**Gates NextGen Design Documentation**

**Document Version:** 1.0  
**Based on Confluence content as of:** [Date of Video, e.g., Dec 20, 2023, or latest modification date visible]

**Table of Contents**

1. Introduction: Gates NextGen Design  
   1.1. Overview  
   1.2. Goal  
   1.3. Background & Strategic Fit  
   1.4. Assumptions
2. Workflow & Design  
   2.1. Tekton Flow (Diagram)  
   2.2. CaaS API Flow (Diagram)  
   2.3. Flow Description
3. Rules Definition & Execution  
   3.1. CaaS (NextGen) Rules Definition & Execution  
   3.2. CaaS (CurrentGen) Rules Definition & Execution  
   3.3. Sample CaaS Drool Response
4. Developer Override / Increase Thresholds  
   4.1. threshold.json  
   4.2. sonarInput.json
5. Tekton Pipeline Gates  
   5.1. Tekton Control Step Diagram  
   5.2. Gate Types Legend
6. Developer Level Gates  
   6.1. Developer Thresholds  
   6.2. thresholds.json (Example)  
   6.3. Thresholds Description (Table)
7. Sector Level Gates  
   7.1. SonarQube  
   7.2. Snyk  
   7.3. Break Glass Exception Process  
   7.4. Maven Plugins Whitelist  
   7.5. Dependency Validator (Dx-Validator)  
   7.6. CDQR Gates  
   7.7. GEMs Gates
8. Dx-Validator (Detailed)  
   8.1. Overview  
   8.2. Code and Endpoints  
   8.3. How it Works
9. Pipeline Hard Gates  
   9.1. Overview  
   9.2. Specific Hard Gates (SonarQube, CDQR, GEMS, Snyk)  
   9.3. LightSpeed Enterprise Adoption Hard Gates  
   9.4. Hard Gates Exception Process  
   9.5. Upcoming Hard Gates (Roadmap)  
   9.6. Gates Roadmap (Table)  
   9.7. FAQ
10. GEMS (Global Event Management)  
    10.1. GEMS Vulnerability  
    10.1.1. Remediation Steps  
    10.2. GEMS Quality Gates  
    10.2.1. Seeing GEMS in Build Dependencies  
    10.2.2. Resolving GEM Warnings  
    10.2.3. Finding GEM Origins  
    10.2.4. GEM Information Table  
    10.3. GEMS (Global Event Management) - Main Page  
    10.3.1. Overview  
    10.3.2. Process Flow  
    10.3.3. Roadmap  
    10.3.4. Backlog  
    10.3.5. RACI Chart  
    10.3.6. Customer Process (Jira Dashboard, User Story Updates)
11. Hard-Gate Exception Processes  
    11.1. Hard-Gate Exception Request Process (Jira Workflow)  
    11.2. Hard-Gate Exception Approval Process (Detailed Steps)  
    11.3. Hard-Gate Exception Approvers (Table)
12. Unify Gates Parity (Table)

**1. Introduction: Gates NextGen Design**

* **Title:** Gates NextGen in LSE using OPA
* **Design Status:** APPROVED
* **Development Status:** IMPLEMENTED
* **Owner:** Sampath, Arunkumar (TECH)
* **Architect:** Mareedu Naga, Viswanath (TECH)

**1.1. Overview**  
This document describes the integration of PBWM controls with LSE pipelines.

**1.2. Goal**  
The goal is to integrate PBWM controls in LSE pipelines; each pipeline will be measured against these controls, essential for the pipeline to move to the next step.

**1.3. Background & Strategic Fit**  
LSE pipelines are missing controls that are required for PBWM applications. In order to migrate all the pipelines from Unify to LSE, we need to implement these controls in LSE.

**1.4. Assumptions**

* Control Engineers & App Managers create/update Rule Groups at Sector or CSI ID or LSE Project scopes as required.
* Developer can optionally override / increase thresholds for a particular pipeline using threshold.json file in the application source code repository.

**2. Workflow & Design**

**2.1. Tekton Flow (Diagram Description)**

* **Trigger:** Developer pushes to App Source Code Repository.
* **Pipeline Start:** Git clone, Compile.
* **Scans:** Sonar Scan, BDUnk Scan (BlackDuck presumably), Checkmarx Scan, GEM Scan.
* **Controls Step:**
  + Executes CaaS API.
  + Pulls Policies from OPA Bundle Server.
  + Generates Score (v1a) OPA (GEMS Metrics).
  + Runs OPA evaluation on policies, producing OPA Data (results).
  + **Conditional Flow (If Allow):**
    - Checks for threshold.json from the App Source Code Repository.
    - If threshold.json exists, runs OPA evaluation against it.
    - Produces OPA results score.
  + **Decision:** If all checks allow:
    - Continue Pipeline.
    - Publish App.
  + If any check fails, the pipeline step is marked as Failure.

**2.2. CaaS API Flow (Diagram Description)**

* **Input:** Execute CaaS API from Tekton or Harness.
* **Main Components:**
  + **Controls Orchestrator:** Collects Rules to run, Collects Data from Providers, Executes Rules against Drools.
  + **RuleMgr:** Manages rule groups by scope (Sector, CSDD, Project).
  + **Controls UI:** Allows App Managers/Controls Engineers to Create/Manage Approvals, Facts, Actions, and convert UI Policy into Drools Rule. Rules are stored in a RuleStore.
* **Data Providers:**
  + **External Data Providers:** CSI, CTC, CDQR.
  + **Pipeline Data Providers:** SonarQube, GEMScanner, Jira, One Approval.
  + **Break Glass Exception Providers:** Jira, One Approval.
  + **Rule Engine:** Drools.

**2.3. Flow Description**

1. Once pipeline is triggered below controls/checks will get executed, these controls will get executed once the steps like compilation (project building), sonar scan, Blackduck/snyk scan, checkmarx scan, GEM scan are executed successfully.
2. **CaaS (Current Gen):** CaaS determines the RuleGroups for the pipeline and execute them; pipeline will move to next step once allowed by CaaS else it is marked as Failure.
3. **CaaS (Next Gen):** In this step Pipeline will be checked against Policies pulled from OPA Bundle Server. eg: Sector level Sonar Hardgates (No blocker bugs or vulnerabilities in code).
4. Pipeline will move to next step if pipeline passed all the controls else it is marked as Failure.
5. **(optional) Developer Override / Increase Threshold:** This step will be executed if threshold.json file; otherwise pipeline will go next step. In this step developer's given quality conditions will be measured against pipeline's runtime values of Quality metrics from various tools like sonar, blackduck.
6. If pipeline passes these conditions it will move to next step or else it will be marked as Failure.

**3. Rules Definition & Execution**

**3.1. CaaS (NextGen) Rules Definition & Execution**

1. Control Engineer creates policy bundle with required rego & data files.
2. Pipeline will execute OPA evaluate to check policy using sector scope data file against runtime values of quality metrics from tools like SonarQube and GEM Scanner.
3. Based on the OPA result and JIRA exception approval pipeline will move to either next step or marked as Failure.

**3.2. CaaS (CurrentGen) Rules Definition & Execution**

1. App Manager or Control engineer can create rules in 3 scopes: csid, project, sector in Control UI and these RuleGroup will be available as an api.  
   *(Sample RuleGroups JSON shown - structure with csId, ruleGroupId, ruleName, sector, scope, project)*
2. CaaS is integrated with data sources like SonarQube, GEM, CTC, CS, CDQR, JIRA etc.
3. Before step CaaS API.
4. CaaS will execute the RuleGroups, during drool execution JIRA exception will be checked as well if it is present. eg: below is sample response from CaaS drool execution whose result is "REJECTED"

**3.3. Sample CaaS Drool Response (Illustrative)**

{

"PRWM-LI-001": {

"result": "REJECTED",

"findings": [

{

"rejection": "false",

"message": "No Condition evaluated to true for the rule - cdqrSoftGate"

}

// ... many similar entries ...

{

"rejection": "false",

"message": "No Data available"

},

{

"rejection": "true",

"message": "Certified Build" // Example of a specific finding

}

],

"additionalInfo": {

"sector": "PBWM"

},

"overallStatus": {

"result": "REJECTED",

"finding": null,

"additionalInfo": null

},

"Hard\_Soft": {

"result": "REJECTED",

"finding": {

"rejection": "true",

"message": "Not a certified master build"

},

"additionalInfo": {

"sector": "PBWM"

}

},

"GEM\_REST": {

"result": "APPROVED",

"finding": {

"rejection": "false",

"message": "No Data available"

},

"additionalInfo": {

"sector": "PBWM"

}

}

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Json

1. Based on the drool execution response, Pipeline will go to next execution step or it will be marked Failure.

**4. Developer Override / Increase Thresholds**

1. If threshold.json file is present in application source code repository, this control will get executed otherwise pipeline will move to next step.
2. threshold.json file which contains runtime values of sonar analysis metrics which pipeline must pass. eg: bugs, vulnerabilities, reliability\_rating etc.

**4.1. sample threshold.json**

{

"security\_rating": 4,

"reliability\_rating": 4,

"coverage": 30,

"test\_success\_density":100

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Json

**4.2. sample sonarInput.json (Input to OPA)**  
A json file with runtime values of sonar metrics of LSE project will be generated which will be given to OPA as input.

{

"sonar": {

"coverage": 0.0,

"security\_rating": 5.0,

"new\_vulnerabilities": 0,

"code\_smells": 435,

"security\_hotspots": 0,

"new\_security\_rating": 1.0,

"duplicated\_lines\_density": 15.3,

"new\_code\_smells": 0,

"vulnerabilities": 0,

"complexity": 10164,

"ncloc": 8836,

"new\_reliability\_rating": 1.0,

"bugs": 0,

"new\_bugs": 0,

"reliability\_rating": 5.0,

"sqale\_rating": 5.0 // (Likely A, B, C, D, E represented numerically)

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Json

1. OPA will evaluate the threshold.json with the input (sonarInput.json) given to it and generates its decision.
2. Pipeline will be marked as Failure in case OPA decision shows policy is violated.
3. If policy is not violated pipeline will move to next step.

**5. Tekton Pipeline Gates**

**5.1. Tekton Control Step Diagram Description**  
A pipeline triggers a "java-maven-task" which includes a "tekton-control-step". This step executes a series of controls sequentially if the previous one passes:

1. **Execute CaaS v1:** Scope (sector, csid, project), Controls: 8 (Hard Gate)
2. **OPA-CDQR:** Scope (sector), Controls: 1 (Soft Gate)
3. **OPA-SonarQube:** Scope (sector, csid, project), Controls: 8 (Hard Gate)
4. **OPA-GEMs Control:** Scope (sector, csid, project), Controls: 1 (Hard Gate)
5. **OPA-Snyk Control:** Scope (sector, csid, project), Controls: 1 (Hard Gate)
6. **Dependency-Validator control:** Scope (pipeline), Controls: 1 (Soft Gate)
7. **Maven-plugin control:** Scope (pipeline), Controls: 1 (Hard Gate)
8. **Sonar Scan Parameter control:** Scope (pipeline), Controls: 1 (Hard Gate)
9. **CCP config. validator control:** Scope (pipeline), Controls: 1 (Soft Gate)

**5.2. Gate Types Legend**

* **Green fill:** Live in LSE
* **Hatched fill:** Dev in progress
* **Solid line border:** Hard Gates
* **Dashed line border:** Soft Gates

**6. Developer Level Gates**

LightSpeed Enterprise Pipelines provides ability to user to optionally override / increase thresholds for a particular pipeline using thresholds.json file in root of application source code repository.  
**Current Scope of Implementation:** Java Maven Builds

**6.1. Developer Thresholds**  
Following is sample thresholds.json, if present pipeline should pass the given thresholds to mark it self success.

**6.2. thresholds.json (Example for Developer Level)**

{

"SonarQube": {

"security\_rating": 4,

"new\_security\_rating": 0,

"reliability\_rating": 3,

"new\_reliability\_rating": 0,

"bugs": 0,

"vulnerabilities": 0,

"security\_hotspots": 0,

"sqale\_debt\_ratio": 2,

"test\_success\_density": 80,

"test\_errors": 0,

"coverage": 80

},

"GEMS": {

"count": 0,

"criticalgem": 0,

"majorgem": 0,

"unspecificgem": 0

},

"Snyk": {

"vulnerabilities\_cvss\_score": 9 // Assumed, was cut off

}

}

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Json

**6.3. Thresholds Description (Table Summary)**  
| Tool | Threshold | Description | Possible Threshold Values (Example: SonarQube Security Rating) | Passing criteria | Pipeline Behavior When Threshold not present |  
| :----------- | :------------------------- | :-------------------------------------------- | :----------------------------------------------------------- | :----------------------------------------------- | :------------------------------------------- |  
| SonarQube | security\_rating | Represents severity of detected Vulnerabilities | 1-5 (1=A, 5=E, E means >0 Blocker) | Security Rating less than or equal to Threshold Value | Skipped |  
| SonarQube | new\_security\_rating | Represents severity of detected Vulnerabilities in new code | 1-5 | New Security Rating less than or equal to Threshold Value | Skipped |  
| SonarQube | reliability\_rating | Represents severity of detected Bugs | 1-5 (1=A, 5=E, E means >0 Blocker) | Reliability Rating less than or equal to Threshold Value | Skipped |  
| SonarQube | new\_reliability\_rating | Represents severity of detected Bugs in new code | 1-5 | New Reliability Rating less than or equal to Threshold Value | Skipped |  
| SonarQube | bugs | The total number of bug issues | Any Number greater or equal than 0 (exc 0) | Number of Bugs less than or equal to Threshold Value | Skipped |  
| SonarQube | vulnerabilities | The total number of vulnerabilities issues | Any Number greater or equal than 0 (exc 0) | Number of Vulnerabilities less than or equal to Threshold Value | Skipped |  
| SonarQube | security\_hotspots | The number of security\_hotspots | Any Number greater or equal than 0 (exc 0) | Number of Security Hotspots less than or equal to Threshold Value | Skipped |  
| SonarQube | sqale\_debt\_ratio | Ratio of actual technical debt... | Any Number greater or equal than 0 (exc 2) | Tech Debt Ratio less than or equal to Threshold Value | Skipped |  
| SonarQube | test\_success\_density | Percentage of successful unit tests | Any Number in Range (0-100) | Unit Test Success Rate greater than or equal to Threshold Value | Skipped |  
| SonarQube | test\_errors | Number of unit test errors | Any Number greater or equal than 0 (exc 0) | Number of Unit Test Errors less than or equal to Threshold Value | Skipped |  
| SonarQube | coverage | Percentage of source code covered with test cases | Any Number in Range (0-100) | Unit Test Code Coverage greater than or equal to Threshold Value | Skipped |  
| GEM Scanner | count | Number of GEMs in libraries used by application | Any Number greater or equal than 0 (exc 0) | Number of GEMs less than or equal to Threshold Value | Skipped |  
| Snyk Scanner | vulnerabilities\_cvss\_score | CVSS Score of Upgradable or Patchable Vulnerabilities | 0.0 - 10.0 | CVSS score less than or equal to Threshold value | Skipped |

**7. Sector Level Gates**

This page describes the types of PBWM Sector Level Gates for LightSpeed CI/CD pipeline.  
Includes: SonarQube, Snyk, Break Glass Exception Process, Maven Plugins Whitelist, Dependency Validator, CDQR Gates, GEMs Gates.

**7.1. SonarQube**

* Static code analyzer. Identifies issues (Bugs, Vulnerability, Security Hotspot, Code Smell).
* Types: Vulnerability, Security Hotspot, Bug & Code Smell.
* Severities: Blocker, Critical, Major, Minor, Info.
* LSE employs SonarQube for identifying these types of issues during CI-Build stage.
* Current SonarQube Hard Gates block pipelines when **Blocker Vulnerabilities** or **Blocker Bugs** are detected in all code base.
* **Scope of Implementation:** Java Maven Builds, NPM Builds, Python build, Java Gradle Build.
* **Type of Implementation:** Hard Gate with Break Glass Exception process.

**7.2. Snyk**

* Library analyzer. Identifies issues based on CVSS score.
* Severities: Critical, High, Medium, Low.
* LSE employs Snyk for identifying these types of issues during CI-Build stage. Snyk Hard Gates block pipelines when Critical Vulnerabilities are detected that are **upgradable** or **patchable**.
* **Scope of Implementation:** Java Maven Builds, Java Gradle Build.
* **Type of Implementation:** Soft Gate.

**7.3. Break Glass Exception Process**

* To formally track exceptions for hard gates, the pipeline will automatically create issues in JIRA. Application Managers can leverage the governed exceptions process to request temporary exclusion.
* **Type of Implementation:** Pipeline Hard Gates.

**7.4. Maven Plugins Whitelist**

* To regulate use of maven plugins across projects. If pipeline detects plugins not in the white-list, pipeline will result in Failure.
* *(List of Whitelisted Maven Plugins displayed - extensive list)*
* **Scope of Implementation:** Java Maven Builds.
* **Type of Implementation:** Hard Gate.

**7.5. Dependency Validator (Dx-Validator)**

* Service used to check effective pom against missing or unwanted dependencies. dx-service-dependencyrule.yml acts as rule.
* For more info, see Dx-Validator section (Section 8 below).
* **Scope of Implementation:** Java Maven Builds - SpringBoot.
* **Type of Implementation:** Soft Gate.

**7.6. CDQR Gates**

* CDQR (Continuous Data Quality Review) errors refer to any deviations in CSI (CSI System Inventory) application level data quality standards.
* Enhanced to enforce soft gates against CDQR errors. If the CSI ID associated with your pipeline is part of Group 1 Application and has any CDQR errors aging over 45 days, the pipeline will display warning in Tekton task logs.
* **Scope of Implementation:** Java Maven Builds - SpringBoot, NPM Builds, Python build, Java Gradle Build.
* **Type of Implementation:** Hard Gate.

**7.7. GEMs Gates**

* GEMS refers to any vulnerabilities in third party libraries used by the application.
* Enhanced to enforce hard gates against GEMS findings. Pipelines will be blocked if GEMS vulnerabilities of any severity are detected.
* **Scope of Implementation:** Java Maven Builds - SpringBoot.
* **Type of Implementation:** Hard Gate.

**8. Dx-Validator (Detailed)**

**8.1. Overview**  
Dependency validator is used to check effective pom for missing or unwanted dependencies. It is exposed over dx-service (e.g. effective-pom endpoint) (relative to the service context path).

**8.2. Code and Endpoints**

* **Code:** https://[bitbucket-url]/bitbucket/projects/CCP/repos/cecdp-dx-service/browse
* ValidatorController.java contains logic.
* **Manual validation via Route URL:** https://[host]/dx-service-gen-[env].[domain]/api/v1/validateEffectivePom
* dx-service-dependencyrule.yml can be got from URL, content will be dynamic based on latest version.
* **Request (example):**

curl -X POST --data-binary @[effectivePomFile.xml] -H "Content-Type: application/xml" https://[host]/dx-service-gen-[env].[domain]/api/v1/validateEffectivePom

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Bash

* **Response (example snippet for a warning):**
* <artifact>
* <groupId>org.springframework.boot</groupId>
* <artifactId>spring-boot-starter-web</artifactId>
* <version>2.2.5.RELEASE</version>
* <!-- status: WARNING -->
* <!-- message: Status WARNING indicates that if version match is not found then an WARNING is reported. -->

</artifact>

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Xml

* **dx-service-dependencyrule.yml example snippet:**
* - org.springframework.boot:
* - spring-boot-starter:
* versions:
* ['2.3.0.RELEASE']
* status: WARNING

message: Status WARNING indicates that if version match is not found then an WARNING is reported.

content\_copydownload

Use code [with caution](https://support.google.com/legal/answer/13505487" \t "_blank).Yaml

**8.3. How it Works**  
First call is made to Black Duck API to fetch relevant jar version and at every release or spring updates. The service then use dependency-rules.yml and CI-CD Team integrated with their branches and effective POM is validated. Jenkins logs example provided.

* **What happens at ERROR mode:** Build will be switched to error mode, builds will fail if failure scenario is found.
* **Enablement by CI-CD:** CI-CD has the ability to run this feature at CSI level in Jenkins. So only at the best of application team this feature will be enabled.
* **Further Functionalities:** Link provided.

**9. Pipeline Hard Gates**

**9.1. Overview**  
In order that PBWMIT maintains a well-managed applications, the CI/CD pipeline will be continuously enhanced to implement Hard Gates to block any deviations prior to deployment.

**9.2. Specific Hard Gates**

* **SonarQube Hard Gates:** Block when Blocker Vulnerabilities or Blocker Bugs are detected in all code base.
* **CDQR Hard Gates:** Enhanced on March 8th to enforce hard gates against CDQR errors. If the CSI ID associated with your pipeline is part of Group 1 Application and has any CDQR errors aging over 45 days, the pipeline will be CI-Build stage blocked.
* **GEMS Hard Gates:** Enhanced on December 8th to enforce hard gates against Critical GEMS findings. Pipelines will be blocked if GEMS vulnerabilities of Critical Severity are detected. (Refer to GEMS Vulnerability page).
* **Snyk Hard Gates:** Block pipelines when Critical Vulnerabilities are detected that are upgradable or patchable.

**9.3. LightSpeed Enterprise Adoption Hard Gates**  
Strategic pipeline solution. Ensure compliance to LSE policies and best practices. Existing Unify CI/CD Pipeline used by PBWM was enhanced on December 8th to enforce LSE eligible applications adhere to usage of LSE pipelines.

**9.4. Hard Gates Exception Process**  
To formally track exceptions for hard gates, the pipeline will automatically create issues in JIRA.

**9.5. Upcoming Hard Gates (Roadmap)**  
| Hard Gate | Target Date |  
| :---------------------------------------------------------------------------------------------------- | :---------- |  
| Pipeline WARN if any Critical Vulnerabilities that are upgradable or patchable detected by Snyk | 15 Sep 2023 |  
| FAIL CI Pipeline if any Critical Vulnerabilities that are upgradable or patchable detected by Snyk | 30 Nov 2023 |  
| Block Pull Requests with Blocker Vulnerabilities or Blocker Bugs detected by SonarQube | TBD |

**9.6. Gates Roadmap (Table - partial example)**  
| Type | Source | Gate | Target Date | Status |  
| :--- | :----------- | :------------------------------------------------------------------------------------------------------- | :---------- | :-------- |  
| Soft | CSI / CaaS | Pipeline WARN if any Blocker Vulnerabilities or Blocker Bugs detected by SonarQube in respective component's New Code Changes | 15 Nov 2021 | DONE |  
| Hard | SonarQube | FAIL CI Pipeline if any Blocker Bugs detected by SonarQube in respective component's New Code Changes | 15 Mar 2022 | DONE |  
| ... | ... | ... | ... | ... |  
| Hard | Snyk | FAIL CI Pipeline if any Critical Vulnerabilities that are upgradable or patchable detected by Snyk | 30 Nov 2023 | TO DO |  
| Soft | SonarQube | Bitbucket Pull Request WARN if any Blocker Vulnerabilities or Blocker Bugs detected by SonarQube in changed code | 31 Mar 2023 | TENTATIVE |  
| Hard | SonarQube | FAIL Bitbucket Pull Request if any Blocker Vulnerabilities or Blocker Bugs detected by SonarQube in changed code | 30 Jul 2023 | BACKLOG |

**9.7. FAQ**

* **My build is blocked, what do I do?**
  + Sonar findings: review in Sonar dashboard.
  + CDQR error: review CDQR report.
  + Snyk findings: review Snyk dashboard.
  + GEM findings: review BlackDuck dashboard.
  + LSE Pipeline: make sure to adopt LightSpeed Enterprise Pipelines.
  + In case of an exception: look for Jira Key and proceed to secure an exception approval.
* **Where do I see my build results if any soft/hard gates failed?**
  + Build results are available in Jenkins logs.
* **[LightSpeed] Where do I see my pipeline results and if any hard gates failed?**
  + Pipeline results are available in tekton task logs. Logs will also contain Jira key generated for exception process.
  + "reliability\_rating":false means one or more blocker bugs are present.
  + "security\_rating":false means one or more blocker vulnerabilities are present.
* **[LightSpeed] How to get sonar project URL?**
  + SonarQube result URL is available in tekton task logs. (Example log lines provided).
* **Where do I check my sonar results?**
  + Sonar scan results are available in SonarQube dashboard.
  + To view Blocker Bugs or Blocker Vulnerabilities in "New Code": Goto SonarQube dashboard -> Select project -> Select branch -> Click on "New Code" tab (under MEASURES).
  + To view Blocker Bugs or Blocker Vulnerabilities in "All Code": Goto SonarQube dashboard -> Select project -> Select branch -> Click on "Overall Code" tab (under MEASURES). Filter for "Blocker" severity issues.

**10. GEMS (Global Event Management)**

**10.1. GEMS Vulnerability**

* GEMS refers to any vulnerabilities in third party libraries. CI/CD pipelines leverage BlackDuck and an in-house developed scanner.
* Process enhanced on December 8th to enforce hard gates against Critical GEMS findings.
* **For Details on GEMS, follow Confluence: GEMS Quality Gates.**
* Build logs show "GEM Vulnerability Report" JSON snippet and "[ERROR] Found Critical GEMs..." message.
* Jira task created for Exception Approval details (JSON snippet shown).

**10.1.1. Remediation Steps**

1. **Remediate the GEMS Vulnerability:**
   * To be done by Developer/App Team.
   * Create a Dependency Tree: dependency:tree -Dverbose (Maven).
   * Remediate GEMS by updating the GEMS with appropriate versions in pom.xml file.
2. **Get Exception Approval for the Jira Task created:**
   * Hard-Gate Exception Request Process
   * Hard-Gate Exception Approval Process
   * Hard-Gate Exception Approvers
   * *Only after following one of the above two mentioned steps, Build will PASS.*

**10.2. GEMS Quality Gates**

* GEMS Controls - Applications using the OCI Pipelines LSE and Unify with an application GEM are Hard Gated via Gate.
* **Note: Jira exception is no longer supported for GEM (refer to long-tail GEMS process).**

**10.2.1. Seeing GEMS in Build Dependencies**

* Found in build console, under "GEM Vulnerability Report" section.
* Example report: {"criticalGems":1,"unspecifiedGems":0,"majorGems":0,"[SEVERITY]":"CRITICAL","component":"Apache Commons Collections","version":"3.2","policy":"SBT\_CRITICAL\_VULNERABILITIES\_ID-20"}

**10.2.2. Resolving GEM Warnings**

* Use latest versions of impacted dependencies and plugins.

**10.2.3. Finding GEM Origins**

* Build log "GEM Vulnerability Report".
* Use maven-dependency-plugin:
  + https://maven.apache.org/plugins/maven-dependency-plugin/
  + dependency:tree displays the dependency tree.
  + dependency:resolve-plugins tells Maven to resolve plugins and their dependencies.

**10.2.4. GEM Information We Are Currently Scanning For (Table - partial example)**  
| Sl | Policies (GEM Name) | GEM Level | Current Version to Use | Hard-gating Start Date | Libraries with this GEM (Example) |  
| :- | :------------------ | :-------- | :--------------------- | :--------------------- | :-------------------------------------------------------------- |  
| 1 | Apache\_Log4j | L0 | 2.17.1 | Jan 27, 2023 | log4j-core |  
| 2 | Spring\_Framework | L0/L1 | Various | Jan 27, 2023 | spring-aop, org.springframework:spring-beans, spring-context... |  
| 3 | Spring\_Boot | L0/L1,L2 | | Feb 10, 2023 | spring-boot-starter-validation, spring-boot-starter-log4j2... |  
| ...| ... | ... | ... | ... | ... |  
| 25 | H2\_Database\_Engine| L2 | | Mar 30, 2023 | h2, h2-mvstore |

**10.3. GEMS (Global Event Management) - Main Page**

**10.3.1. Overview**

* Welcome to the GEM home page. Security or cyber debt refers to the accumulation of vulnerabilities.
* **Who We Are:** The Cyber Debt team provides strategic direction to prevent the introduction of vulnerabilities in CITI's environment.
* **PBWM Key Stakeholders:** Application Teams, Regional Safety and Soundness Leads, SPOCs, System Admins, CISO, Global Patching Team, CTI.
* **GEM Operating Model:** PBWM GEM Governance (Product Manager, SMSs, etc.)

**10.3.2. Process Flow (Table - high level)**  
| Step | Description | Roles | Artifacts (Example) |  
| :--- | :--------------------------------------------------- | :-------------------- | :----------------------------------------------------- |  
| 1 | Identification and classification of cyber vulnerability | Enterprise InfoSec | |  
| 2 | Launch development of solution to remediate vulnerability | Enterprise InfoSec | |  
| 3 | Activation of the GEMS | Enterprise InfoSec | (Screenshot of activation announcement) |  
| 4 | Creation of GEMS backlog items | Governance Team | |  
| 5 | Development of remediation plan | Safety & Soundness, App Dev | |  
| 6 | Implementation of the solution | App Dev, Prod Mgmt | |  
| 7 | Governance meetings | Governance Team | |  
| | *Diagram: GEM Data Sources -> Data Staging -> Planning & Remediation -> Reporting/Data Analysis -> Escalation* | | |

**10.3.3. Roadmap (GEM UPLIFT - TIMELINE)**

* Shows activities from Q2 2022 through 2023/Ongoing across various themes like Organizational Changes, Process Publishing, Escalation Forums, Data Retention, etc.

**10.3.4. Backlog**

* Maintained in two locations, JIRA and Excel tracker in SharePoint.

**10.3.5. RACI Chart (Example Row)**  
| Project Activity / Deliverable | SSM Team/ Info SEC | Patching Team | SAs/ CTI/ DevOps | CIO SPOC/ Safety & Soundness | App managers | GEMS Governance Team |  
| :--------------------------------- | :----------------- | :------------ | :--------------- | :--------------------------- | :----------- | :------------------- |  
| Publish Vulnerability assessments Feed file | I | R,A | R | I,C | I,A,C | I |  
| Patching the Vulnerability | I | R,A | R | I,C | I,A,C | I |  
*(R=Responsible, A=Accountable, C=Consulted, I=Informed)*

**10.3.6. Customer Process (Jira Dashboard, User Story Updates)**

* **JIRA - How to login:** Link provided, use SSO.
* **What is it?:** Jira dashboard represents user stories for each GEM instance.
* **Manage Dashboard:** Steps to add GEMS NAM DASHBOARD as a favorite.
* **Rich Filters:** Available for viewing issues by category.
* **How to Update a User Story:** Each GEM instance is an individual user story with a detailed subtask.
  + **New Entry:** Edit Subtask, update patching date, change status to Planned.
  + **Planned:** If target date changes, edit and update, change status to Plan Replan.
  + **Replan:** If Replan date is on Target but has been retargeted, update, change status to Replan\_Issue.
  + **Past Due:** SPOC/ App Dev to work with production management to align new date.
  + **Issue:** CIO SPOC, App Manager, GEMS team and Patching teams to work together.
  + **Remediation: No Action**

**11. Hard-Gate Exception Processes**

**11.1. Hard-Gate Exception Request Process (Jira Workflow)**

* Existing Jira exception process would be replaced by OneApproval based process.

1. Open the Jira ticket created for the issue and click on the "Request Exception Approval" button.  
   *(Screenshot of Jira issue with "Request Exception Appr..." button highlighted)*
2. Once requested, Exception Approval Status would be changed to "APPROVAL REQUESTED".  
   *(Screenshot of Jira issue with "Exception Approval Status: APPROVAL REQUESTED" highlighted)*
3. Only approvers can see button to approve exception and grant timebound exception.  
   *(Screenshot of Jira issue showing "Exception Approval" button for approvers)*
4. Once approved, Exception Approval Status will be changed to "DECISION MADE" and Approval Decision, along with Exception Expiry Date will be displayed.  
   *(Screenshot of Jira issue with "Exception Approval Status: DECISION MADE", "Exception Approval Decision: Approved", "Exception Approval Expiry Date" highlighted)*

**11.2. Hard-Gate Exception Approval Process (Detailed Steps)**  
Guidelines & Process for temporary exceptions of CI/CD Hard Gates using Gatr.

* **First Exception:**
  1. App Manager/Delegate Login to JIRA and Search for JIRA issue.
  2. Ensure the App Manager/Delegate has access to the project.
  3. App Manager or delegate should click on "Request for Exception" field in the JIRA issue.
  4. Provide a brief description of your reason for requesting an exception.
  5. Provide remediation plan & mitigation action with target date (Target date not to exceed 30 days).
  6. Upload app manager's email approval in the JIRA issue.
  7. Upload Technology Head's (CIS) email approval in the JIRA issue.
  8. Upload PBIO's email approval in the JIRA issue.
  9. **Note:** We can only provide exception for **30 days**.
* **Second Exception:** Similar steps, App team should click on the JIRA issue link, put JIRA issue in 'Plan' Status, provide reason.
* **Third Exception:** Similar steps, provide a CAP/Project ID with the target date for remediation.
* **Exception Approval in Jira:**
  1. The actual risk is being accepted by the Technology App manager and PBIO. The S&S approval is to ensure proper governance on this process.
  2. **Please note: Jira exception is no longer supported for GEM (refer to long-tail GEMS process).**
* **How to move to Plan status?:**
  1. Click on "Request for Exception" field in JIRA. *(Screenshot shown)*
  2. Click on Edit. *(Screenshot shown)*
  3. Provide Target Date (Date should align with agreed exception date) & click on Update button. *(Screenshot shown)*
  4. JIRA issue will change to planned status. *(Screenshot shown with "Plan Status: Remediation Action Plan Requested" highlighted)*

**11.3. Hard-Gate Exception Approvers (Table)**  
| Rule Type | NAM S&S Primary & Wealth (all) | APAC S&S Primary & Wealth (all) | MEX S&S Primary |  
| :----------- | :----------------------------- | :------------------------------ | :---------------------------------- |  
| SonarQube | John Newcomer Balaji Kaliamoorthy | Panday, Vivek1 | Kuri Parra, Ricardo Jimenez Lopez, Fredy Corona, Guillermo Sarabia-Aguilar, Jorge Ernesto |  
| CDQR Errors | Kumar, Rajesh15 (For Error # 65,66,67,68 & 151) For the rest: Brandon Roberts | Panday, Vivek1 | Kuri Parra, Ricardo ... |  
| APH | Maharana, Seemo Dsouz, Stanlin | Panday, Vivek1 | Kuri Parra, Ricardo ... |  
| GEMS (Tier 1 Tier 2 Apps) | **Please note: Jira exception is no longer supported for GEM (refer to long-tail GEMS process)** | | |

**12. Unify Gates Parity (Table - partial example, this table is very extensive)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Pipeline Phase** | **Gate** | **Outcome** | **PBWM** | **Gate Location** | **Status** | **Benefit?** |
| 1 | Onboarding | AppDev community cannot edit Pipeline or BitBucket project / repo or tool configurations – CI71981AS-257 | Block | PBWM | BitBucket Permissions (Enabled through Driver) | Done | For example Better Quality. Cannot override Pipeline Core Config? ... |
| 2 | Onboarding | Pipeline core configs cannot be edited post onboarding | Block | PBWM | LSE Pipeline Factory | Done |  |
| 3 | Onboarding | Valid CSI is used for onboarding – CI71981AS-272 | Block | PBWM | LSE Onboarding API | Done |  |
| 4 | Code Commit | All code commits must have a valid jira key tagged – CI71981AS-280 | Block | PBWM | BitBucket Configuration (Enabled through Driver) | Done |  |
| 5 | Code Commit | All code commits / merge to release branch must have a pull request with min 1 approval | Block | Global | BitBucket Configuration (Enabled through Driver) | Done |  |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 38 | CI | No Blocker Vulnerabilities detected by SonarQube | Block | PBWM | CaaS v2 | Done |  |
| 39 | CI | No Blocker Bugs detected by SonarQube | Block | PBWM | CaaS v2 | Done |  |
| 40 | CI | No CDQR Errors that are past due 45 days | Block | Group 1 CSI | CaaS | Done |  |
| 41 | CI | No GEMS components detected by BlackDuck GEM Scanner | Block | PBWM | CaaS v2 | Done |  |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 55 | CD | Mandatory canary deployment mode for controlled environments | Block | Cloud Platforms - PCF / OSE | Harness Step (Harness Driver) | Done |  |