines the requirements for an Elevated Permissions User.

Elevated Account Management:

Elevated privilege (System Administrator) access will only be provided to those with a proven and absolute need for the access to do their job. The following requirements for Elevated Permissions are requested and stored by the DTC EPAS Team.

1. Background Investigation Dates – requested from and provided by B3 FSO; received and stored by the DTC EPAS in VA Teams.
2. Completed course certificate for TMS #1357076 Information Security Role-Based Training for System Administrator – requested from and provided by user; received and stored by the DTC EPAS in VA Teams.
3. User’s Supervisor approval for requested Elevated Permissions profile; received and stored by the DTC EPAS in VA Teams.
4. COR review and approval of EPAS form (MyVA Elevated Privileges) – reviewed, approved, and stored in the VA EPAS system.
5. The Salesforce System Owner is provided a weekly report of new Elevated Permission Users – provided by the DTC EPAS Team.

Elevated Account Maintenance:

Once one year has passed from submission, the DTC EPAS team completes the supervisor and steps again and submits a new EPAS.

Full process and steps are captured in the DTC EPAS SOP for Salesforce, available at the following link: https://us.promapp.com/b3groupinc/Process/Minimode/Permalink/FJ54MbXhKuWTkhr1nujqn

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

Community Users are deactivated via the Automated Deactivation Process. Therefore, they will be deactivated 45 Days after their “Last Login” date.

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 45 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. Audits of User Deactivation, Reactivated Users and migration of Users (not logged in for 180 Days) to the Minimum Access Profile will take place on a regular basis.

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.

Current audit Standard Operating Procedures include the following reports reviewed weekly, unless otherwise needed:

1. Report of “Users Deactivated Last Night”
2. Report of “Users Reactivated within the Last Week”.

3. Report of “Users Deactivated Over 45Days Ago”.

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 backups are now automated with incremental backups happening daily. Export includes data backup only.

Additionally, audit Logs are downloaded every week and saved into the corresponding backup folders.

Preethi Thota and David Fry have access to the folders.

Backup Org’s information:

Org	Backup Schedule
VAPM	5:00 AM UTC/1:00 AM EST
VAEMPL	6:00 AM UTC/2:00 AM EST
VHA-GOV	7:00 AM UTC/3:00 AM EST
VALOB	7:30 AM UTC/3:30 AM EST
VA-VET	8:00 AM UTC/4:00 AM EST
VA-MAIN	8:00 AM UTC/4:00 AM EST

Risks with the current backup process:

- Data and Files are backed up independently. Data is hosted in an AWS SQL RDS. Files are hosted in AWS S3 buckets.
- Data storage constraints do not allow for snapshot backups. Backups only consist of a single active backup per org.
- Deleted data in Salesforce will be deleted from the backup after 30 days.
- Salesforce Weekly Exports don’t include metadata. Metadata exists in GitHub repositories.
- The system supports automated restoration, but current processes and data ownership require restoration be manual.

Data Archiving

In accordance with NARA guidelines, VA exports data for decommissioned modules directly from the Salesforce backup system. See the data archival policies here (link TBD)

Any questions or concerns please raise an Architectural Support Request or reach out to DTC Architecture at VADTCSalesforceArchitecture@va.gov

Application Modernization Standards & 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 in a timely manner.

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 provide 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 seven 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/Perf 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 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. ***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 and the Release object in Copado. No deployments will be approved without proper documentation and customer sign offs in the enhancement request.

Project Teams are responsible for designing and configuring their modules within their respective team development sandbox. Project teams must go through the CodeScan Quality Gate and Code Review process before they can 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 are ready to deploy to Reg and Staging/Perf. 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 those 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 (RTM, BO Sign off, Deployment tasks attached to user stories, release record created); the VA DTC team reviews user stories on daily Go/No Go call, and deploys to next environment (REG/Staging).

After the package has been moved to REG, integrators should test again and 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. 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 and attend the Go/No Go call. The VA DTC team will then deploy 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. 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

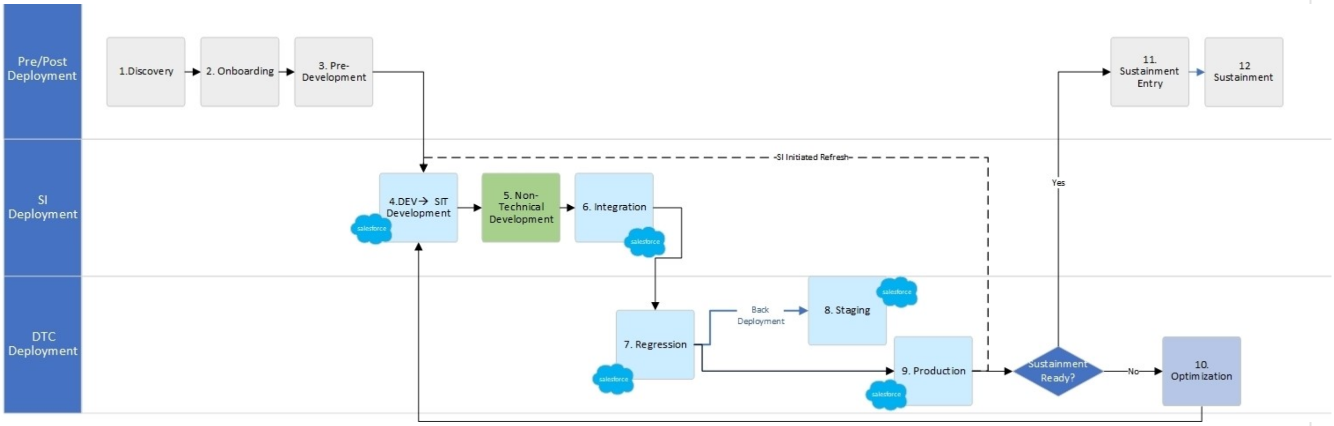


Figure 1: Deployment Process

The figure above graphically depicts the flow of code for deployments that are conducted on the VA Salesforce Platform. Deployment tasks 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.

Table 3: Deployment Procedure. See Readiness Release Checklist for more information.

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 Deployment Tasks	Project Team	The Deployment Tasks are manual pre and post-steps performed as part of deployment and are listed on their copado user story.
Deploy to Team SIT Sandbox	Project Team	Project teams are required to pass the CodeScan Quality Gate, Code Review process, and pull request process prior to deploying to their SIT sandbox.
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 will deploy to the consolidated INT environment.
Conduct Integration Testing	Project Team	Testing is performed in consolidated INT sandbox.
Remediate defects and deploy	Project Team	Defects are remediated in the DEV sandbox and migrated to the consolidated SIT and INT by the Project Teams.
Conduct REG Deployment	DTC Team	The DTC Team uses the release records, enhancement requests, and Copado user stories to migrate the configuration to the REG environment.
Conduct Staging /PERF Deployment	DTC Team	The DTC Team uses the release records, enhancement requests, and Copado user stories to migrate the configuration to the Staging/Perf environment.
Deploy to Production	DTC Team	The DTC Team uses the release records, enhancement requests, and Copado user stories to migrate the configuration to Production.
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 deploying a bug fix.

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 Deployment for New Modules

DTC supports and recommends dark deployments to Production prior to going live in Production for larger or more complex modules.

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 Tasks**: The deployment tasks 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.
  - **Copado 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.

Source 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.

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**: The "Contact" object will require special scrutiny before adding record types. For other objects, it is important to record type 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 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: Due to the multi-module (PaaS) implementation of the VA Salesforce platform, the use of the out-of-the-box functionality, "Role Hierarchies" is prohibited. Alternitively, the VA supports record sharing by using nested public groups, called Functional Queues. Each module should carefully design this "hierarchy" separately based on their business' use case (ensuring least privilege is still respected) 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.

Regression tests should be performed by the project team that originally developed and tested the capability. For modules in sustainment, 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

One of the greatest responsibilities of the DTC is to reduce the VA Salesforce platform level of Technical Debt. The DTC ownership of technical debt is limited to platform level issues as well as module issues for those modules identified as “in sustainment”. However, the DTC is responsible for identifying technical debt within the entire ecosystem. The following tools are used to identify technical debt within the VA Salesforce ecosystem:

- **Salesforce 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.
- **Salesforce Releases and Updates**: Salesforce has releases and updates that can bring new functionality but also deprecate legacy functionality or change how existing functionality operates, which requires review to understand the impacts to the platform.
- **Security Reviews and Assessments**: The DTC and the VA security team performs recurring as well as manual reviews of Permission Set usage, Profile usage, as well as other platform and module level functionality to identify concerns or poor practices that could create vulnerabilities.
- **Code Review**: The DTC performs code reviews for all internally and externally developed features. These reviews can identify existing technical debt as well as newly created tech debt that requires remediation.
- **Salesforce Signature Support**: The VA DTC has a partnership with Salesforce Signature Support which supplies reviews of our Salesforce platform on a regular basis. These reviews provide information regarding concerns around the following areas:
    - Overall Configuration.
    - Code Quality
    - Code Performance
    - System Security
    - System Scalability
    - Overall Recommendations

In leveraging these tools on a daily/weekly/monthly/quarterly and annual basis, the DTC team prioritizes the technical debt that is of the greatest risk and allocates resources or works hand-in-hand with the SI’s to create plans and establish timelines for remediation.

DTC Coding Standards

This section provides guidance regarding Coding Standards supported and enforced by the DTC. Governance around coding standards will be applied via code reviews and automated code scans.   This area of governance is a VA requirement per the DTC ATO in an effort to ensure the delivery of quality software that provides the requisite security, performance, and scalability on our multi-module platform.

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 at scale, use of code on the VA Salesforce Platform should be avoided
- The VA Salesforce DTC can work with project teams during design to identify declarative alternatives to code if needed
- Some code formatting is stylistic in nature and as such is not covered in this document.   Though the following concepts are encouraged:
  - Intelligent use of white-space to aid in readability
  - Declarations made at the beginning of their respective blocks
  - Opening curly brace should be on the same line as the declaration statement
  - Curly braces should always be used and never omitted when optional
  - Line Wrapping used to aid in organization and readability but not overused

Guidelines for Comments and Naming

Classes and methods use standard and meaningful naming conventions with comment blocks. 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.

Module acronyms should be separated from the rest of the component name by an underscore. For module name changes, consistency is preferred over accuracy. If your module's name is changed please keep using the existing prefix.

Type	Convention	Rationale/Example
Apex Classes	[Module Acronym]_[Meaningful Class Name] in Pascal Case	Class name should be meaningful and indicate the module for easy reference Example: VBA_CSSchoolsUtilities
Trigger Action Class	[Module Acronym]_TA_[Object][DML Contexts (optional)] in Pascal Case	Class name should include the object name. Contexts are optional but will help maintainability. Wherever possible, combine contexts into a single class Example: VBA_TA_CustomObjectWhatItDoes
Apex Controllers	[Module Acronym]_[Meaningful Class Name][Controller] in Pascal Case	Appending Controller to the end of name indicates that the code is an Apex Controller Example: VBA_CSSchoolsController
Lightning Web Components	[Module Acronym]_[Meaningful Class Name] in Pascal Case	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 Class Name] in Pascal Case	Prepending the module in front of the aura component clearly indicates which module uses this component. Example: CARMA_VeteranSearch
Test Classes	[Module Acronym]_[TestedClassName /FunctionName][Test] in Pascal Case	Appending Test to the end of the name indicates the code is a Test class. Example: VBA_CSSchoolsControllerTest
Triggers	[ObjectName][Trigger] in Pascal Case	Prepending the object name in front of the trigger clearly indicates the object on which the trigger is acting Example: VBA_CSSchoolsTrigger
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)

Constants, Enumerations	Meaningful name in UPPER_CASE and be descriptive	Use upper case for constants or enumerations to clearly identify their use and to distinguish them from variables. Example: INDIVIDUAL_RECORD_TYPE

Casing Definitions

- Pascal Case - First character of all words are Upper Case and other characters are lower case.
  Example: BackColor
- Camel Case - First character of all words, except the first word are Upper Case and other characters are lower case.
  Example: backColor
- Upper Case - All characters of all words are Upper Case using underscores to separate words
  Example: BACK_COLOR

Comments

Classes and methods should contain an information block following the ApexDoc format.   Details about the ApexDoc format can be found here: Salesforce Foundations: ApexDoc on GitHub

Class Comment Block

Class Comment Blocks should contain details about the creation of the class and a good description of what the class is for. Including a changelog in the comments is optional.
Information Block Format

```
/**
* @author [Author Name]
* @date [Creation Date]
*
* @description  [Brief description of the class and what it does]
*/
```

Method Comment Block

Method Comment blocks should contain details about the creation of the class and a good description of what the class is for. Including a changelog in the comments is optional.
Information Block Format

```
/**
* @description [Brief description of the method]
* @param [parameter 1 defined in method declaration]
* @param [parameter 2 defined in method declaration]
* etc... (continue for additional params)
*
* @return [explanation of the methods return value]
*/
```

Inline Comments

Inline comments should be made to provide detail about the complex operations happening within the method. It is unnecessary to comment every action that is easily explained by looking at the code (e.g. No need to comment variable declaration)

Information Block Format

```
/**
 * preferred way to comment in a method
 **/
public void doSomething() {
    boolean isValid = false;
    Integer recCount = 0;

    //iterate through list of objects created this week to check validity
    for ( Object p : objectList ) {
        if ( condition ) {
           isValid = true;
        }
        else {
            isValid = false;
        }
    }
}
```

Guidelines for Test Classes

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

Guideline	Rationale	Security-Related
Design and develop test classes ahead of or shortly after the code to be tested is written	Test Driven Development is a solid way ensure business logic is well established and tested, though not always feasible under time constraints. Writing your test methods in conjunction with your tested code aids in creating effective testing of expected conditions and enforced inclusion of error handling	
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. Aligning your test methods with "Service Layer" methods is a great way to test business use cases.	
Simulate business functionality using System.RunAs	Run all DML and verifications in test methods as a business user. Do not rely on the user running the test to have appropriate permissions.	
Create test data within test classes and methods	All test data should be created with test classes to ensure expected results coincide with conditions in test data. Create testing utility classes for reusable data creation methods	
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.	
Do not use @isTest (SeeAllData=true)	This could allow someone to remove or modify data that they should not have access to.	

Anonymous Apex Scripts Review Policy

Technical oversight and reviewing anonymous Apex scripts before running them in Salesforce is important for several reasons:

- **Security:** Anonymous scripts can contain malicious code that could compromise data integrity, leak sensitive information, or create security vulnerabilities within the Salesforce environment.
- **Data Integrity:** Running unreviewed scripts can lead to unintended data modifications or deletions, including beyond the module data they’re intended for, which might be difficult or impossible to recover from.
- **Scalability:** Scripts may not have been properly designed to run at production-scale and are not able to accomplish their intended purpose with the large quantity of data or require numerous attempts to rerun scripts or worse, requests for the person running the script to manually adjust parameters on the fly.
- **Compliance:** ensure we are following compliance and governance policies that require code reviews to ensure adherence to standards and regulations.
- **Error Prevention**: Reviewing scripts helps identify and correct errors or logic flaws before execution, preventing potential disruptions to business processes.
- **Auditability:** Having a review process ensures there is a record of what code has been executed, which is essential for troubleshooting and audits.

Overall, reviewing anonymous Apex scripts helps maintain the security, stability, and integrity of the Salesforce environment. An alternative to writing and running anonymous Apex scripts in Salesforce is to use standard development processes that provide more control, structure, and safety. DTC recommends the use of utility Apex classes even if only executed one time in production.

- Apex classes allow for better organization, testing, and deployment of code. These are stored in the Salesforce metadata and can be managed through the development lifecycle.
- Execution errors are minimized as large blocks of code are not copy and pasted which can lead to errors. You can be certain of exactly what code was run.
- Unit Tests must be written that provides validation that your code is executing as intended and with appropriate permission sets for your module.
- Queueables, batch apex and other processes designed for high data-volumes can be used to ensure scalability.
- Nebula Logger and other logging utilities can be used to capture output reducing the need for error prone copy and pasting of system debug logs.

Using these alternatives helps ensure code is properly reviewed, tested, and deployed, improving overall system reliability and security. That said, if you have a valid use case to utilize anonymous apex scripts, you may take the following approaches to receive approval to run in full copy /production environments:

1. To request preemptive review of your anonymous Apex scripts, please submit an Architecture Support Request. Once approved, you may include a comment in your script to refer to the case # for appropriate approvals.
2. If not preemptively approved, the DTC Release team will send anonymous apex scripts to the architecture team for approval during your deployment to REG and Staging/PERF. Note: this can cause some delays in your deployment while we are waiting for approval.

Requirements and Best Practices

- A clear description of Business Need as well as Business Owner approval is required for any script execution.
- Always include a description of the necessary permission sets for executing and scripts as administrators likely do not have the necessary object and field level security to support running them by default.
- If the script is used to schedule an ongoing process (eg. Scheduled Apex) or if there is an appropriate Process User the script should be run as, include that in the request.
- Any script should include minimal logic. Ideally it will be a one line script. Example: MOD_OneTimeJobs. populateExistingRecordLookups();
- If a job is truly intended to be used one time use class comments to indicate when it should be removed and proactively put a story in your backlog to remove the code when no longer needed.

Rule Severity "Translation"

PMD Severity	CodeScan Severity	VA Severity	BLOCKS PULL REQUESTS
1	Blocker	Critical	**YES**
2	Critical	High	**YES**
3	Major	Moderate	
4	Minor	Low	
5	Info		

Code Reviews

Code reviews will be performed manually as well as systematically using Linting tools such as Sonarqube, eslint, CodeScan and/or PMD.

The following points are meant to provide the guidance being enforced by the DTC during code reviews.

Some points, marked as required (Reject Reason column = Y), will result in the rejection of a given code branch and will require remediation before it can be promoted to upper environments.

Other points will be enforced less rigidly and could end up in an exception being logged for future remediation or simply logged as tech debt.

Apex

Category	Issue	Description	Impact	Security-Related	Reject Reason?	Effective Date
Code	SOQL in loop	Do not put SOQL statements within a loop to avoid the "Too many SOQL Queries" error	Critical		Y	09/29/2022
Code	SOQL Injection Vulnerability	Always use static queries or bind variables as well as ***String.escapeSingleQuotes()*** to guard against SOQL injection	Critical	Y	Y	09/29/2022
Code	Enforced Sharing	Explicitly callout ***with sharing***, ***without sharing***, or ***inherited sharing*** to be purposeful in your intent. Avoid "Without Sharing" unless necessary	Major	Y		09/29/2022

Code	Without Sharing - insufficient modularization	If the without sharing is used, it should be limited to just the methods/functions that require it, not for an entire set of functions - especially for external facing pages/components	Major	Y		09/29/2022
Code	Comments	Comment blocks and within the code should be included as they can help with readability and future maintenance of code	Informational			09/29/2022
Code	DML in loop	Do not put DML statements within a loop to avoid the "Too many DML Statements" error	Critical		Y	09/29/2022
Code	Dereference array without size checking	This can cause an error if you have an insufficient number of items in the array	Major			09/29/2022
Code	Assigning query result to a non-list	Queries almost always return a list of things. Be mindful to always assign your queries to lists unless limiting to a single record programmatically	Major			09/29/2022
Code	SOQL with no where clause / limit	All SOQL Queries should be selective and performant. Use ***Limit*** or ***Where*** clauses to avoid record limits. (***Where*** clauses should search against indexed fields if possible)	Critical		Y	09/29/2022
Code	SOQL without explicit user/system mode	SOQL should specify WITH USER_MODE and should check field/object accessibility. SYSTEM_MODE should only be used with approval.	Major			09/29/2022
Code	Hardcoded Values	No IDs, Names, Picklist Values, Descriptions, Strings, Numbers, etc. are hard-coded. Favor custom settings/labels/metadata when appropriate (hardcoded IDs will be rejected)	Critical	Y	Y	09/29/2022
Code	System.debug statements	Production code should not include debug statements - these should be cleared out before going to production	Major			09/29/2022
Code	Logic Issues	Avoid bad logic scenarios, Unreachable code, Endless Loops, No If/Else etc...	Major			09/29/2022
Code	Invalid use of @future	Avoid invalid use of @future annotation to avoid hitting limits as well as blocking other actions due to daisy-chained @future callouts	Critical			09/29/2022
Code	Callouts in loop	Do not put callouts within a loop to avoid transaction or 24 hour limits for callouts	Critical		Y	09/29/2022
Code	Insufficient Modularization	Code could have been written in a more modularized manner for reusability and readability	Informational			09/29/2022
Code	Email setTargetObjectId to User	When sending a ***SingleEmailMessage***, always apply a user record to the ***setTargetObjectId*** *parameter to avoid it counting against org limits*	Critical		Y	09/29/2022
Code	Code not Bulkified	Use bulkified code working with collections as opposed to single records to avoid limits and excess processing	Critical			09/29/2022
Code	Insufficient Modularization	Code could have been written in a more modularized manner for reusability and readability	Informational			09/29/2022
Code	Require curly braces	All if, else, loops, etc. must use curly braces, even for a single line execution. This prevents accidental errors when a second line is added but no braces are included.	Critical			09/29/2022

Trigger

Category	**Issue**	**Description**	**Impact**	**Security-Related**	**Reject Reason?**	**Effective Date**
Triggers	Trigger Framework	Always use Trigger Action Framework if available in your org	Critical		Y	09/29/2022
Triggers	Avoid Multiple Triggers	In orgs without the Trigger Action Framework (or legacy app updates) only 1 unmanaged trigger per object is allowed	Critical		Y	09/29/2022
Triggers	Logicless Triggers	Triggers should not have any logic in them. Logic should be in the handler class and work should be performed in the helper class	Critical		Y	09/29/2022
Triggers	Recursion Handling	Recursion handling should exist within the trigger	Major			09/29/2022
Triggers	Trigger Bypass	All triggers should support the requisite bypasses either via custom settings or permissions (see trigger framework documentation and put "DisableTriggersFlag" in sObject Trigger Setting Bypass Permission field)	Critical		Y	09/29/2022
Triggers	Before Triggers	Updates to the current record should always be made in the Trigger Before context	Major			09/29/2022

Visualforce

Category	Issue	Description	Impact	Security-Related	Reject Reason?	Effective Date
Visualforce	Cross-Site Scripting	Avoid XSS	Critical	Y	Y	09/29/2022
Visualforce	Headers	All files should have a header	Major			09/29/2022
Visualforce	Parameter Cross-Site Scripting	Sanitize all parameters in Apex or Visualforce	Critical	Y	Y	09/29/2022

Error Logging

Category	Issue	Description	Impact	Security-Related	Reject Reason?	Effective Date
Error Handling	Error Handling Framework	Always use Nebula Error Logging Framework if available in your org	Critical		Y	09/29/2022
Error Handling	Try/Catch Locations	Try/Catch statements should exist as high in the stack as possible	Informational			09/29/2022
Error Handling	Error Handling	All errors should be handled in a meaningful way. Either catch and expose the error or log the error and move on. Try not to use generic Exception; specify the exception type you are catching.	Major			09/29/2022

Lightning Web Components

Category	Issue	Description	Impact	Security-Related	Reject Reason?	Effective Date
Lightning Web Component	Limit Server Actions	Calls to server are limited (pass info between components when possible)	Informational			09/29/2022
Lightning Web Component	Limit SOQL Scope	Queries only contain columns in SELECT that are used/needed (excepted when using data factories)	Informational			09/29/2022
Lightning Web Component	UI Optimization	Use pagination, SOQL ***Limit*** + ***Offset*** when appropriate	Informational			09/29/2022
Lightning Web Component	Unnecessary Data Loads	Avoid unnecessary data loads by using "Lazy Load" pattern or data caching to only load data as/when needed	Informational			09/29/2022
Lightning Web Component	Manage Event Scope	Do not define events at a great scope than necessary (e.g. Using Pub/Sub to pass events from parent to child components)	Informational	Y		09/29/2022
JavaScript	No 3rd Party JavaScript	Do not use any 3rd party JavaScript libraries without previous approval	Critical	Y	Y	09/29/2022
JavaScript	Vulnerable JavaScript	Do not use JavaScript libraries that are known to be insecure	Critical	Y	Y	09/29/2022
JavaScript	No Console	Do not use **console.log**, **console.error**, etc. as this is a security risk	Critical	Y		
JavaScript	Require curly braces	All if, else, loops, etc. must use curly braces, even for a single line execution. This prevents accidental errors when a second line is added but no braces are included.	Critical			

Test Classes

Category	Issue	Description	Impact	Security-Related	Reject Reason?	Effective Date
Test Class	Meaningful Test	Test are meaningful and cover positive, negative and bulk scenarios validating expected outcomes	Critical		Y	09/29/2022
Test Class	User Uniqueness	Test user may overlap with existing users if do not have a timestamp-based name for uniqueness & cause errors running the script	Critical		Y	09/29/2022
Test Class	Start/Stop test location	***Test.startTest()*** and ***Test.stopTest()*** should be used and should surround JUST the method being tested & in order to test the governor limits correctly				09/29/2022
Test Class	Asserts missing	***System.assert()*** must be included in all test methods to support meaningful tests	Critical		Y	09/29/2022

Test Class	TestSetup annotation	Consider using the ***@testSetup*** annotation to help with data setup for multiple methods & speed up test class execution				09/29/2022
Test Class	View All Data	No test classes should leverage the ***SeeAllData=true*** annotation	Critical	Y	Y	09/29/2022
Test Class	Test run in user scope	All tests should be run in the correct user scope. Create the user with the appropriate permissions for testing and use ***System.runAs()***	Critical	Y	Y	3/01/2024
Test Class	Test code coverage	At least 75% of every trigger and class must be covered, and 79% of all code must be covered. Goal is to achieve 85% by January 31, 2024.	Critical		Y	09/29/2022
Test Class	Insufficient comments	Test should include appropriate comments (block and inline) to aid in understanding what is being tested and the expected outcome				09/29/2022
Test Class	Integration Testing	Tests leverage mock testing framework to test web service callouts				09/29/2022

Object Definition

Category	Issue	Description	Impact	Security-Related	Reject Reason?	Effective Date
Object Definition	Field Data Sensitivity Level	All new fields should have a Data Sensitivity Level provided	Critical	Y	Y	09/29/2022
Object Definition	Field Compliance Categorization	All new fields should have a Compliance Categorization provided	Critical	Y	Y	09/29/2022
Object Definition	Field PII / PHI Correctly Identified	Any field to contain PHI or PHI must be categorized as PII or HIPAA with a minimum sensitivity of Confidential	Major	Y		09/29/2022
Object Definition	Field Description Object Description	All objects and fields should have a Description provided	Critical		Y	09/29/2022

Best Practices, Tips and Other Info

The section provides some useful information and tips regarding the guidance provided above.   This section will be expanded as new content is found to be valuable and available but is meant to be informational only.

Frameworks

Below are some helpful points about the frameworks used and enforced via our coding standards

Trigger Action Framework

- Why did we choose the trigger action framework? See Decision: Trigger Framework
- Trigger Action Framework; GitHub repository and documentation (does not have DTC updates): https://github.com/mitchspano/apex-trigger-actions-framework
- Apex Hours / Mitch Spano overview of Trigger Action Framework with video: https://www.apexhours.com/trigger-actions-framework/

Nebula Logging Framework

- Why did we choose the Nebula Logger for our logging framework? (Documentation in development)
- Nebula Logger DTC GitHub repository with recipes: https://github.com/department-of-veterans-affairs/va-nebula-logger
- Nebula Logger non DTC GitHub repository and documentation: https://github.com/jongpie/NebulaLogger
- SalesforceBen Nebula Logger "how-to" and review: https://www.salesforceben.com/easily-debug-salesforce-using-nebula-logger/

Apex Coding Tips

Below are some thoughts, tips and tricks for special coding scenarios

Null Handling

- Understanding why null handling is important is of great use. Salesforce What Can NULL Do to Me?

- Salesforce now supports null safe handlers, use to them to avoid getting ***NullPointerException*** errors. Salesforce Safe Navigation Operator

- Managing nulls in your SOQL queries is a great way to reduce additional efforts to manage nulls.   Salesforce Using null in SOQL Queries

Secure Coding

- Tips for mitigating SOQL injection vulnerabilities: Salesforce SOQL Injection

- Documentation on apex security and sharing: Salesforce Apex Security and Sharing
- Direct link to Salesforce security class documentation for enforcing secure data access: Salesforce Security Class

Asynchronous Processing

All too often, we see asynchronous solutions implemented for the “wrong” reasons. When reviewing whether or not to implement processing asynchronously please consider:

- The different types of asynchronous processing and when to use each: Salesforce Asynchronous Apex

- Salesforce governor limits with regard to asynchronous processing. These limits are for the entire org and are shared with all asynchronous Apex**:** Batch Apex, Queueable Apex, Scheduled Apex, and Future methods: Salesforce Execution Governors and Limits

Performance

Below are some thoughts as to how to ensure your solutions as performant as possible.

- When writing your SOQL meant to read over large volumes of data, use the Salesforce "Query Planning Tool" in the dev console to evaluate the performance of the query and optimize: Salesforce Developer Console Query Plan Tool FAQ
- Understand your options for data access when writing your Lightning Web Components such as: Salesforce Lightning Data Service
- Salesforce has limited caching options but they do exist. Understand the Salesforce caching options (Cache access has known to be inconsistent, ensure your use case is a very good fit) Salesforce Platform Cache
- Salesforce provides a tool set that can analyze platform usage and provide suggestions. While not a comprehensive view of custom solutions, it's encouraged to run in development orgs every so often. Salesforce Run the Salesforce Optimizer App

Analytics

It is highly suggested to move Business Intelligence solutions to true BI environments. Salesforce has partnered with Tableau to overhaul Einstein analytics into the new Tableau CRM platform. Below are helpful considerations for the development and deployment of Salesforce Analytics/Tableau CRM components.

- Datasets are powerful data collection utilities but can be problematic and cause platform throttling if not designed correctly.   It is good to understand the considerations before employing datasets: Salesforce Considerations Before Integrating Data into Datasets
- Dashboards are a key component to analytics, here are some best practices for Tableau CRM dashboards: Salesforce Best Practices for Building Your Own CRM Analytics Dashboard

Definition of Done

In an agile environment the "Definition of Done" is more of an abstract concept than most would think. "Done" is relative to the effort and will have different definitions based on scope.   The definition below is meant to define "Done" for a single sprint before promotion to DTC managed higher environments

Governance Artifacts in Place

- Security Risks identified, logged as a secRisk, and approved by VA Security Team
- Design Record created with design document attached, reviewed and approved by DTC Architecture
- Release record created with any pre/post deployment steps clearly defined

Code Complete and Tested

- All bugs identified, remediated, or properly documented
- Static Code Analysis performed with “Blockers” or “Critical” vulnerabilities remediated
- Unit Tests with 85% or greater code coverage successfully ran and complete
- Branch code conflicts resolved.

QA Testing Complete

- Regression Testing for backwards compatibility complete
- Compatibility Testing on multiple browsers complete
- Performance Testing with large data volumes complete
- 508 Compliance Testing complete as required

Code Review

- Code Review complete and any issues found (via manual review or automated code scans) that are considered blocked "Reject Reason = Y" have been remediated

Conclusion

Though this checklist is a tool meant for code review and unifying teams of developers to rally around best practice, the checklist may be good to review and reflect upon as an individual contributor to your codebase.

If you have any changes, suggestions or questions, please email us at : VADTCSFCOEArchAdmin@va.gov

Salesforce Change Control

The VA DTC team provides multiple levels of change control from an application or modules ideation to its end-of-life. These levels of change control exist to not only enforce that development efforts are approved and beneficial, but also to introduce quality gates to ensure the software solution is secure, performant, maintainable, and can scale. The DTC has multiple core teams that provide varying levels of change control.

Onboarding

New projects will be required to go through our intake process where funding is verified, data security is reviewed with the requisite artifacts being created (DSC, PIA, PTA). Before a project can move into development all of the steps must be completed and the project must be approved by VA Leadership.

Design and Development

Project teams should supply a “High Level Design (HLD)” as part of their intake process for review. Additionally, project teams must have their initial architecture technical designs reviewed and approved no later than 10% into their project 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 “Templates” section of the VA DTC Salesforce Architecture confluence page here: https://confluence.devops.va.gov/x/5oPjAg

Development must go through both security and code quality controls. Any development that requires external integrations or use of external packages must go through security review utilizing the DTC Security Risk Assessment process. Security reviews occur during Security Risk Assessment reviews every Wednesday at 10:30 am.

Integrations

For any new integration patterns for Salesforce, that integration design, in addition to architecture review, must go through the DTC Integration ARB (architecture review board) and get approval.   Integration architecture reviews occur during the Integration ARB meeting every Wednesday at 3pm.

Code Reviews

Code reviews will be performed both systematically and manually. After pull requests are submitted there will be an automated scan performed to enforce the coding standards defined above. Additionally, after a successful code scan, DTC Architects will perform a manual code review on the pull request which must be approved before Reg deployment. Any critical issues found must be resolved prior to deployment to production, other non-critical issues must be remediated per the feedback from DTC Architecture.

Post Development Sustainment

Before an application or module can move into a support model (a.k.a. Sustainment) with the DTC, project teams must have completed the following change controls:

- The application must have all exceptions identified and marked as “remediation required” fully remediated.
- The application must successfully pass an automated code scan without any critical defects found.
- All automated testing scripts or quality testing documents must be handed over to the Digital Assurance team for future QA testing.
- All 508-compliance audits must have been completed and any application specific issues found, fully remediated.
- Finally, a complete review of the application (including a demo) must be performed, and knowledge transfer performed.

508 Compliance Overview

Salesforce commercial off the shelf (COTS) defect is a bug and/or design flaw within the Salesforce platform that effects disability accessibility of a feature and/or functionality on the platform. COTS defects are generally not remediated immediately. Fixes typically are during one of the three Salesforce release cycles per year. Remediation of COTS defects are from Salesforce Support Team.

The DTC DA - Compliance Team assists with governance of 508 compliance of VA Salesforce modules and remediation of COTS defects by:

- Being a liaison between module teams, VA 508 Team, and Salesforce Help
- Managing and tracking the status of 508 COTS defects for modules
- Submitting COTS defects to Salesforce Help for triage
- Tracking and reporting statuses of COTS defects and when they are scheduled to be fixed by Salesforce
- Testing previously noted 508 COTS defects to verify if defect still exists.
- Collaborating with module team and VA 508 SME as needed.

References

- VA Directive 6500, VA Cybersecurity Program
- VA Handbook 6500, Risk Management Framework for VA Information Systems VA Information Security Program
- VA Knowledge Service (KS)
- NIST 800-53, revision 4, Security and Privacy Controls for Federal Systems

Rescissions

There are currently none.

Review

The SOP will be reviewed at least once per year for editing and will also be edited any time any policies or procedures are changed or updated.

Current SOP Owner

Name	Role	Email
Svatava Simpson	Salesforce COE	SVATAVA.SIMPSON@va.gov

Revision History

Date	Changed By	Notes
02 Oct 2023	Jen Wisniewski	Salesforce Team Review
26 Feb 2024	Jen Wisniewski	Updated *Test Run In User Scope* under Test Classes to include, Critical Impact, positive Reject Reason, with an effective date of 3/1/2024.
09 Jul 2024	David Schach	Add Security column to rules. Add rules enforced with CodeScan.