**Personalization SRE & DevOps**

* [The Team](#_Our_Organization!)
* What we do?
  + [DevOps Collaboration Model](#_DevOps_Collaboration_Model)
  + [Technical Debt](#_Technical_Debt)
    - [Technical Debt Tracker](#_Technical_Debt_Tracker)
      * [Automation Debt Tracker](#_Automation_Debt_Tracker)
      * [Splunk Dashboards Tracker](#_Splunk_Dashboards_Tracker)
    - [Automation Debt](#_Automation_Debt)
      * [Build Automation](#_Build_Automation)
    - [NFR Debt](#_NFR_Debt)
      * [Where NFR's are Handled?](#_Where_NFR's_are)
      * [**Reliability**](#_Reliability)
        + [Logging](#_Logging)
        + Usability / Learn-ability
        + Alert and Monitoring
        + Error / Exception Handling
        + Self-Healing
        + Resiliency
      * Sustainability / Maintainability
        + Accessibility
        + Code Quality and Deployment
        + Extensibility
        + Maintainability
        + Testability
      * Performance
        + Availability
        + Capacity and Scalability
        + Performance Testing
        + Redundancy / FailOver
      * Recovery
        + Back Up Plan
        + Data Retention
        + Disaster Recovery
      * Security
        + USAA Security Compliance
        + Security Scans
      * Compliance
        + Audit and Control
        + Compliance Standards
        + Documentation
      * Base-lining Applications

# [Welcome to Personalization SRE & DevOps Home Page!](#_top)

## Personalization SRE & DevOps emphasizes the collaboration and communication with all stakeholders in a product life cycle in an agile fashion while working towards achieving the state of continuous delivery pipeline while ensuring best in class customer experience through high availability of business capabilities

* Personalization SRE & DevOps BD Home

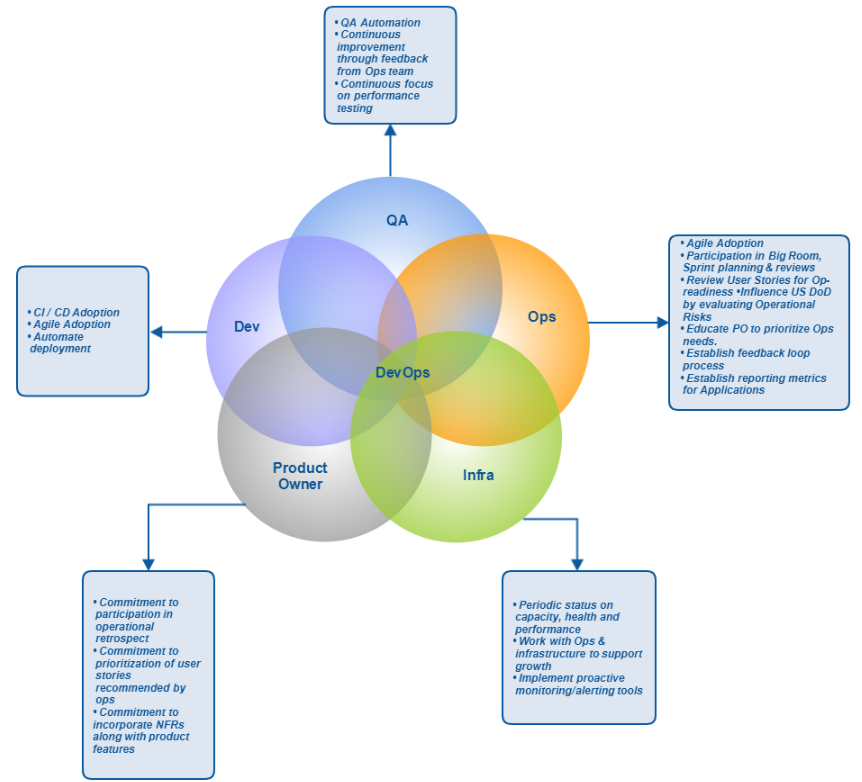
# Our Organization!

## ****We are Personalization SRE & DevOps Team****

Personalization SRE & DevOps emphasizes the collaboration and communication with all stakeholders in a product life cycle in an Agile fashion while working towards the '"Nirvana" state of continuous delivery pipeline and zero NFR & automation debt. Our primarily contribution includes enabling automation in all areas of development, testing, build and deployment, active collaboration with scrum masters and product owners in creating Non-Functional Requirements (NFRs) related user stories in to feedback loop & prioritizing to address technical and automation debt, ensuring the communication between Infrastructure, Development teams & Business partners on operational efficiency.

# DevOps Collaboration Model

DevOps Team - works in a close collaboration with the surrounding Eco-system, continuously updated with the developments and adapting to the changes around. Below pictorial presentation show cases some of the key interactions which goes into the smooth functioning of DevOps team



# Technical Debt

# Automation Debt

We as Dev-Ops team, have an important goal to reduce the manual activities and/or the typical boiler plate code which takes out a substantial amount of time from a developers schedule, everything from version control, build, testing to deployment - our goal is to make the developer focus on building business features - and we do this through***"Too Chain Automation"***of the CICD pipeline.

# NFR Debt

On other hand while the system is being built - we ensure that post deployment the system is maintainable. "Technical Debt" as per common industry definition - is a concept in software development that reflects the implied cost of additional rework caused by choosing an easy solution "now" instead of using a better approach that would take longer. Technical debt can be compared to monetary debt. If technical debt is not repaid, it can accumulate 'interest', making it harder to implement changes later on. Unaddressed technical debt increases software entropy.

Along with the core business user requirements, ***"NFR Debt"*** related user stories are also inserted into the product back-log. These user stories span across the below four categories, make sure the changes to the system comply with the standards and best practices and reduces the overall "NFR Debt"

1. Reliability
2. Sustainability / Maintainability
3. Recovery
4. Performance

# Technical Debt Tracker

## What is Technical Debt? How to Identify & Track Technical Debt?

Our team track Technical Debts to improve our development practices

Concretely communicating technical debt and its consequences is of interest to both researchers and software engineers.

Without validated tools and techniques to achieve this goal with repeatable results, developers resort to ad-hoc practices, most commonly using issue trackers or backlog-management practices to capture and track technical debt.

Studies revealed that technical debt has entered the vernacular of developers as they discuss development tasks through issue trackers.

Even when developers did not explicitly label issues as technical debt, it was possible to identify technical debt items in these issue trackers.

We use our results to motivate an improved definition of technical debt and an approach to explicitly report it in issue trackers. Here is how to describe the classification methods and some implications of tracking debt for both practice and research.

## Classification of Technical Debt

#### Executable or data related

To act on technical debt, it must be related to concrete development artifacts, such as code, implementation units, data models, build scripts, and unit tests. We classified any issue that did not mention a concrete development artifact as not technical debt. Classification from this point onward requires articulation of fuzzy concepts, such as defect, bug, and design concerns. Defects and bugs are incorrect functionalities visible to end users; technical debt tends to result from design and system issues not visible to the user. We separated defects from system improvement issues. Similarly, we separated new features, a type of system improvement, from cases where underlying design limitations resulted in feature requests.

#### Type > Defect type > Incorrect functionality

For example, a button doesn't work in the user interface or the system crashes. We classify these issues as not technical debt.

#### Type > Defect type > Design consideration

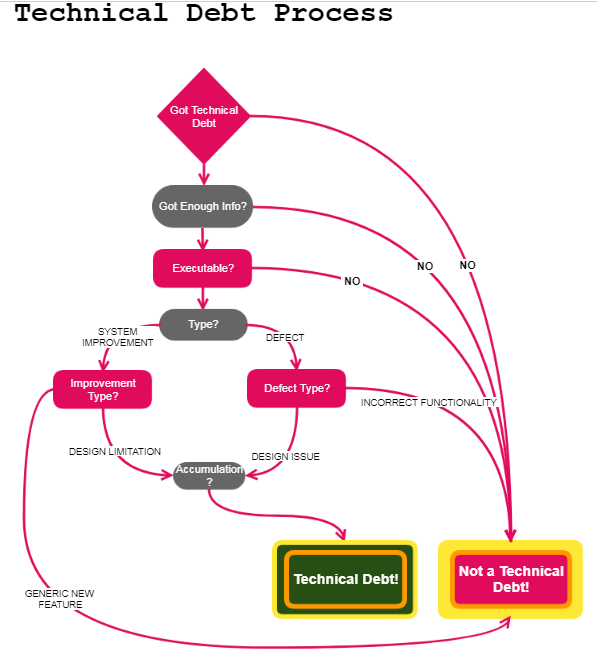
Several defects affected a quality attribute, such as availability, security, or performance; in other items, cleanup activities affected maintainability. We classify these issues as design considerations. If we also found evidence of accumulation of unintended side effects, or an estimate that they would accumulate; we classify these issues as technical debt. Examples include duplicate code, nonstandard binding, type mismatches, inconsistent implementation, and unused classes.

#### Type > Improvement type > Feature

We classify new features that were system improvements, such as adding a new node to a sensor component or removing a drop-down box, as not technical debt.

#### Type > Improvement type > Design limitation

Some issues described system improvements to remedy design limitations, such as the inability to add a new feature quickly, maintainability issues, or consequences of refactoring. To handle such cases, we introduced the design limitation category. When evidence of side effects was not clear, even for issues that clearly mentioned refactoring to remedy a design limitation, we classify the issue as not technical debt



# Automation Debt Tracker

DevOps has been contributing since 2016 on automating various aspects of an application Engineering life cycle as well as optimizing its NFR debt. In subsequent sections we will see the details of the various automations, technical optimizations and process optimizations achieved (or are WIP). Application teams can take a look at the details and also try to leverage the reusable components already built for their application needs

Below are the Use-Cases in which we have automated the deployment process

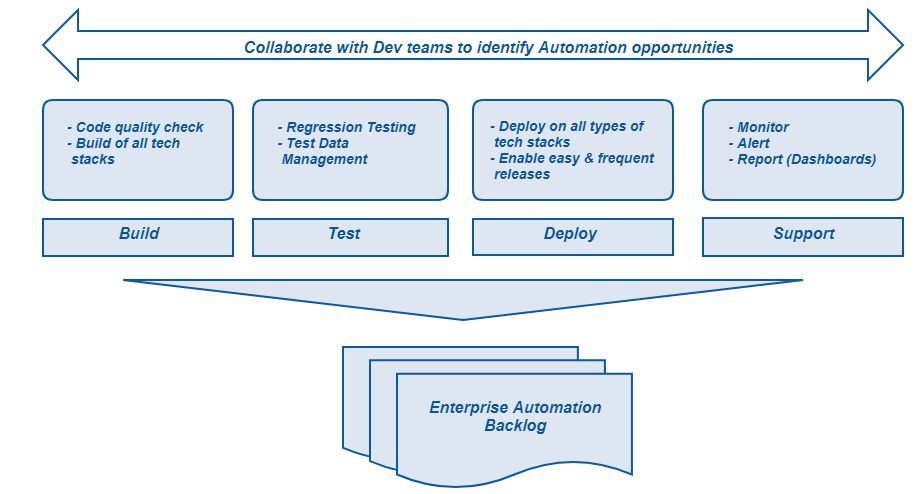
# Splunk Dashboards Tracker

Splunk Dashboards are prepared for various needs as outlined below

1. Application specific reporting
   1. Domain related KPI's being tracked as per the application needs - indicating the application is behaving functionally as intended to
   2. Vital operational stats of the application being monitored, like API count, Batch job execution tracking etc..
2. Infra Specific reporting - tracks the underlying infrastructure and possible Alerts
   1. CS3 availability
   2. Other Infrastructure related reports

# Automation Debt

DevOps Team works across the Application Maintenance Life Cycle, identifying opportunities to automate the manual tasks across the development, and even post deployment into Support



# Build Automation

Jenkins One click Build and Deployment will help in automated deployment your code to cluster and Rolling back to earlier version deployed in case of issues in current deployment.

**Steps for Build Automation**:

1. Get neccessary access -> Refer  Requirements for Build-Automation
2. Raise JIRA Ticket to onboard Project in Gold<**Optional** in case not already onboarded in BDEC Jenkins>
3. Download our build deploy code in local machine :  
   GIT :    
   Branch: Master or Deploy
4. SVN to GIT Repository Migration [**Optiona**l ,only if project is using SVN repository]   -> Refer SVN To GIT Migration and <GIT steps>
5. Create Bundle.xml & Pom.xml for Projects containing only shell scripts or is a multimodule Project [optional, no need to do if project is already having its Pom.xml] : Refer Bundle.xml & Pom.xml and Build-Deployment
6. Build of usecase Code at local machine
7. Create folder for DEPLOY and ROLLBACK in BDEC Jenkins :
8. Build Job in BDEC Jenkins for all environment types :Gold-Dev,Silver,Prod-test,Gold-Snapshot BDEC Evironments
9. Create a new Job or copy and existing job by putting "/ " and update the configuration details
10. Create Configuration files Properties file in config folder GIT in local machine and push the code in remote repository  
     1)create global and then in gold-dev/relase/prod test ->folders (copy already existing and edit parameters accordingly)
11. Raise JIRA Ticket to onboard Project in Platinum with Prod-Test Job <as the Gold and Production BDEC Jenkins Links are different>:Refer Raising JIRA Ticket for BDEC Jenkins

**Supported Deployment** :One click Automation supports artifact deployment in Production/Gold  as below:

1. Batch Job
2. API-Tomcat
3. UI

**Software Needed :**

1. Tortoise SVN
2. GIT  [ It will download GIT cmd , GIT bash, GIT GUI ]

# NFR Debt

Nonfunctional Requirements (NFR's or system qualities) are just as critical as functional Epics, Capabilities, Features, and User Stories and ensure the usability and efficacy of the entire system. They are persistent qualities and constraints that, unlike functional requirements, are typically revisited as part of the Definition of Done (DoD) for each Iteration & Program Increment. Failing to meet any one of them can result in systems that fail to satisfy internal business, user, or market needs or that do not fulfill mandatory requirements imposed by regulatory or standards agencies.

In upcoming subsections, we would be referring to the Reliability, Sustainability and Performance. Below table depicts a quick summary of the 2 levels of the "NFR" - talks about ***"What"*** these NFR's are. We also have "Tools" outlined which talks about **"How"**to meet these NFR's. In short duration the team will also be outlining the key ***"metrics"*** which will be used to measure the progress of each application on the NFR's. Again it is important to note that not all NFR's are measurable, for example "testability" is an extrinsic property and is rather intangible but its effects can definitely be perceived.

Where NFRs are handled?

| **Category** | **Definition** | **Standard NFR’s** | **Tools Used To Measure/ Validate** |
| --- | --- | --- | --- |
| Reliability | Ability of system or application to be resilient and selfheal including ability to monitor and support   | **Run-time Qualities** | **Concept** | **Where Applied (5 W's)** | | --- | --- | --- | | Alert Management | Requirements on the infrastructure and/or application to raise alerts in response to error or exception conditions | Use of discrete and consistent Error Codes | | Service Level Reporting | Requirements for the capture of data for any reduction of reports on system performance against agreed service levels |  | | System Monitoring and Measurement | The expected monitoring and measurement approaches to be taken for the live service infrastructure and applications |  | | Maintenance and Administration | Adding and removing users and their privileges, monitoring the application and software distribution and installation |  | | Event logs | A record of events taking place in the execution of a system in order to provide an audit trail | Ability to dynamically change the logging level (i.e. Log4j) | | Transaction logs | Record changes to the stored data to allow the database to recover from crashes or other data errors and maintain the stored data in a consistent state |  | | Logging  Alert and Monitoring  Error/Exception Handling  Self-Healing  Resiliency | * Tivoli/Nimsoft/Dynatrace/Gomez * Elk / Splunk * Custom code |
| Sustainability | Maintainability of a system or application including code quality, extensibility, testability, usability, and ability to be repeatedly and easily deployed   | **Run-time Qualities** | **Concept** | **Where Applied (5 W's)** | | --- | --- | --- | | Flexibility | The degree to which the system (code, parameters, components, modules etc.…) is intended to support change |  | | Portability | The extent to which the system supports platform and product independence, so that it could be ported to alternative platforms |  | | Release Support | The ways in which the system will support the introduction of the initial release, phased rollouts, and future releases | Blue/Green and Canary deployments | | Code Quality  Maintainability  Extensibility  Usability  Testability  Accessibility  Deployment | * PMD / Findbugs / Checkstyle / CAST * Review Board  / Crucible * JUnit / NUnit / Cobertura / NCover * HP QC / Rally * Jenkins / LARA / LISA / XL Release |
| Performance | Ability of a system or application to perform and scale under load conditions   | **Run-time Qualities** | **Concept** | **Where Applied (5 W's)** | | --- | --- | --- | | Throughput requirements | The ability of the system to execute a number of transactions within a given unit of time | Established by business requirements Designed/built to meet business requirements Validated/Documented via Performance, Endurance (hrs/days) and Stress (2-3x) testing Capture how many, how fast and success/failure rate | | Response times | The allowable distribution of time which the system takes to respond to requests | Established by business requirements Designed/built to meet business requirements Validated/Documented via Performance, Endurance (hrs/days) and Stress (2-3x) testing | | Additional "functional" tests | The allowable distribution of time which the system takes to respond to requests | Negative testing Destructive testing Security testing | | Capacity requirements | Ability of the business system to ensure that the system continues to meet or exceed its service level agreements | Established by business requirements Designed/built to meet business requirements Validated/Documented via Performance, Endurance (hrs/days) and Stress (2-3x) testing | | Scalability | Ability to continue to function well as it changes in size of volume in order to meet used need | Established by business requirements Designed/built to meet business requirements Validated/Documented via Performance, Endurance (hrs/days) and Stress (2-3x) testing | | Availability | The minimum proportion of time of given online service hours that the service should be available |  | | Application Restarts | The minimum occurrence of manual or automated application restarts (reboot of container, application pool recycle, etc) allowed in a time period. |  | | Performance  Scalability  Redundancy/Failover  Availability  Capacity Plan | * CAST * HP Load Runner * JMeter / BlazeMeter |
| Recovery | Ability of a system or application to recover when systems fail or under disaster conditions.   | **Run-time Qualities** | **Concept** | **Where Applied (5W's)** | | --- | --- | --- | | Offline requirements | The times available, if any, for non-online operations including batch processing and system maintenance activities |  | | Resilience | The reliability and resilience characteristics of the systems and/or its subcomponents | Error/Exception Handling BIA/RTR requirements must be met Consider Active/Passive GDHA | | Recoverability | Ability to handle and recover from unexpected failures and outages of components both with and within and outside the system | Self-Healing | | Disaster Recovery | Ability to recover from and continue service following a major incident such as the loss of an entire data center |  | | DR  Backup Plan  Data Retention Plan | * GDHA / GTM / DR Solution Design * Avamar * USAA Audit and Compliance |
| Security | Ability of a system or application to secure data / systems under threats | Security Compliance  Security Scans | * USAA Security Compliance |
| Compliance | Ability of a system or application to comply to USAA Standards | Audit and Control  Standards Compliance  Documentation | * USAA Audit and Compliance * USAA Technology Road Map * Rally / JIRA / JavaDoc / NDoc / SAD and SID |

# Where NFR's are handled?

Below chart showcases where in the application life cycle NFR have to be taken care of. It will be a guiding chart for the development team and more specifically the architects as well to keep in mind, in order to review and guide the development process

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Legend*** | | ***Design For*** | | ***Implement For*** | | | ***Test For*** | | ***Leverage In*** | | | ***Does Not Impact*** | | | |
| ***D*** | | ***I*** | | | ***T*** | | ***L*** | | | ***N/A*** | | | |
| Category | NFR | | Envision | | Build/Execute | Unit Testing | SIT Testing | Regression Testing | | User Acceptance Testing | Production Readiness Testing | | Performance Testing | Deploy | LiveOps | |
|  | | | | | | | | | | | | | | | | |
| Reliability | Logging | | D | | I | T | N/A | N/A | | N/A | N/A | | N/A | N/A | L | |
| Alert and Monitoring | | D | | I | T | T | Sometimes | | N/A | T | | N/A | N/A | L | |
| Error/Exception Handling | | D | | I | T | T | Sometimes | | N/A | T | | N/A | N/A | L | |
| Self Healing | | D | | N/A | N/A | T | Sometimes | | N/A | T | | N/A | N/A | N/A | |
| Resiliency | | D | | N/A | N/A | T | Sometimes | | N/A | T | | N/A | N/A | N/A | |
|  | | | | | | | | | | | | | | | | |
| Sustainability | Code Quality | | N/A | | I | N/A | N/A | N/A | | N/A | N/A | | N/A | N/A | L | |
| Maintainability | | D | | I | N/A | N/A | N/A | | N/A | N/A | | N/A | N/A | L | |
| Extensibility | | D | | I | N/A | N/A | N/A | | N/A | N/A | | N/A | N/A | N/A | |
| Usability | | D | | I | N/A | T | Sometimes | | T | T | | N/A | N/A | N/A | |
| Testability | | D | | I | N/A | T | Sometimes | | N/A | N/A | | N/A | N/A | N/A | |
| Accessibility | | D | | I | N/A | T | Sometimes | | T | T | | N/A | N/A | N/A | |
| Deployment | | D | | I | N/A | L | L | | L | L | | L | L | N/A | |
|  | | | | | | | | | | | | | | | | |
| Performance | Performance | | D | | I | T | N/A | Sometimes | | N/A | T | | T | N/A | N/A | |
| Scalability | | D | | I | N/A | N/A | Sometimes | | N/A | T | | T | N/A | N/A | |
| Redundancy/Failover | | D | | N/A | N/A | T | Sometimes | | N/A | T | | N/A | N/A | N/A | |
| Availability | | D | | N/A | N/A | T | Sometimes | | N/A | T | | N/A | N/A | N/A | |
| Capacity Plan | | D | | N/A | N/A | T | N/A | | N/A | N/A | | N/A | N/A | N/A | |
|  | | | | | | | | | | | | | | | | |
| Recovery | DR | | D | | N/A | N/A | T | Periodically | | N/A | N/A | | N/A | N/A | N/A | |
| Backup Plan | | D | | N/A | N/A | T | N/A | | N/A | N/A | | N/A | N/A | N/A | |
| Data Retention Plan | | D | | N/A | N/A | T | N/A | | N/A | N/A | | N/A | N/A | N/A | |

# Reliability

It is the ability of system or application to be resilient and to self heal which includes ability to monitor and support. It covers the following aspects which are discussed in detail in subsequent sub sections below where we would like to discuss them detail and how they help us work better on operations support:

Logging as a Service - Basic has been rolled out to all application teams and DevOps team is training the consistency across all engineering teams to establish a process in place.

| **Sr No** | **Type** | **Description** | **Tools Used To Measure / Validate** |
| --- | --- | --- | --- |
| 1 | **Logging**  **Log Format**  **Error Logging**  **Log Retention**  **Log Purging**  **Log Size**  **Other Log Best Practices**  **Logging Data Maturity**  **Logging as a Service** | This section contains recommendations for best practices of logging which help us better alert and monitoring | Tivoli  Nimsoft  APP Dynamics |
| 2 | **Alerts and monitoring**  **Batch Job**  **Infrastructure Alerts**  **Real Time**  **Splunk Dashboards** | This section covers the alerts and monitoring check | Elk  Splunk |
| 3 | **Usability** | It is the ease of use and learn ability of a human-made object such as a tool or device |  |
| 4 | **Error / Exception Handling** | This section covers recommendations for   * exception handling is the process of responding to the occurrence, during computation, of exceptions * anomalous or exceptional conditions requiring special processing * often changing the normal flow of program execution | Custom Code |
| 5 | **Self-Healing** | This section covers details about how quickly system recovers after failure |  |
| 6 | **Resiliency** | This section covers the capacity of a system to continually change and adapt yet remain within critical thresholds |  |

# Logging

In this section we would be aiming to achieve log better logging so we can monitor the use case. Logging is the act of keeping a log of activities**.**A log file is a record that keeps track of events that occur in an operating system or software runs or messages of interfacing applications.

## ****Logging available in various Programming Languages****

Each language we use, there is a way to provide logging and error handling. Currently we have recommendations forJava, UNIX, Python, Big data Tools. Below is table where we have Programming Languages and their recommended Logging Tool/API which would help in making logging more readable and easy to use for debugging.

| **Programming Language** | **Tools Recommended** | **Brief Description** | **Related Links (External)** |
| --- | --- | --- | --- |
| Java | Sl4j  Log4j | The Simple Logging Facade for Java (SLF4J) serves as a simple facade or abstraction for various logging frameworks (e.g. java.util.logging, logback, log4j) allowing the end user to plug in the desired logging framework at deployment time.  Log4J is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License. | [SL4j Guide](https://dzone.com/articles/how-configure-slf4j-different)  [Log4j setup](http://www.codejava.net/coding/how-to-configure-log4j-as-logging-mechanism-in-java) |
| Unix | Log4sh | [log4sh](http://sourceforge.net/projects/log4sh)  runs along the same lines as the other excellent  [logging services](http://logging.apache.org/)  from the  [Apache Software Foundation](http://www.apache.org/) . It adds to that list the ability to integrate powerful logging capabilities into a shell script. | [Log4sh Guide](https://sites.google.com/a/forestent.com/projects/log4sh) |
| Python | - | Python has API to for Logging, please refer the helpful link for more details | [Python Logging](https://docs.python.org/2/howto/logging.html) |
| Scala | Sl4j Wrapper | The Simple Logging Facade for Java or (SLF4J) serves as a simple facade or abstraction for various logging frameworks, e.g. java.util.logging, log4j and logback, allowing the end user to plug in the desired logging framework at deployment time. | [Scala Logging](https://github.com/typesafehub/scala-logging) |
| Apache Pig | Log4j | Apache Pig is an ETL tool where we can use Log 4j for logging.  If you want to debug whole script during execution then you need to write below code at top of your script  -- set the debug mode on SET debug 'on' -- set a job name of your job. SET job.name 'my job'  Another way  Step1: copy the log4j config file to the folder where my pig scripts are located.  cp /etc/pig/conf.dist/log4j.properties log4j\_WARN  Step2: Edit log4j\_WARN file and make sure these two lines are present  log4j.logger.org.apache.pig=WARN, A  log4j.logger.org.apache.hadoop = WARN, A  Step3: Run pig script and instruct it to use the custom log4j  pig -x local -4 log4j\_WARN MyScript.pig | [Pig Latin Log4j setup](https://stackoverflow.com/questions/16627748/pig-batch-mode-how-to-set-logging-level-to-hide-info-log-messages) |
| Apache Hive | Log 4j | We can setup logging in Hive 2 ways:  hive --hiveconf hive.root.logger=DEBUG  I ended up creating a hive-log4j.properties file in my home directory with following settings:    log4j.rootCategory=DEBUG,console log4j.appender.console=org.apache.log4j.ConsoleAppender log4j.appender.console.target=System.err log4j.appender.console.layout=org.apache.log4j.PatternLayout log4j.appender.console.layout.ConversionPattern=%d{yy/MM/dd HH:mm:ss} %p %c{1}: %m%n    and started hive shell using CLASSPATH=$HOME hive which adds your home directory having hive-log4j.properties in front of the classpath and so is picked up. | [Hive Logging](https://stackoverflow.com/questions/27854359/how-to-run-hive-in-debug-mode)  [Hive Debugging](https://cwiki.apache.org/confluence/display/Hive/DeveloperGuide#DeveloperGuide-DebuggingHiveCode) |

# Logging Data Maturity

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Maturity** | **Novice** | **Practitioner** | **Expert** |
| * Uses logging to understand basic operational visibility for applications * Instruments code to employ Basic Application Log Levels: Info/Debug/error   + Logs to a central log repository   + Uses Plain-text Format         **Shift to next level of maturity: TBD**   * Logs indicate business transactions | * Application Logs identify:   + Who?   + What?   + How many?   + How fast?      * Uses well-defined structure   + Key-value   + JSON   + XML      * Applies consistency in:   + Application naming conventions   + Component naming conventions   + Proper Time-Stamp formatting      * Uses Proxy account identifiers  (no PII)     **Shift to next level of maturity**:   * Fully utilizing Enterprise Logging Framework * Utilizing log level data for capacity planning * Having logs available to view by other teams | * Fully complies with the Enterprise Logging Standard * Fully utilizes the Enterprise Logging Framework * Employs Machine Learning for Predictive analytics * Full framework integration (feature toggle) * Selective logging on the fly * Encrypted fields     **Shift to next level of maturity:**  Democratize the data |

# Log Format

This section has details about the recommended different log formats and benefits from using them, currently recommendation are for JAVA only, further enhancements with other languages will be coming soon

### ****Real Time API Log Format****

#### **1** - **Recommended pattern of Log for Splunk dashboard**

Splunk dashboard is a monitoring and alert/incident generation tool which helps in monitoring the application. Below log pattern creates a readable log  is recommended which Splunk can consume. It ensures all mandatory information required for debugging an issue are present in logs.

Date HH:MM:SS,SSS [<Unique Transaction ID Coming from Request>] SplunkLogUtility : [Api\_Type:Offer\_Recommendations\_API | StartTime:1480921819319 | EndTime:1480921819964 | TotalTime:<value> |ExternalService1:<Success/Fail> <Time\_Taken by it to process> | ExternalService2:<Success/Fail> <Time\_Taken by it to process>

***2*** -***Unique Transaction Id***

Log statement should include Unique Transaction Id in Logs. Easiest way to achieve this is by using set name of current thread as unique transaction ID using purpose is While debugging we want achieve that helps in debugging easier and better->to find  brief description.purpose ,

**Method with below mentioned Log4j pattern.**

**Thread.currentThread().setName(<>)**

Our team recommended **Log4j pattern.**

“[%t] %d{yy/MM/dd HH:mm:ss,sss} %-5p %c{1}:%L - %m%n”

#### **3**- **External service call**

Log should capture something which can be used to identify request uniquely (Input Request ID passed to call service if any), Response code, Response Message received from Service

***4*** - ***Unique Error Code***

For any external Service call / database call failure: log should print unique Error Code like. <EXTERNAL\_SERVICE>\_DAO\_ERR, <HBASE/SOLR>\_DAO\_ERR.

* **Exception Type Printout** :Log statement should include the type of exception like TimeOutException etc in same statement
* **Redundant error Logs removal**: Error code should be printed only once per error. For e.g. it should not get printed again in log statement printed for Splunk OR again in exception trace.
* **For data base call** : Use different Error code which cause impact to caller AND no impacted to caller
* If it is timeout OR connection issue , log it separately

***5 - Different Response Code***

For different scenario should be logged. For e.g.

* Different response code for different missing field or unexpected value in request <**depend upon app to app**>
* Different response code for success with data OR success without data **<depend upon app to app , if not then result return in logs >**

***6 - Batch Jobs Logs format***

* Include log statement to indicate start and end of Job, each data extraction from different system, internal processing steps as well data transfer to external team
* Include log statement to indicate failure of process, data extraction , internal processing steps & data transferred to external team
* To keep separate log files for each job as they would help debugging the issue better

# Log Retention

In this section we are going to describe different languages and their ways to retain logs so they can be accessed for longer periods. Below are few recommendations

1. **Tomcat Log Retention** : Logs to be retained for 10 days in tomcat and then copied up to one year in platinum edge node as Edge node[space allocated is around 20TB per use-case] has 7\* times more space than Tomcat servers[300 GB]  depending upon use-case we can decide the max retention period for logs.
2. **Logging Level** :Use Warn, Exception and Error level of logging to reduce log size

***Reasons to compress logs for retention***

* Data is mostly stored and not frequently processed. It is usual Data warehouse scenario. In this case space saving can be much more significant than processing overhead
* Compression factor is very high and thereof we save a lot of IO.

Below are different Programming Language/Tools and recommended ways to compress the log files [At end of page you can find details about different compression techniques]

| **Programming Language/Tool** | **Recommendations/**  **Compression Technique** | **Summary and Benefits** | **Description** | **Useful Link (External)** |
| --- | --- | --- | --- | --- |
| Apache Hive | Snappy Codec [preferably with parquet file format] | Snappy is one of the fast compression/decompression tools. It formerly known as Zippy.  Benefits:  1.Decompression is very fast and thereof we have a some gain with little price | Create a Snappy File  hive> SET hive.exec.compress.output=true;  hive> SET mapred.output.compression.codec=[org.apache.hadoop.io](http://org.apache.hadoop.io/).compress.SnappyCodec; hive> SET mapred.output.compression.type=BLOCK;  hive> create table hivetest2 (a int, b int);  OK  Time taken: 0.15 seconds  hive> select \* from hivetest2 limit 1;  OK  1       1  Time taken: 0.171 seconds  $ hadoop dfs -ls /user/hive/warehouse/hivetest2  Found 1 items  -rw-r--r--   3 hdfs supergroup       4021 2012-01-19 10:20 /user/hive/warehouse/hivetest2/000000\_0.snappy  Read a Snappy File  hadoop fs -cat /user/hive/warehouse/hivetest2/000000\_0.snappy | head -5  .¸11  22  33 | [Snappy Codec for Hive](https://mapredit.blogspot.in/2012/01/use-snappy-codec-with-hive.html) |
| Unix | Gunzip |  | Gunzip filename |  |
| Java | - | Through code we can achieve compression | StringBuilder sb = new StringBuilder();   sb.append("Test String");   File f = new File("<current directory>\test.zip");   ZipOutputStream out = new ZipOutputStream(new FileOutputStream(f));   ZipEntry e = new ZipEntry("mytext.txt");   out.putNextEntry(e);   byte[] data = sb.toString().getBytes();   out.write(data, 0, data.length);   out.closeEntry();   out.close();  This will create a zip file in the current directory named 'test.zip' which will contain single file called 'mytext.txt'.  You can add more entries to zip and could specify a sub directory like:   ZipEntry e = new ZipEntry("folderName/mytext.txt"); |  |

**Different compression techniques:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Compression format | Tool | Algorithm | File extension | Splittable | Compression Ratio | Description |
| gzip | gzip | DEFLATE | .gz | No | 22% | gzip is naturally supported by Hadoop. gzip is based on the DEFLATE algorithm, which is a combination of LZ77 and Huffman Coding |
| Snappy | N/A | Snappy | .snappy | No | 40% | Snappy is a compression/decompression library. It does not aim for maximum compression, or compatibility with any other compression library; instead, it aims for very high speeds and reasonable compression. For instance, compared to the fastest mode of zlib, Snappy is an order of magnitude faster for most inputs, but the resulting compressed files are anywhere from 20% to 100% bigger. On a single core of a Core i7 processor in 64-bit mode, Snappy compresses at about 250 MB/sec or more and decompresses at about 500 MB/sec or more. Snappy is widely used inside Google, in everything from BigTable and MapReduce to our internal RPC systems |

# [Log Purging](https://enterprise-confluence.aexp.com/confluence/display/GMD/Log+Purging)

In this section we would be sharing working demo script for purging log.

**Tomcat Log Purging script**: It will delete data older than 30 days on specified tomcat server.

#!/bin/bash

ServerList="server1,server2 "

echo "ServerList is ::$ServerList"

for server in ${ServerList//,/ }  
do

echo "Running for Server server::$server"

ssh <usecaselogin>p@$server <<EOF

find /opt/lucidworks/data/logs/\* -mtime +30 -delete  
exit 0  
EOF

echo "Done for Server server::$server"

done

echo "Done for All Server"

# Log Size

In this section we would be covering different aspects to maintain log size and areas of concern with growing log size which we have seen with our experience.

### ****Areas of Concern****

1. **Splunk challenges with bigger log size:** Splunk, as it takes more time to fetch records and display result on dashboard and maintain at backend 1gb logs per day for 3 months total duration]
2. **System resource consumption increase with increase in space**
3. Overtime when it grows, CPU **process time increases**.
4. **Use rotation policies**: Logs can take up a lot of space. Maybe compliance regulations require you to keep years of archival storage, but you don't want to fill up your file system on your production machines. So, set up good rotation strategies and decide whether to destroy or back up your logs.
5. **Reduction in log size can be achieved**  
   - Removal of redundant lines printed :  for eg: below 2 lines printed are common  
     - Hbase indexing completed   
     - Indexing completed for Hbase in 5 sec.  
   - Using compression techniques to reduce size of files being stored

**Recommended Log Recording Level for Log4j**

Information, Warning and Error are recommended Log levels as they are self-explanatory as small in size. Debug and Fatal are mostly big in size. Below is a table describing different log Levels and their respective description.

| **Log Level** | **Description** |
| --- | --- |
| Fatal | Fatal is reserved for special exceptions/conditions where it is imperative that you can quickly pick out these events. I normally wouldn't expect Fatal to be used early in an application's development. It's usually only with experience I can identify situations worthy of the FATAL moniker experience do specific events become worth of promotion to Fatal. After all, an error's an error. |
| Error | Error is used to log all unhandled exceptions. This is typically logged inside a catch block at the boundary of your application |
| Warning | Warning is often used for handled 'exceptions' or other important log events. For example, if your application requires a configuration setting but has a default in case the setting is missing, then the Warning level should be used to log the missing configuration setting. |
| Information | The Information level is typically used to output information that is useful to the running and management of your system. Information would also be the level used to log Entry and Exit points in key areas of your application. However, you may choose to add more entry and exit points at Debug level for more granularity during development and testing. |
| Debug | This is the most verbose logging level (maximum volume setting). I usually consider Debug to be out-of-bounds for a production system and used it only for development and testing. I prefer to aim to get my logging levels just right so I have just enough information and endeavor to log this at the Information level or above. |

# Other Log Best Practices

While the above section covers exhaustively all the aspects to effectively log in an Enterprise standard way, teams can chose to follow the below best practices

1. **Walk the code**: This will help understand what inputs were passed in request or state of job before failure.
2. **Aggregation**: All Log & Exception data to one place
3. [**Structured format**](https://stackify.com/what-is-structured-logging-and-why-developers-need-it/)**:**Would help in making logs highly indexed and searchable.

# [Usability / Learn-ability](https://enterprise-confluence.aexp.com/confluence/pages/viewpage.action?pageId=145584645)

|  |  |
| --- | --- |
| NFR: | Usability |
| Definition: | * Usability is the ease of use and learnability of a human-made object * The object of use can be a software application, website, book, tool, machine, process, or anything a human interacts with! * Usability includes methods of measuring usability, such as needs analysis and the study of the principles behind an object's perceived efficiency or elegance * In human-computer interaction and computer science, usability studies the elegance and clarity with which the interaction with a computer program or a web site (web usability) is designed * Usability can be driven by look and feel standards - screen element density, layout and flow, colors, UI metaphors, keyboard shortcuts, and internationalization / localization requirements – languages, spellings, keyboards, paper sizes, etc |
| Guidance: | 1. Usability generally includes five characteristics, which must be met for the users of a product: effective, efficient, engaging, error tolerant, and easy to learn 2. Effectiveness is the completeness and accuracy with which users achieve specified goals It is determined by looking at whether the user’s goals were met successfully and whether all work is correct 3. Efficiency can be described as the speed (with accuracy) in which users can complete the tasks for which they use the product ISO 9241 defines efficiency as the total resources expended in a task Efficiency metrics include the number of clicks or keystrokes required or the total ‘time on task’ 4. An interface is engaging if it is pleasant and satisfying to use The visual design is the most obvious element of this characteristic 5. The ultimate goal is a system which has no errors An error tolerant program is designed to prevent errors caused by the user’s interaction, and to help the user in recovering from any errors that do occur Some guidelines for preventing errors are:   \* Make it difficult to take incorrect actions Design links and buttons to be distinctive, use clear language, avoiding technical jargon, and be sure that dependent fields or choices appear together \* Make it difficult to take invalid actions Limit choices when possible to those which are correct, provide clear examples for data entry, present only appropriate navigation options \* Make it difficult to take irreversible actions Provide the ability to back track, provide means to undo or reverse actions, avoid dead-end screens Don’t indiscriminately use confirmations – users become insensitive to them \* Plan for the unexpected Allow for users to add new entries, take exceptional routes through the interface or make choices you did not predict Be polite about "correcting" mistakes that may arise from this lack of foresight |
| Examples: |  |
| Attachments: |  |
| Associated Standards: | Refer to the Guide on the practices |
| Version: | 1 |
| Acceptance Criteria: | Development teams thru architecture review, peer code reviews, and customer beta releases will review their applications to ensure they meet customer usability acceptance criteria |

# Alert and Monitoring

In this section we would touch base monitoring the state of an application and raising alerts in case of issue. **Monitoring** is observing and check the progress or quality of (something) over a period of time, keep under systematic review and to raise alert in case of any issue An **Alert** informs us of conditions which otherwise we would not been have aware of otherwise.

Below is table with various programming languages and Tools recommendations for same.

| **Programming Language** | **Tools Recommended** | **Brief Description** | **Helpful Link (External)** |
| --- | --- | --- | --- |
| Java | Splunk[Recommended]  Loggly    Storm | **Splunk** provides the platform for Operational Intelligence where it helps to search, monitor, analyze and visualize machine data  **Loggly** is also a robust log analyzer, focusing on simplicity and ease of use  **Apache Storm** is a free and open source distributed realtime computation system. **Storm** makes it easy to reliably process unbounded streams of data, doing for realtime processing what Hadoop did for batch processing. **Storm** is simple, can be used with any programming language, and is a lot of fun to use | [Java:Log Management Tools](https://dzone.com/articles/7-log-management-tools-java) |
| Unix | Nagios      LogWatch  GrayLog | [Nagios](http://list.xmodulo.com/nagios.html): is an enterprise-class network and infrastructure monitoring system which comes with extensible monitoring and [alerting](http://xmodulo.com/configure-nagios-audio-alerts-mobile-notifications.html) capabilities. Nagios plugins (example [here](http://labs.consol.de/lang/en/nagios/check_logfiles/)) can turn Nagios into a centralized log monitoring server, where you can view the status of custom log checks and get notified of any threshold breaches.  [logwatch](http://xmodulo.com/monitor-log-file-linux-logwatch.html): an open-source log parser and analyzer which can interpret a wide range of common service and application logs and generate customizable HTML reports ready for email delivery.  [Graylog](http://list.xmodulo.com/graylog.html): a fully-integrated log management platform which is capable of collecting, indexing, storing and analyzing virtually any kind of data (both structured and unstructured) from remote servers. Written in Java, Graylog is easy to set up, and requires little maintenance. Optionally, its input interface can be integrated with different log collectors such as rsyslog, syslog-ng and [logstash](http://jpmens.net/2012/08/06/my-logstash-and-graylog2-notes/). It also features a great and easy-to-understand web-based dashboard with pre-defined views for quick access. | [Unix:Log Management Tools](https://www.tecmint.com/best-linux-log-monitoring-and-management-tools/) |
| Python | Splunk | **Splunk** provides the platform for Operational Intelligence where it helps to search, monitor, analyze and visualize machine data |  |
| Scala |
| Bigdata Tools[Pig] |

# Batch Job

In this section we would be describing monitoring and alerts related to Batch Process.

Below table covers various aspects which would be monitored and alert would be generated in case of issue

| **Name** | **Description** | **API/Tool Recommendation** | **Resolution** |
| --- | --- | --- | --- |
| Failure Incidents | Job Failure alerts are there through incidents through API/Splunk [Mandatory]  Alert through mails are discouraged as they might be missed | Ticketor Robo API/Splunk | As per log RCA needs to be carried. |
| Report | Reports to be generated for monitoring purpose or for purpose of helping debug other issues | Splunk | NA |
| Success Mail | Success Mails are discouraged as they might end up populating mailbox. | NA | NA |

# Infrastructure Alerts

In this section we would be describing monitoring and alerts related to Infrastructure for different applications.

Below table covers various aspects which would be monitored and alert would be generated in case of issue.

| **Name** | **Description** | **Tool Recommendation** | **Resolution** |
| --- | --- | --- | --- |
| Platinum Edge Node overload status | Platinum edge Node CPU utilization gets overloaded to process[s] which are consuming memory more than allocated to their queue | Nagios | DevOps and Dev Team need to check overloading process id and take action accordingly |
| Tomcat status | There are 2 status for Tomcat: Down[not running] and Recovery[recovered from issue] | Nagios/Splunk | If within 20-30 mins after Down alert if we do not receive a recovery alert ,we might need to immediately restart it |
| Out Of Memory Error | We recommend in logs to have out of memory error to printed so using Splunk we would be able to monitor it | Splunk | Once alert is generated we need to see restart after taking necessary backup of heap logs and find the cause of it. |
| CPU utilization status | We monitor the CPU utilization limit and raise alert when they breach threshold counts[e.g - +70% of total usage we raise alert ] | Splunk | To check over CPU using process and their cause. |
| Thread counts status | We monitor the thread counts and raise alert when they breach threshold counts [e.g - +200 thread counts we raise alert ] | Splunk | To check thread dumps for debugging. |
| Disk space usage status | We monitor the space and raise alert in case of space utilization is less than 20% | Splunk | To clear very old large files [e.g. logs and unwanted dump] |

# [Real Time](https://enterprise-confluence.aexp.com/confluence/display/GMD/Real+Time)

In this section we would be describing monitoring and alerts related to Real Time API.

Below table covers various aspects which would be monitored and alert would be generated in case of issue.

| **Name** | **Description** | **Tool Recommendation** | **Resolution** |
| --- | --- | --- | --- |
| Total Transaction Counts | Here we monitor the total transaction counts which get updated with real time recording. | Splunk | In case of more than 5% timeouts, alert/Incident is raised and needs to be debugged for RCA. |
| Timeouts per Application and DB tier applications | Here we monitor the timeouts which get updated with real time recording with error messages | Splunk | in case of threshold breach which is defined by Dev Team alert/Incident is raised and needs to be debugged for RCA. |
| Transaction per Host Monitoring | We monitor the trend of transactions occurring over different hosts | Splunk | In case of uneven distribution Nagios setup needs to be verified. |
| Reports | As per requirement of different team reports are generated on a set of frequency for purpose of monitoring | Splunk | NA |

# Splunk Dashboard.

Splunk dashboard helps in monitoring the use case from different perspective and raising preconfigured alert [Incident recommended] in case of issue. Splunk has been integrated with Service Now to directly raise incidents in respective queue as mails might get ignored.

**Below are 3 types of Dashboard**

#### **Real Time Dashboard**

This Dashboard helps monitoring various aspects of API in real time like offers/types of API/ using logs and shows timeouts for different. Please refer link Real Time for more details. Refer below pic for a sample Infra Dashboard.

#### **Infra Dashboard**

The Infra Dashboard helps in monitoring the infrastructure related issue like: Thread counts, memory utilization, CPU utilization, Solr, hbase health, etc

 Please refer link Infrastructure Alerts for more details .Refer below pic for a sample Infra Dashboard.

#### Batch Job Dashboard

The dashboard helps in monitoring batch related issues like job failure, job running status and step at which job is currently running as per the logs. Please refer link Batch Job for more details. Currently we have no sample for Batch Job Dashboard.

# Error / Exception Handling

**Note:** Currently we have Error Handling Mechanism for Java and Unix only, will be coming up with enhancements for other languages soon.

1. **Error Handling Mechanism**
   1. Our team recommends as below for handling errors/exceptions  :

**UNIX**

There is no try/catch in bash; however, one can achieve similar behavior using && or ||

Using ||: If command1 fails then command2 will run as :   < command1 || command2  >    < command1 && command2>

The closest approximation of try/catch is as follows :

{ # try

command1 &&

#save your output

} || { # catch  
# save log for exception   
}  
Also bash contains some error handling mechanisms, as well

set -e  
It will immediately stop your script if a simple command fails"

**JAVA**

"add try catch blocks 34 public void getBookIds(int id) {  
35 try {  
36 book.getId(id); // this method it throws a NullPointerException on line 22  
37 } catch (NullPointerException e) {  
38 throw new IllegalStateException(""A book has a null property"", e)  
39 }}  
This might give you a stack trace that looks like:

**try statement —**[**http://docs.python.org/reference/compound\_stmts.html#try**](http://docs.python.org/reference/compound_stmts.html#try)**exceptions —**[**http://docs.python.org/library/exceptions**](http://docs.python.org/library/exceptions)**"**

1. **Exception Trace :**to be shown which includes reference to line no,function name ,issue source

**JAVA**

stack trace code

1. **Customized Error:**  
   All error messages should be descriptive and easy to understand for error condition for all exception and error

# Resiliency

|  |  |
| --- | --- |
| NFR: | Resiliency |
| Definition: | * Resilience is the capacity of a system to continually change and adapt yet remain within critical thresholds * Resiliency is the ability of a server, network, storage system, or an entire data center, to continue operating even when there has been an equipment failure, power outage or other disruption. * From a software perspective, resiliency specifies the capability of the software to maintain its performance over time * Unreliable software fails frequently, and certain tasks are more sensitive to failure (for example, because they were not designed with restart capability, or because they must be run at a certain time) |
| Guidance: | Checkpoints / Recovery: For all distributed applications appropriate checkpoint / recovery logic must be designed into and maintained by:     > Batch jobs should be designed to include checkpoints to allow re-start from the point of failure.     > The re-start / re-try process must be automated. There should be a limited and configurable number of re-tries to avoid a failed process re-starting indefinitely     >  Large, long running jobs should be avoided and should be broken down into several smaller jobs or have multiple checkpoints to allow re-start.     > Batch jobs should identify, record, and bypass noncritical failures to allow the greater process to complete. This allows for expedited issue investigation. |
|  | Other Resources: 1. USAA Reference Architectures 2. USAA RA Team Prescriptive Guidance |
| Examples: |  |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Checkpoint and Recovery logic is implemented and successfully tested. |

# Sustainability / Maintainability

Sustainability is maintainability of a system or application including code quality, extensibility, testability, usability, and ability to be repeatedly and easily deployed. While we will discuss the best practices, tools and recommendations for each of these areas in the sub-sections, the below tabular table provides an overview of the key aspects of Sustainability

|  |  |  |
| --- | --- | --- |
| ****NFR**** | ****Definition**** | ****Tools Used to Measure / Validate**** |
| Code Quality | Software or code quality refers to two related but distinct notions that exist wherever quality is defined in a business context. Software/code functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications. That attribute can also be described as the fitness for purpose of a piece of software or how it compares to competitors in the marketplace as a worthwhile product. Software/code structural quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly. Code quality can be achieved through Code Reviews, Code Coverage of tests, reduced defect level by phase (‘shift left’), Non-Functional Requirements throughout process, feedback loop from production, and reporting driven to developer’s desktop  (visual and real-time); dip in metrics would trigger action. | CAST  Sonar  Checkstyle |
| Maintainability | Maintainability is the ease with which a product can be maintained in order to isolate defects or their cause, correct defects or their cause, prevent unexpected breakdowns, maximize a product's useful life, meet new requirements, make future maintenance easier, or cope with a changed environment. In short, maintainability is defined as the probability of performing a successful repair action within a given time and measures the ease and speed with which a system can be restored to operational status after a failure occurs. | Review Board  Crucible |
| Deployment | Deployment is the tools and processes, ideally fully automated, that take artifacts and put them on a desired development, testing, staging, or production system. Ideally the deployment model would support tools and processes that totally minimize or eliminate any manual intervention. The deployment package can consist of installers, binaries, and any configuration data that is required to install and configure an application or system to ensure a fully functioning deployment. |  |
| Extensibility | Extensibility (not to be confused with forward compatibility) is a system design principle where the implementation takes future growth into consideration. It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality. The central theme is to provide for change – typically enhancements – while minimizing impact to existing system functions. Extensibility is a design principle defined as a system’s ability to have new functionality extended, in which the system’s internal structure and data flow are minimally or not affected, particularly that recompiling or changing the original code is unnecessary when changing a system’s behavior, either by the creator or other programmers. Because systems are long lived and will be modified for new features and added functionalities demanded by users, extensibility enables developers to expand or add to the capabilities and facilitates systematic reuse. | JUnit  NUnit  Coburtera  NCover |
| Usability | Usability is the ease of use and learnability of a human-made object. The object of use can be a software application, website, book, tool, machine, process, or anything a human interacts with. Usability includes methods of measuring usability, such as needs analysis and the study of the principles behind an object's perceived efficiency or elegance. In human-computer interaction and computer science, usability studies the elegance and clarity with which the interaction with a computer program or a web site (web usability) is designed. Usability can be driven by look and feel standards - screen element density, layout and flow, colors, UI metaphors, keyboard shortcuts, and internationalization / localization requirements – languages, spellings, keyboards, paper sizes, etc. | HP QC |
| Testability | Testability is the degree to which a software artifact (i.e. a software system, software module, requirements- or design document) or system supports testing in a given test context. If the testability of the software artifact or system is high, then finding faults in the system (if it has any) by means of testing is easier. Testability is not an intrinsic property of a software artifact or system and can not be measured directly (such as software size). Instead testability is an extrinsic property which results from interdependency of the software or system to be tested and the test goals, test methods used, and test resources (i.e., the test context). A lower degree of testability results in increased test effort. In extreme cases a lack of testability may hinder testing parts of the software or software requirements or system at all. | Jenkins  LARA  LISA  XL Release |
| Accessibility | Accessibility refers to the design of products, devices, services, or environments for people with disabilities.The concept of accessible design ensures both “direct access” (i.e. unassisted) and "indirect access" meaning compatibility with a person's assistive technology (for example, computer screen readers). Accessibility can be viewed as the "ability to access" and benefit from some system or entity. The concept focuses on enabling access for people with disabilities, or special needs, or enabling access through the use of assistive technology; however, research and development in accessibility brings benefits to everyone. |  |

# Code Quality and Deployment

Of all the 7 Sub-NFR's in Sustainability - Code Quality & Deployment have specific tools in place today to help cover the ***"How"*** part in the development life cycle itself. The other NFR's are much more intangible & dependent on the overall Architecture and design of the system. For example, a Micro-services based architecture deployed on cloud will perhaps have a far better maintainability and extensibility compared to a custom deployed monolithic application. These aspects are usually covered by the Dev-Ops team while the regular participation of the Dev-Ops team in the scrum calls and in the initial design stages.

### This section has details about the recommended code quality & deployment practices.

Current recommendation is for JAVA. Recommendations for other programming languages will be coming soon. Software or code quality refers to how it meets non-functional requirements that support the delivery of the functional requirements, such as robustness or maintainability, the degree to which the software was produced correctly. Code quality can be achieved through Code Reviews, Code Coverage of tests, reduced defect level by phase (‘shift left’), Non-Functional Requirements throughout process, feedback loop from production, and reporting driven to developer’s desktop (visual and real-time); dip in metrics would trigger action.

| **Name** | **Recommendations** | | **Tools** |
| --- | --- | --- | --- |
| Code Quality | We recommend to use mentioned tools to test quality of code and application | | Jmeter/sonar cube/ PMD / Findbugs / Checkstyle / CAST |
| Deployment | We recommend deployment should happen automatically [based on code commit]  or one click (after code changes are committed in repository is done) Application team convenience | | Jenkins / LARA |
| NFR: | | Code Quality | |
| Guidance: | | Naming Convention: 1. A consistent and contextually relevant naming convention should be used for variables, constants, functions, files, UI labels, etc. to assist in code searches and troubleshooting.  Enumerations Instead of Constants: 1. Enumerated Lists or lists selected from the database should be used wherever possible instead of hard coded constants. This allows maintenance/modifications to the list to require less effort.  Data Validation: 1. Date validation is critical to the efficient operation therefore data from input feeds and User Interface (UI) should be validated before processing/storage. | |
|  | | Code Quality Can Be Achieved Thru:     • Code Reviews     • Code Coverage of tests     • Reduced defect level by phase (‘shift left’)     • Non-Functional Requirements throughout process     • Feedback loop from production     • Reporting driven to developer’s desktop  (visual and real-time); dip in metrics would trigger action | |
|  | | Other Resources: 1. USAA Reference Architectures 2. USAA RA Team Prescriptive Guidance 3. USAA Technical Road Maps 4. USAA Standards and Policies | |
| Examples: | |  | |
| Attachments: | |  | |
| Associated Standards: | |  | |
| Version: | | 1 | |
| Acceptance Criteria: | | Validated by code review. | |

# Extensibility

|  |  |
| --- | --- |
| NFR: | Extensibility |
| Details | * It is the ability to add additional functionality or modify existing functionality without impacting existing system functionality * We cannot measure extensibility when the system is deployed, but it shows up the first time you must extend the functionality of the system * We should consider the following when you create the architecture and design to help ensure extensibility: low coupling, interfaces, and encapsulation |
| Guidance: | 1Up front design needs to allow for addition, where dditions are to be made in small, incremental steps, and where work elements are to be separated into comprehensible units 2 Extensible source code must be supported with extensible build systems, directory trees, database access mechanisms, and so on  3 Code is more likely to be extensible if you follow the SOLID Principle (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion) Many of these principles come for free if you do testdriven development 4 Write modular code that is loosely coupled This allows for extension in stages 5There are three different forms of software extensibility: white-box extensibility, gray-box extensibility, and black-box extensibility, which are based on what artifacts and the way they are changed 6 White-Box Extensibility: Under this form of extensibility, a software system can be extended by modifying the source code, and it is the most flexible and the least restrictive form There are two sub-forms of extensibility, open-box extensibility and glass-box extensibility, depending on how changes are applied Open-Box extensibility, changes are performed invasively in open-box extensible systems; ie original source code is directly being hacked into It requires available source code and the modification permitted source code license Open-box extensibility is most relevant to bug fixing, internal code refactoring, or production of next version of a software product Glass-Box extensibility, (also called architecture driven frameworks) allows a software system to be extended with available source code, but may not allow the code to be modified Extensions have to be separated from the original system in a way that the original system is not affected One example of this form of extensibility is object-oriented application frameworks which achieve extensibility typically by using inheritance and dynamic binding 7 In black-box extensibility (also called data-driven frameworks) no details about a system’s implementation are used for implementing deployments or extensions; only interface specifications are provided This type of approach is more limited than the various white-box approaches Black-box extensions are typically achieved through system configuration applications or the use of application-specific scripting languages by defining components interfaces 8 Gray-box extensibility is a compromise between a pure white-box and a pure black-box approach, which does not rely fully on the exposure of source code Programmers could be given the system’s specialization interface which lists all available abstractions for refinement and specifications on how extensions should be developed |
| Examples: |  |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Teams thru architecture review and peer code reviews will place extensible technical debt related user stories in their back log |

# Maintainability

|  |  |
| --- | --- |
| NFR: | Maintainability |
| Definition | Maintainability is the ease with which a product can be maintained in order to isolate defects or their cause, correct defects or their cause, prevent unexpected breakdowns,maximize a product's useful life, meet new requirements, make future maintenance easier, or cope with a changed environment  In short, maintainability is defined as the probability of performing a successful repair action within a given time and measures the ease and speed with which a system can be restored to operational status after a failure occurs. |
| Guidance: | **Maintenance and Change Windows:** 1. The time periods in which changes can be made will be based on the Configuration Items (CI’s) that are associated with each ServiceNow RFC (Request For Change).   (Note:  A CI is any component [e.g. hardware, software, etc.] that can be managed to deliver an IT service.) 2. There is specific key information that must be defined for each CI such as the groups (e.g. Owner, Approvers, Reviews) that provide support for this CI as well as a maintenance schedule that determines when changes be implemented. If possible, maintenance windows should be designed to have zero downtime. 3. This maintenance or change schedule represents the time period in which in which any change (e.g. install a security or OS patch, application code deployment, etc.) can be performed.  (Note:  This time period must be aligned and agreed upon by the business as well as GBCC.) 4.  Down time during the scheduled maintenance window should not count negatively towards business availability metrics. 5.  Change windows should be clearly documented.  Global applications need to take into account the servicing hours of each market to assess whether separate infrastructure is required for certain markets. 6. All systems should follow the established change management processes. This includes Change Quality and Release Management reviews prior to implementing any changes in production. |
|  | Third Party Software Usage: 1. For all Third Party Software leveraged as an application or comprising a component of an application  a comprehensive list of information pertaining to the maintenance contracts should be provided.  This list must include:     • Software Name and Version     • Software Vendor Name     • USAA relationship manager for software or service provider     • Third Party Vendor Contacts       Third Party Support terms, hours of support, issue engagement process, etc.      • Support Contract End date/Renewal date,      • Corresponding cost center to be charged for contracts renewal, etc.     • Purchase record and order # leveraged to procure the software     • Software Licensing            > If software licenses are required there should be an alerting system in place for license renewal to occur well in advance of the expiration date.             > It should be made clear as to which group is responsible to renew any licenses prior to deployment.              >  It should be made clear as to which group will manage and administer those licenses  2. Additional Third Party Considerations     • Impact communications protocol     • Bridge participation     • Change management and release management processes     • Monitoring and alerting     • Incident and problem ticket “bridging” or correlation with USAA ticketing systems     • Impact and problem ownership |
|  | **Third Party Fufillment and Software as a Service:** 1. If third parties are engage to perform certain activities to complete a process such as  offer fufillment or Software as a Service such as Credit bureau reporting, a comprehensive list of information pertaining to vendor should be provided.  This list must include: • Description of processes the service is providing   What USAA AIM IDs and processes have a dependency on the vendor  Impact if third party cannot provide the service  • Alternate flows if applicable •  Vendor Name • Third Party Vendor Contacts • Third Party Support terms, hours of support, issue engagement process, etc  • Support Contract End date/Renewal date     • Third Party hours. The hours of support should be considered in respect to impacts on availability SLAs.     2. Additional Third Party Considerations     • Change Management     • Impact communications protocol including advance notification of outages     • Bridge participation     • Change management and release management processes     • Monitoring and alerting     • Incident and problem ticket “bridging” or correlation with USAA ticketing systems     • Impact and problem ownership |
|  | **Decommissioning:**  If certain hardware or software is being decommissioned, a decommissioning plan needs to be in place whereby the decommissioned components can be recovered. |
|  | **Communication:** 1. A detailed list of business contacts (distribution lists) should be provided for notification and/or escalation to ensure appropriate Incident and/or Problem communication flows to all impacted parties. 2. All existing applications which are substantially modifying or adding additional capabilities and / or functionality must engage and provide assistance in the creation of support documentation for use by the Service Desk and other technical support resources prior to the date of implementation of the modified application/system. Please note this activity must be initiated in the Build Phase to insure adequate lead time is available to complete the support documentation prior to implementation. The process initiation occurs through the submission of a “Help Desk Operational Support – Engagement” ITSC Request as listed within the IT Service Catalog. Dependent upon complexity of the User Case.  3. Stakeholders should be identified to whom the support team can raise risks, trends, and provide other pertinent and actionable items. |
|  | **Servicing:** 1. If this is a new process/product and there is a customer base which requires servicing, the servicing group needs to be funded, trained and aligned to service the customers.  2. If this is an enhancement to an existing process/product, the servicing group needs to be funded, trained, and aligned to support any new capabilities.  3. The servicing group needs to be equipped with all the necessary tools to support the customer base. |
|  | **Continuous Delivery**: Continuous Delivery can be achieved through: 1. Automated Build 2. Automated Test (Unit, Integration, User Acceptance, Performance) 3. Automated Deploy  4. Production-like Test Environments on-demand  Use of automation should be incorporated into the design: 1. unit test should be built into all code 2. Automated regression testing scripts should be created for all functionality using service virtualization tools such as LISA   3. the use of software quality measurement tools such as CAST should be considered 4. Automated deployment tools such as LARA should be considered |
| Examples: |  |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Maintenance and Change Windows: The agreed upon maintenance and functional changes windows should be documented in the Configuration Item.  Third Party Software Usage: Documented procedures for third party should be in place.  Decommissioning: Plan submitted to capacity planning team for application and saves documented in EXCEL application.  Communication: Turnover documentation is signed off by the SDM team .  Servicing: Documented business alignment.  Continuous Delivery: Successful execution of automation including negative test cases. |

# Testability

|  |  |
| --- | --- |
| NFR: | Testability |
| Definition: | * Testability is the degree to which a software artifact (i.e. a software system, software module, requirements- or design document) or system supports testing in a given test context. * If the testability of the software artifact or system is high, then finding faults in the system (if it has any) by means of testing is easier. * Testability is not an intrinsic property of a software artifact or system and cannot be measured directly (such as software size). * Instead testability is an extrinsic property which results from interdependency of the software or system to be tested and the test goals, test methods used, and test resources (i.e., the test context). * A lower degree of testability results in increased test effort. In extreme cases a lack of testability may hinder testing parts of the |
| Guidance: | 1. Make sure that the code is easy to understand and designed to be flexible. 2. Implementing meaningful peer review to find out if the code really is easy to understand and designed to be flexible  3. Design for software testability. 4.  Adopt Test Driven Development (TDD) into your SDLC process. 5. Write the test first, then the code. 6. Design classes using dependency injection. 7. Separate UI code from its behavior using Model-View-Controller or Model-View-Presenter. 8. Do not write static methods or classes. 9. Program using interfaces not concrete classes. 10. Isolate external dependencies. 11. Mark methods as virtual for methods that you intend to mock. 12. Ensure developers write testable code and the actual tests are used to verify the code before committing changes to the source code repository. 13. Establish a code coverage metric and incorporate this metric into your CI pipeline. 14. Establish a consistent Test Data Suite that can be leveraged as part of your test strategy. |
| Examples: |  |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Teams will incorporate a TDD practice into their delivery pipeline. Teams thru architecture review and peer code reviews will place TDD technical debt related user stories in their back log. |

# Performance

In this section we would work on recommendations for Performance NFR and its description. Performance is the speed at which content is delivered to users and how responsive the system is and ability of a system or application to perform and scale under various conditions

We recommend setting threshold for below key indicator which would help in monitoring the application and generating alerts in case of failure.

| **Key Indicators** | **Description** |
| --- | --- |
| **Processing time** | Overall Interaction time for request response threshold should be set and recorded to track performance of application as whole |
| **Response time** | Server Response time to customer requests in less than one second or time setup by Application team |
| **Querying time** | Querying the database/applications like Solr or HBase,etc should take less than setup by Application team |
| **Standard NFR's** | **Tools used to measure / Validate** |
| Capacity and Scalability | HP Load Runner |
| Performance Testing | DynaTrace  AppDynamics  CAST |
| Redundancy / Failover | JMeter |
| Availability |  |

# Availability

|  |  |
| --- | --- |
| NFR: | Availability |
| Definition |  |
| Guidance: | End to End Processing: 1. If an interactive application’s availability results are to be measured or reported back to the business, an appropriate automated mechanism must be incorporated to provide the necessary metrics that make up the measurement.  2. Interactive applications must meet the established availability Service Level Agreements (SLA). SLAs must be clearly defined, documented, aligned and proven   3. The following components must be included or addressed to achieve availability SLA:  To deliver a product each application within the end to end process must have established and agreed upon levels of service for delivery.  The aggregate time consumed for the E2E processes cannot exceed the expectations established with the customers of that end product. A mechanism must be defined to measure end user experience The end-to-end process must be designed, tested, and validated to have appropriate checkpoints and tracking for each step automated within the process to identify any degradation of performance   4. All availability measurements reported must reflect any agreed upon impacts from planned Maintenance and or Functional Change windows in alignment with the business requirements for that product and/or application |
|  | Critical Business Processes: For any critical business process, particularly those with a high risk of control and compliance issues,  build in such a way to mitigate any potential issues.       •  Design application with a way to recover from failure events    •  Gather all data required to successfully complete the entire process up front to avoid issues with fulfillment if process can't complete with partial data.     •  Test connectivity to any services that are critical to fulfillment prior to presenting offers   •  Consider alternate process flows in the event that some interfaces are not available.    In the event of a complete outage, consider what alternate methods are available to complete the process e.g. use of green screens; downlines etc. |
| Examples: |  |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | End to End Processing: Agreed upon SLAs must be documented and the appropriate measuring mechanisms implemented to ensure that the SLA is measured and deviations are reported on so that appropriate action can be taken to circumvent SLA degradation.  Critical Business Processes: Successful testing of negative use cases. |

# [Capacity and Scalability](https://enterprise-confluence.aexp.com/confluence/display/GMD/Capacity+and+Scalability)

This section has details and recommendations about the NFR Capacity and scalability and benefits from using them.

**Capacity** is  the amount of resources made available to the system

**Scalability** is the ability of a system, network, or process to handle a growing amount of work in a capable manner or its ability to be enlarged to accommodate that growth.

* For example, it can refer to the capability of a system to increase its total output under an increased load when resources (typically hardware) are added. Scalability, as a property of systems, is generally difficult to define and in any particular case it is necessary to define the specific requirements for scalability on those dimensions that are deemed importan.
* Methods of adding more resources for a particular application fall into two broad categories: horizontal and vertical scaling.
  1. To scale horizontally (or scale out) means to add more nodes to a system, such as adding a new computer to a distributed software application.
  2. To scale vertically (or scale up) means to add resources to a single node in a system, typically involving the addition of CPU or memory to a single computer.It provides more resources for the hosted set of operating system and application modules to share.   
       
     Below Table consists of  recommendation on different parameters for Capacity and Scalability NFR

|  |  |
| --- | --- |
| **Name** | **Recommendation** |
| **Storage** | The system must store data effectively and must anticipate the time remaining until all available storage is filled up |
| **Growth requirement** | The storage estimates for upcoming 1 year as the system gets used more |
| **Throughput** | We recommend setting up system throughput threshold as for how many request system should be able to handle for queries per minute, with the capability to increase with larger demand |

# [Performance Testing](https://enterprise-confluence.aexp.com/confluence/display/GMD/Performance+Testing)

In this section we are going to describe Performance Testing (NFR) and how it helps to assess production readiness. Performance testing is a type of testing intended to determine the responsiveness, throughput, reliability, and/or scalability of a system under a given workload.

We achieve it with below:

* Evaluate against performance criteria
* Compare performance characteristics of multiple systems or system configurations
* Find the source of performance problems
* Support system tuning
* Find throughput levels

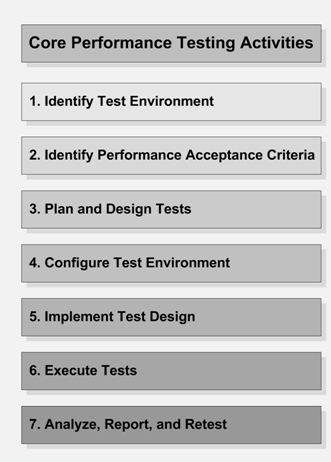
**PERFORMANCE Testing Techniques**

Load testing: A type of performance testing which helps to evaluate whether the expected performance is met or not under the actual conditions of use.

Stress testing: A type of performance testing designed to determine the maximum load the application can handle and breakpoint of the application.

Volume testing: A type of performance testing where the application behaviors is evaluated when the application is subjected to large volumes of data.

**Core Activities of Performance Testing** :Activities recommended below for achieving better production system using Performance Testing:



# [Redundancy / FailOver](https://enterprise-confluence.aexp.com/confluence/pages/viewpage.action?pageId=145584708)

|  |  |
| --- | --- |
| NFR: | Redundancy/Failover |
| Guidance: | 1.  Redundancy should be employed at all levels of the application for the purpose of load balancing, fault tolerance, and failover (web, app, DB servers). 2. Testing environments should validate this redundancy prior to going live (settings, scenarios, recovery time, etc.). 3. If a redundant architecture is required it should be employed at all levels of the application for the purpose of load balancing, fault tolerance, and failover (web, app, DB servers). 4. If redundancy is required, the testing environments should validate this redundancy prior to going live (settings, scenarios, recovery time, etc.) |
|  | Other Resources: 1. USAA Reference Architectures 2. USAA RA Team Prescriptive Guidance |
| Examples: | Queue failover - test the failover of a given queue. Application failover - test that application continues to run in a cluster if application server goes down. Database failover - test that database continues to run in a cluster if database server goes down. |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Documented test results for implementation of redundancy with signoff from SDM. |

# [Recovery](https://enterprise-confluence.aexp.com/confluence/display/GMD/Recovery)

This section has details about the recommendations about different ways to recover application. Recovery is the ability for a system to prepare and respond to a disaster.

Below is the table consisting of recommendations for recovering application and to prepare it for such situations.

|  |  |  |
| --- | --- | --- |
| **Name** | **Recommendation** | **Tools Used To Measure / Validate** |
| **Disaster Recovery Document** | We recommend application to have DR Plan ready for the application system where it should be consisting of information about details such that referring to it restoration to working state is achievable. | GDHA  GTM  DR Solution Design |
| **Backup frequency** | The above document update frequency should be near to 6 -12 months | Avamar |
| **Data Retention Plan** |  | USAA Audit and Compliance |

# [Back Up Plan](https://enterprise-confluence.aexp.com/confluence/display/GMD/Back+Up+Plan)

|  |  |
| --- | --- |
| NFR: | Backup Plan |
| Guidance: | Operational Backups: 1. Operational Backups should include: • Full database backups should be taken weekly • Full system should be taken weekly. • For full Database and System backups conducted on a weekly basis Daily Incremental backups must be taken to insure complete recovery capability is achievable • Applications requiring Disaster Recovery  restoration within 24 hours or less must have  full daily backups |
| Examples: |  |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Backup and recovery processes are successfully tested. |

# [Data Retention](https://enterprise-confluence.aexp.com/confluence/display/GMD/Data+Retention)

|  |  |
| --- | --- |
| NFR: | Data Retention Plan |
| Guidance: | 1. Business records and data should be stored and retained in accordance with the following Policies, Standards and Procedures:  • Records Management Policy -  • Information Backup and Restoration Standard  • Electronic Records Backup and Recovery Standard  • Electronic Records Backup and Recovery Procedure  2. A well-defined purge process shall be designed in the original implementation that includes purging the following in accordance with business and compliance/regulatory requirements:  • Database Tables  • Application Log Files  • Error Log Files  • Feed input/output Files |
| Attachments: |  |
| Associated Standards: |  |
| Version: | 1 |
| Acceptance Criteria: | Documented SDM alignment. Record retention conforms to policy. |

# [Security](https://enterprise-confluence.aexp.com/confluence/display/GMD/Security)

This section has details about the recommendations about different ways to secure application. Below is the table consisting of recommendations for securing application and to prepare it for such situations.

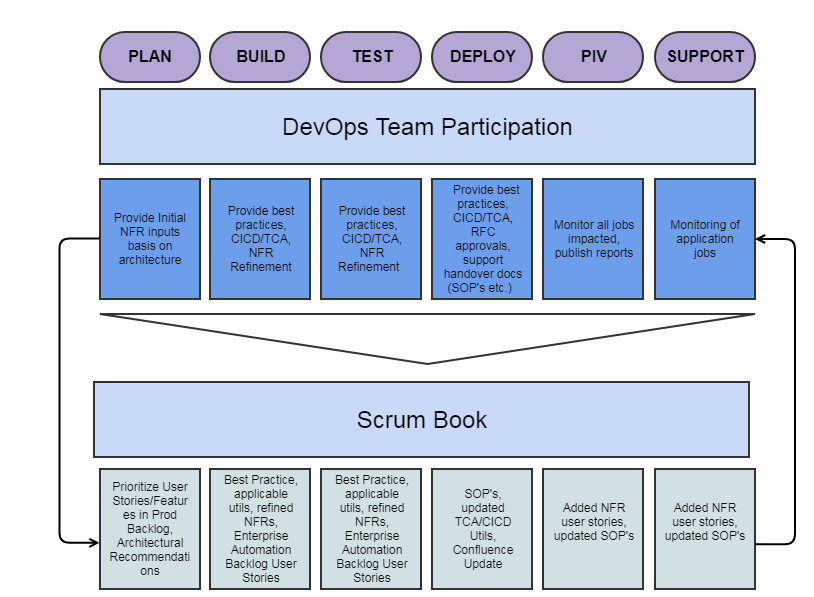
|  |  |  |
| --- | --- | --- |
| **Name** | **Recommendation** | **Tools Used To Measure / Validate** |
| **Security Compliance** |  | USAA Security Compliance |
| **Security Scans** |  | Trustwave (Cenzic) |

# [USAA Security Compliance](https://enterprise-confluence.aexp.com/confluence/display/GMD/Amex+Security+Compliance)

|  |  |
| --- | --- |
| NFR: | Security Compliance |
| Guidance: | User ID/Password/Data Administration: 1. In cases where user access to a production application and/or data is necessary for a particular role, function or need, an Access Administrative Process must be leveraged in accordance with the User Access Management Procedure defined within Access Control Standard. The following criteria must be in place:  • Defined Process and execution Owner for access provisioning, GBCC will not be responsible for and/or own Access Administration processes • Process must conform with the Identity Management Lifecycle guidelines as defined. • Access must be provisioned and entitled using an approved provisioning process or tool as defined within the Information Security Standards  • Access must be associated with an active Employee or Contractor Number (ECN), not applicable for external customer facing applications  • Be disabled or suspended when inactive for 180 days or less  • Be deleted when inactive for 365 days  • Be disabled upon termination of an individual’s employment or business relationship  • Be re-certified for need and entitlements when the owner is transferred or has a job code change |
|  | Faceless IDs: 1. Applications should use Faceless IDs as per IS09.02 ID Management standard.  2. Where compliance restricts the use of non-expiring Faceless IDs, an Application ID management plan shall be developed to minimize manual effort in changing Application ID passwords (including minimizing the number of required application IDs). |
| Examples: |  |
| Attachments: |  |
| Associated Standards: | Access Control Standard Information Access Policy |
| Version: | 1 |
| Acceptance Criteria: | User ID/Password/Data Administration: User IDs and Passwords comply with the security requirements.  An owner is documented and the alignment on the ownership is documented.  Faceless IDs: Faceless IDs comply with standard.  An Application ID management plan is documented. |

# [Integrated Scrum & Feedback Loop](https://enterprise-confluence.aexp.com/confluence/pages/viewpage.action?pageId=142154991)

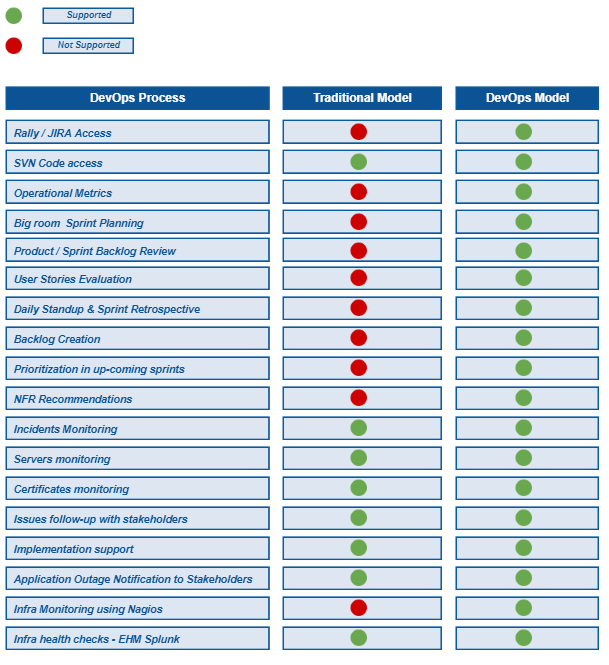
DevOps plays an integral role across the life cycle of an application. As indicated in previous sections - DevOps teams collaborates with various stake holder across the ecosystem to ensure smooth running of the application, however the below diagram shows the close working of the DevOps team with the development tea



**Below table has description how we integrate with different scrum calls:**

| **Different Scrum Calls** | **Our Role** |
| --- | --- |
| **Standup Call & Feedback Loop** | * To understand the current sprint stories and their impact to production * We share recommendations from our experience about changes required * For e.g. logging as recommended for splunk dashboard, logging for bad response along with request, etc * Discussion of prioritized issues with Dev team which would help understand different aspects of it and to accelerate its resolution * To track all RFC schedules and changes being implemented through them to ensure better support * Discussion regarding repeated impacts or repeated tickets |
| **Pre-Sprint Homework** | * To create the user stories in Backlog in rally before  going to sprint call .The user story can be our additions to the application where we automate any manual task or steps we think are required which would add to support of application |
| **Sprint Planning Calls and Backlog Grooming session** | * To bring up our added user stories and ideas to business partners & Dev team to discuss the benefits after implementing * Work on understanding other functionality being planned for development and forecast how it is going to impact existing usage in E3 |
| **Weekly DevOps Call with Application Team** | * It is recommended to have weekly calls with DevOps to better align on upcoming changes to production & future plans |
| **Sprint Retrospective Meetings** | •Discussing Business functionality which was created during the Sprint is demonstrated to the Product Owner and its plan to move into E3 and impact to E3 |

# [Traditional v/s DevOps Model](https://enterprise-confluence.aexp.com/confluence/pages/viewpage.action?pageId=143178354)



# Production Support Task List (SO / Dev)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | Subcategory | Task | Operations | Development |
| Access Management | Development | Service ID Management, Access Request Escalations, etc for Development Sites/Tools |  | X |
| Access Management | Service Delivery | Service ID Management, Access Request Escalations, etc for Service Delivery Sites/Tools | X |  |
| Business Continuity | Business Continuity Execution | Execute Disaster Recovery Plan |  | X |
| Business Continuity | Business Continuity Planning | Create and update Disaster recovery plan and other business continuity activities |  | X |
| Business Continuity | Business Continuity Testing | Participation in DR Activities. (Including Pre-planning, Approvals, and post-activities) |  | X |
| Change Management | Application | Scheduled database job modification activities (Hold/Stop/Restart/Rerun) | X |  |
| Change Management | Application | Development Change Review (Code review, Walkthrough, Transition docs, Release plan, Feedback). |  | X |
| Change Management | Application | Identify proactive steps to avoid potential issues from a change, such as holding a batch job. |  | X |
| Change Management | Application | Post implementation support/Validation for application changes |  | X |
| Change Management | Batch | Adhoc Batch Job execution. |  | X |
| Change Management | Certificates | Certificate renewals  (If Certificate exists inside code) |  | X |
| Change Management | Certificates | Certificate renewals  (If Certificate exists outside code) | X |  |
| Change Management | Communication | Change Impact Notifications | X |  |
| Event Management | Alerts | Update existing alerts and turn off obsolete alerts as necessary. |  | X |
| Event Management | Alerts | Identify alerting gaps and opportunities | X |  |
| Event Management | Application | Analyze Warning or Alert(Pre-Incident) | X |  |
| Event Management | Application | Create new application alerts as needed to effectively monitor the platform. |  | X |
| Event Management | Dashboards | Create new dashboards for operational needs | X |  |
| Event Management | Dashboards | Create new dashboards for development, and business needs |  | X |
| Event Management | Monitoring | Daily Monitoring of all production services (Alerts, Tickets, Servers, Services, Batch Jobs, Distributed systems, Dynatrace/Gomez, App Health, Payment, Capacity and SLA monitoring | X |  |
| Incident Management | Application | Adhoc scripts/queries to identify incident impacts. |  | X |
| Incident Management | Application | Advanced Application Log Analysis |  | X |
| Incident Management | Application | Advanced Server/Virtual Server service restarts (JVMs, Windows, Linux, etc.). Such as when special application procedures are needed. |  | X |
| Incident Management | Application | Circumvention of documented application issues | X |  |
| Incident Management | Application | Circumvention related to undocumented /complex Application issues |  | X |
| Incident Management | Application | Detailed analysis of Application performance issues (e.g.: SP failures, long running queries, Jobs, Timeout issues, deadlocks, etc.) |  | X |
| Incident Management | Application | Initial Acceptance & Triage of all Incidents (Business or Technical) | X |  |
| Incident Management | Application | Initial Analysis of Application Performance issues (e.g.: SP failures, long running queries, Jobs, Timeout issues, deadlocks, etc.) | X |  |
| Incident Management | Application | Initial Application Log Analysis for Incidents | X |  |
| Incident Management | Application | Server/Virtual Server service restarts (JVMs, Windows, Linux, etc.) | X |  |
| Incident Management | Batch | Circumvention of documented Job Failures | X |  |
| Incident Management | Batch | Advanced Batch Job Failure Analysis / Circumvention |  | X |
| Incident Management | Batch | Batch Data Issue - Data modification required, override needed, or recovery after job completion. |  | X |
| Incident Management | Batch | Batch job modification activities (Hold/Release/Force/Bypass/Re-run) | X |  |
| Incident Management | Batch | Initial analysis of impact, design, or data related issues for Batch Jobs. Includes coordination with interfacing 3rd parties | X |  |
| Incident Management | Communication and Bridging | Incident Notification/Proactive communication using INS or other tools. | X |  |
| Incident Management | Communication and Bridging | Initiation of high severity bridge calls | X |  |
| Incident Management | Communication and Bridging | INS Distribution list updates & Maintenance | X |  |
| Incident Management | Communication and Bridging | Notifying Banks, downstream systems, and interfaces of impacted batch jobs (Delays, failures, etc.) | X |  |
| Incident Management | Communication and Bridging | Participation in High Severity Bridge Calls | X | X |
| Incident Management | Middleware support | Co-ordination for MQ, Datapower, or other middleware related issues | X |  |
| Knowledge Management | Knowledge Maintenance | Continually maintain existing Knowledge documents (Runbooks, Service Mapping, etc) | X |  |
| Knowledge Management | Knowledge Transfer | When rolling out new products. Provide Best practices, tools, processes, transition docs, and guidelines for support teams |  | X |
| Knowledge Management | SOP (Standard Operating Procedures) Creation | Create and maintain SOP documents for use by the ECC |  | X |
| Problem Management | Compliance / Corrective Actions | CAT/Compliance related activities |  | X |
| Problem Management | Coordination | After all Ptasks are closed, review problem ticket for accuracy and close. | X |  |
| Problem Management | Coordination | Problem Record Creation, Assignment of Ptasks, and mapping Incidents to the problem. | X |  |
| Problem Management | Coordination | Coordination of Post-Mortems / Follow-up calls for High Severity Issues | X |  |
| Problem Management | Development | Root Cause, Permanent Resolution,  and other assigned Ptask completion. |  | X |
| Reporting | Availability | Tracking and Reporting Application Availability | X |  |
| Reporting | SLA Management | Service Operations Ticket SLA Reporting | X |  |
| Reporting | SLA Management | Development Ticket SLA Reporting |  | X |
| Reporting | SLA Management | Infrastructure Ticket SLA Reporting | X |  |
| Request Fulfilment | Data Request | Adhoc Business/Data Requests |  | X |