[Skip to end of metadata](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode" \l "page-metadata-end)

* Created by [Kedar Bhave](%20%20%20%20/display/~BCAD%0a" \o "), last modified on [18/Apr/18 11:32 AM](https://confluence.cms.gov/pages/diffpagesbyversion.action?pageId=91305382&selectedPageVersions=11&selectedPageVersions=12)

[Go to start of metadata](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#page-metadata-start)

* [Pre-requisites](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-Pre-requisites)
  + [Other Pre-requisites](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-OtherPre-requisites)
    - [Spring Cloud Config Server](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-SpringCloudConfigServer)
      * [Config Files naming convention](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-ConfigFilesnamingconvention)
      * [Instructions for Config Files on Filesystem](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-InstructionsforConfigFilesonFilesystem)
      * [Instructions for Config Files on Git](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-InstructionsforConfigFilesonGit)
    - [Sensitive Configuration](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-SensitiveConfiguration)
    - [RabbitMQ](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-RabbitMQ)
      * [Defining Users / Permissions](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-DefiningUsers/Permissions)
    - [Postgresql](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-Postgresql)
      * [Defining Exchanges / Queues](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-DefiningExchanges/Queues)
* [Pecos-Reference-Service](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-Pecos-Reference-Service)
  + [Required Components](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-RequiredComponents)
  + [Building](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-Building)
    - [Using Gradle](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-UsingGradle)
    - [Using Docker](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-UsingDocker)
  + [Connection](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-Connection)
    - [Context](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-Context)
  + [Swagger UI](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-SwaggerUI)
  + [RabbitMQ Config](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-RabbitMQConfig)
  + [Gateway Service Config](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-GatewayServiceConfig)
* [EA Samples](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-EASamples)
* [Starting a New Project](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-StartingaNewProject)
* [Source Control](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-SourceControl)
  + [Git Clients](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-GitClients)
  + [Git basic Tutorials:](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-GitbasicTutorials:)
  + [Git Documentation on Confluence:](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-GitDocumentationonConfluence:)
  + [Ground Rules for Developers on PECOS 2:](https://confluence.cms.gov/display/PECOSII/Developer+%28Backend%29+Onboarding?src=contextnavpagetreemode#Developer(Backend)Onboarding-GroundRulesforDevelopersonPECOS2:)

Pre-requisites

A few things to get started:

* Java 8 (JDK, not just JRE)
* Gradle 4.6
* Spring Tools Suite (STS) - Eclipse based IDE with Gradle plugin / IntelliJ : Your choice!
* Git
* Docker / Docker toolbox
* VirtualBox
* Postgresql (local or Docker image)
* Cassandra (local or Docker image)

Other Pre-requisites

**Spring Cloud Config Server**

Clone this from <https://github.cms.gov/CPI-PECOS-Platform/pecos-config-server>

You can use Config files on Filesystem or from Github. For local development, you might want to use filesystem if you want to change the configs often.

**Config Files naming convention**

<serviceId>-<Env>.yml

serviceId is defined in Microservice's bootstrap.yml

spring:  
  application:  
    name: <serviceId>

**Instructions for Config Files on Filesystem**

In Config Server bootstrap.xml

spring:  
  cloud:  
    config:  
      server:  
        native:  
          searchLocations: [file:/Users/kedarbhave/Work/ConfigRepo/](http://file/Users/kedarbhave/Work/ConfigRepo/)  
          add-label-locations: false

  profiles:  
    active:  
    - native  
SPRING\_PROFILES\_ACTIVE: native # IMPORTANT if you want to use Native config

**Instructions for Config Files on Git**

You will need your Git token for this.

In Config Server bootstrap.yml

cloud:  
  config:  
    label: ${BRANCH:master}  
    server:  
      git:  
        uri: <https://github.cms.gov/CPI-PECOS-Platform/pecos-config-server.git>  
        search-paths: ConfigRepo  
        username: ${GIT\_USER}  
        password: ${GIT\_ACCESS\_TOKEN}

**Sensitive Configuration**

Sensitive Configuration such as Passwords, keys, etc is kept in Openshift Secrets which makes it available to the container as Environment variables. Developers can define these environment variables on their local machine to simulate that (until such time that they can install/configure Minishift in their dev environment.

**RabbitMQ**

Install locally or Docker Image using docker import.

Follow Mac Instructions: <https://www.rabbitmq.com/install-standalone-mac.html>

Follow Windows Instructions: <https://www.rabbitmq.com/install-windows.html>

**Defining Users / Permissions**

<Scripts will be available soon. Till then it is manual process.>

**Postgresql**

Stopping Postgresql on Mac OS X using the Terminal :

sudo su <postgres user : 'postgres' by default>

pg\_ctl stop -W -D <Data Dir> -m fast

**Defining Exchanges / Queues**

<Scripts will be available soon. Till then it will be done in code.>

Pecos-Reference-Service

Clone this service from <https://github.cms.gov/CPI-PECOS-Dev/pecos-reference-service>

Required Components

* Cassandra
* Config Server
* RabbitMQ
* Gateway Service

Building

**Using Gradle**

gradle clean build

**Using Docker**

From inside the Microservice directory,

docker build . --build-arg service\_port=4444 -t application

Connection

In application.yml,

server:  
 port: ${PORT:4444}

**Context**

gov.hhs.cms.pecos.appService.controller.AppController

@RequestMapping("/application")

Swagger UI

http://localhost:<port>/swagger-ui.html

Swagger UI let's you interact with the service using the API endpoints as documented there.

RabbitMQ Config

In resources/application.yml,

rabbitmq:  
 port: 5672  
 host: localhost  
 username: guest  
 password: guest

Gateway Service Config

In resources/application.yml,

gateway:  
 url: http://localhost:8090/npiverification

EA Samples

Repository: <https://github.cms.gov/CPI-PECOS-Dev/pecos-ea-samples>

Sample code for various technology options explored by EA.

Starting a New Project

Use the Spring Initializr scaffolding tool. This can be used on a Linux based system from the command line or from its UI.

Spring Initializr UI: <https://start.spring.io/>

Command Line:

curl <https://start.spring.io/starter.zip> -d dependencies=web,actuator,hateoas,cloud-config-client -d type=gradle-project -d language=java -d javaVersion=1.8 -d bootVersion=2.0.0.M7 -d packaging=jar -d baseDir=<Project Dir> -d groupId=gov.hhs.cms.pecos -d applicationName=<Project Name> -d name=<Project Name> -d artifactId=<Project Name> -d description="<My Awesome Service>" -o <Project Name>.zip

Unzio the project thus created / downloaded, and open it in your IDE.

Source Control

Git Clients

Mac OS:

* SourceTree
* GitX
* Eclipse
* IntelliJ

Windows

* SourceTree

Git basic Tutorials:

* Yes, a Hello World Tutorial for Git too! - <https://guides.github.com/activities/hello-world/>

Git Documentation on Confluence:

Ground Rules for Developers on PECOS 2:

* Developers do not create Repositories
* Developers do not create Branches
* Developers do not give access

[Skip to end of metadata](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#page-metadata-end)

* Created by [Praveen Palety](%20%20%20%20/display/~PY9I%0a), last modified by [Richard Hutchinson](%20%20%20%20/display/~HWR9%0a) on [11/Dec/18 12:37 PM](https://confluence.cms.gov/pages/diffpagesbyversion.action?pageId=85266238&selectedPageVersions=26&selectedPageVersions=27)

[Go to start of metadata](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#page-metadata-start)

* [Introduction](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Introduction)
  + [12-factor Application Methodology](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-12-factorApplicationMethodology)
* [Domain Model and Microservices Architecture](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-DomainModelandMicroservicesArchitecture)
* [Microservices Best Practices and Patterns](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-MicroservicesBestPracticesandPatterns)
  + [Full Stack](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-FullStack)
  + [Hypermedia-Driven Orchestration](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Hypermedia-DrivenOrchestration)
  + [OpenAPI 3.0](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-OpenAPI3.0)
  + [Service Endpoints](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ServiceEndpoints)
  + [Resource Naming](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ResourceNaming)
  + [Versioning](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Versioning)
  + [Healthcheck](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Healthcheck)
  + [Logging](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Logging)
    - [Correlation Id](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-CorrelationId)
  + [Messaging Standards](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-MessagingStandards)
* [Programming Standards](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ProgrammingStandards)
  + [Code Style](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-CodeStyle)
  + [Configuration](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Configuration)
    - [Spring Cloud Config Server for Externalized Configuration (For non-secure configs)](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-SpringCloudConfigServerforExternalizedConfiguration(Fornon-secureconfigs))
      * [Config Server Location](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ConfigServerLocation)
      * [Config Files Naming](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ConfigFilesNaming)
      * [Configuration Files](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ConfigurationFiles)
    - [Encryption/Decryption Based Security](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Encryption/DecryptionBasedSecurity)
      * [Spring Cloud Config](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-SpringCloudConfig)
      * [Openshift Secrets (Preferred)](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-OpenshiftSecrets(Preferred))
      * [Security using Vault](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-SecurityusingVault)
  + [Java Package Structure](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-JavaPackageStructure)
  + [Naming Conventions](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-NamingConventions)
  + [Coding to Interfaces Technique](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-CodingtoInterfacesTechnique)
  + [Error Handling and Exceptions](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ErrorHandlingandExceptions)
    - [Error Codes and Messages:](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ErrorCodesandMessages:)
    - [Exceptions Handling:](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-ExceptionsHandling:)
  + [Unit Testing](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-UnitTesting)
  + [Localization](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Localization)
* [Documentation](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Documentation)
  + [JavaDoc documentation](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-JavaDocdocumentation)
  + [General Comments](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-GeneralComments)
  + [API documentation](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-APIdocumentation)
* [PECOS Reference Microservice and EA-Samples](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-PECOSReferenceMicroserviceandEA-Samples)
  + [Prerequisites:](https://confluence.cms.gov/display/PECOSII/Coding+Standards?src=contextnavpagetreemode#CodingStandards-Prerequisites:)

Introduction

12-factor Application Methodology

The twelve-factor app is a methodology for building software-as-a-service apps that –

* Use **declarative** formats for setup automation, to minimize time and cost for new developers joining the project;
* Have a **clean contract** with the underlying operating system, offering **maximum portability** between execution environments.
* Are suitable for **deployment** on modern **cloud platforms**, obviating the need for servers and systems administration.
* **Minimize divergence** between development and production, enabling **continuous deployment** for maximum agility.
* And can **scale up** without significant changes to tooling, architecture, or development practices.

For more information, please refer to [http://12factor.net](http://12factor.net/)

Domain Model and Microservices Architecture

While designing Microservices using the DDD methodology, the focus is on organizing the code by aligning it to business problems and using the same ubiquitous language. In DDD, each Bounded Context becomes a Microservice. A microservice should not simply be designed around the CRUD operations for an entity. But boundaries defining related functions should be used to define them.

Microservices Best Practices and Patterns

Microservices is an Architectural style that structures an application as a set of loosely coupled services which implement business capabilities. Each service implements a set of related functions. In PECOS-2, this could mean an Application Microservice, an Case Management Microservice, an Identity and Access Management Microservice, a Reporting Microservice and so on.

* Small, focused scope, Independent, and Encapsulated (bounded context)
* If deploying a microservice requires simultaneous and time sensitive updates to multiple other services, then something is wrong.
* Data Sources must be private to a Microservice
* Microservices must be versioned
* Language agnostic protocols such as REST/HTTP, AMQP/MQTT instead of JMS, JSON are strongly recommended

A service must be small enough to be developed by a small team and to be easily tested. A useful guideline from object-oriented design (OOD) is the Single Responsibility Principle ([SRP](http://www.objectmentor.com/resources/articles/srp.pdf)). The SRP defines a responsibility of a class as a reason to change, and states that a class should only have one reason to change. It make sense to apply the SRP to service design as well and design services that are cohesive and implement a small set of strongly related functions.

The application also be decomposed in a way so that most new and changed requirements only affect a single service. That is because changes that affect multiple services requires coordination across multiple teams, which slows down development. Another useful principle from OOD is the Common Closure Principle (CCP), which states that classes that change for the same reason should be in the same package. Perhaps, for instance, two classes implement different aspects of the same business rule. The goal is that when that business rule changes developers, only need to change code in a small number - ideally only one - of packages. This kind of thinking makes sense when designing services since it will help ensure that each change should impact only one service.1

1. Microservices.io

Full Stack

Every Microservice has the entire stack – Database to UI. The persistence layer is a Polyglot architecture with a mix of SQL and No-SQL Technologies.

Hypermedia-Driven Orchestration

API should not just return static results, but it should also return controls or possible actions at that point. This let's us orchestrate loosely coupled micro services for complex workflows.

This is achieved by including Self-linking URLs in response. We will use Spring HATEOS library for doing this.

More details: [Restful API Documentation and Structure Standards](https://confluence.cms.gov/display/PECOSII/Restful+API+Documentation+and+Structure+Standards)

OpenAPI 3.0

Microservices shall follow OpenAPI 3.0 standard to document service contracts.

Springfox Swagger plugin is used to achieve this.

http://<api-url>/swagger-ui.html

http://<api-url>/swagger.json

More details: [Restful API Documentation and Structure Standards](https://confluence.cms.gov/display/PECOSII/Restful+API+Documentation+and+Structure+Standards)

Service Endpoints

For variations to an endpoint, multiple endpoints will be created rather than adding a lot of confusing parameters to the same endpoint.

This can be achieved by using appropriate HTTP Verbs, new endpoint identifiers and url structure.

HTTP Verbs:

* GET: Retrieve a resource
* POST: Create a resource
* PUT: Update or create within an existing resource
* PATCH: Partially modify an existing resource
* DELETE: Remove the resource

Resource Naming

Resource names and corresponding URLs shall use plural nouns such as /dispositions or /applications

Versioning

Every microservice should have a major API version in the URL, such as:

http://<api-url>/v1

This version info should be carried to the logs for each Microservice-version combination. PECOS 2 Microservices will either follow Semantic Versioning with MAJOR, MINOR and PATCH version info or just a MAJOR version in API URLs.

Multiple concurrent versions shall be maintained (for a short period of time) to support older clients.

Healthcheck

* Every microservice should have a healthcheck endpoint that reports on the end-to-end health status for that microservice.
* More (Healthcheck Specification)

Logging

Use a centralized logging service that aggregates logs from each service instance. (CloudWatch on AWS?)

Rules to be followed while Logging:

* Include API Version Info
* Consistent Log Format
* When you receive an initial request for processing create and pass a Correlation Id

**Correlation Id**

A Correlation Id should uniquely identify an entity/request in the system. Create a correlation ID for each incoming request in every service so that this correlation ID can identify the context. In a web application, this is the front controller that first handles incoming HTTP requests. In case of Microservices, the idea is simply this -

When a user-facing service receives a request it creates a correlation ID, and:

* passes it along in the 'X-Correlation-Id' or 'X-Request-Id' HTTP header to every other service
* includes it in every log message

For Synchronous processing, one could simply set this id in the ThreadLocal construct.

Messaging Standards

[RabbitMQ Developer Documentation](https://confluence.cms.gov/display/PECOSII/RabbitMQ+Developer+Documentation)

[Messaging Naming Guidelines](https://confluence.cms.gov/display/PECOSII/Messaging+Naming+Standards)

Programming Standards

Code Style

Using checkstyle with google\_checks.xml. Run checkstyle using the gradle task.

For IDE integration, you can use the specific formatter from the Git repository.

Import Code template from the Git repository (eclipse-codetemplates.xml) to include a standard Copyright text into your Java files.

Configuration

YAML based configuration. Environment specific and general config.

* Cloud Config Server for general config with environment specific files stored in filesystem or git.
* Openshift Secrets (with Environment variables for Development environment) for secure configurations such as passwords etc.
* Routing configuration in Kong (mapping between routes and endpoints)
* No configuration will be in dockerfile / docker-compose

**Spring Cloud Config Server for Externalized Configuration (For non-secure configs)**

Cloud Config Server is a Spring Boot Application that allows you to externalize configuration for Microservices.

In PECOS, all non-secure configuration will be stored in Config server.

**Config Server Location**

config server is assumed to run on port 8888. To modify this behavior, you change the config server location in bootstrap.yml as follows:

spring.cloud.config.uri: <http://myconfigserver.com>

**Config Files Naming**

The environment specific resources are parametrized by three variables:

* {application} maps to "[spring.application.name](http://spring.application.name)" on the client side;
* {profile} maps to "spring.profiles.active" on the client (comma separated list); and
* {label} which is a server side feature labelling a "versioned" set of config files.

For example, appService-dev.yml (without {label})

**Configuration Files**

By default, the configuration files for remote clients are pulled from a git repository :

spring:

  cloud:

    config:

      server:

        git:

          uri: <https://github.com/spring-cloud-samples/config-repo>

The other way is to use File based configuration.

spring:

  cloud:

    config:

      server:

        native:

          searchLocations: file:/<Dir>/

Most online documentation for Config Server recommends the use of Git in Production environment mainly because it offers versioning, branching and such. However, it may not be desirable or feasible to have access to a Git repository from our Production environment, in which case, we should use File based configuration option in PECOS.

**Encryption/Decryption Based Security**

For more secure configuration settings, there are 3 options:

**Spring Cloud Config**

Does support Encryption / Decryption of Configuration using JCE. It supports both Symmetric and Asymmetric Cryptography, the latter being more secure to use in Production environments.

This allows one to use Config server for managing secure config items such as Passwords.

For Asymmetric you need to create a keystore and configure it in application.yml.

**Openshift Secrets (Preferred)**

Configuration such as passwords will be stored in Openshift Secrets and will be presented to the Microservice application as environment variables.

**Security using Vault**

One could use the Config server along with Vault for keeping sensitive configuration data such as passwords.

Java Package Structure

Base package : gov.hhs.cms.pecos

main

java

[Base package]

assembler (for HATEOS)

resource (for HATEOS)

controller

model

repo

resources

application.yml

bootstrap.yml

Naming Conventions

* All package names should be lowercase
* All class names should start with an uppercase letter
* Variable names starting with lowercase letter and should be meaningful. Variable names such as ‘i’, ‘j’ should not be used
* All enums should be uppercase

Coding to Interfaces Technique

The technique of coding to interfaces involves using interfaces, rather than objects, as the primary method of communication. Coding to interfaces is a technique by which developers can expose certain methods of an object to other objects in the system.  The developers should write code that does not interact directly with an object as such, but rather with the implementation of that object's interface.

Error Handling and Exceptions

**Error Codes and Messages:**

* Each Microservice shall have its own configured range of Error codes.

**Exceptions Handling:**

* Deciding between checked and unchecked exceptions :
  + Use Unchecked if Client code can’t do anything about it
  + Use Checked if Client code can recover from the exception
  + As far as possible, avoid throwing generic Exception class instance.
  + Do NOT suppress exceptions at the lower tiers. A SQLException at the Data Access Layer should not be caught and suppressed. It should not go to the Business layer as is either. Convert that and throw a RuntimeException with meaningful message to the Business layer above it.
  + Use of finally and clean up
  + Log exceptions only once

Unit Testing

Tools: Junit, Mockito, AssertJ

Guidelines for testing will be covered under QA standards

Localization

Use of Resource Bundles is recommended even if we have no plans to localize. The bundles should be named as per standard Java practices.

Guidelines for frontend will be covered under UI/UX standards

Documentation

JavaDoc documentation

Every class, every method should have the JavaDoc style /\*\* \*/ comment block, clearly describing the Purpose, [Author, Version] (for Class), [Input, Output] (for methods).

General Comments

All comments except ones appearing on the same line as valid Java code should be multi-line comments of the form –

/\* .. comment .. \*/

API documentation

OpenAPI 3.0.0 using Swagger

* In ApiInfo as defined in the Application class, make sure to have the correct Contact information in case there is a need to contact someone for support.
* Every Controller method should have the @ApiOperation annotation explaining what the method does in short as the “value”
* Document valid responses for each using @ApiResponses annotation

PECOS Reference Microservice and EA-Samples

These repositories provide standard reference implementations and Samples for various technology solutions used in PECOS-2. These can be cloned and used as starting points by new Dev Teams coming on board.

Reference Microservice: <https://github.cms.gov/CPI-PECOS-Dev/pecos-reference-service>

EA-Samples: <https://github.cms.gov/CPI-PECOS-Dev/pecos-ea-samples>

**Prerequisites:**

* Java 8
* Gradle
* Spring Tools Suite (STS) - Eclipse based IDE / IntelliJ
* Git
* Docker / Docker toolbox
* VirtualBox
* Postgresql (local or Docker image)
* Cassandra (local or Docker image)

[Skip to end of metadata](https://confluence.cms.gov/display/COMS/Selenium+Bites+%28cheat+sheet%29+for+block+of+codes?src=contextnavpagetreemode#page-metadata-end)

* Created by [Johnson Hiwale](%20%20%20%20/display/~HCWX%0a) on [31/May/18 9:51 AM](https://confluence.cms.gov/pages/viewpreviousversions.action?pageId=99764655)

[Go to start of metadata](https://confluence.cms.gov/display/COMS/Selenium+Bites+%28cheat+sheet%29+for+block+of+codes?src=contextnavpagetreemode#page-metadata-start)

**1-Capturing full page screenshot, with vertical scrollbar-**

Using shutterbug jar full page screen capture can be done,It is capable of handling vertical, horizontal scrollbar effectively and captures a full screenshot.

String timeStamp = new SimpleDateFormat("HH\_mm\_ss").format(new Date());  
String screenshot\_timestamp = null  ;  
screen\_timestamp = screenshotname+"\_"+timeStamp;  
Shutterbug.shootPage(driver,ScrollStrategy.VERTICALLY).withName(screen\_timestamp).save("Locl drive location");

**2-Linking screenshot in TestNG reports-**

To link captured a screenshot to testNg report below line of code can be called, this code link screenshot with click to open become hyperlink which opens captured screenshot.

Reporter.log("<a href="+"E:\\screenshot\\"+screenshot\_timestamp+".png"+">click to open

**3-Method to handle Sync Issues in web automation-**

This is the best way to handle sync issues, by using a for loop with an upper limit of time

public WebElement Element\_Finder(WebDriver driver, By locator, int T) throws InterruptedException  
{  
WebElement myElement = null;  
//driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(0));  
for (int i = 1; i < T; i++)  
{  
try  
{  
System.out.println("Iteration -->"+ i);  
myElement = driver.findElement(locator);  
//This method is written in #4  
WebElementHighlight(myElement, driver);  
break;  
}  
catch (Exception ex)  
{  
Thread.sleep(1000);  
System.out.println("Did not found element with locator"+locator);  
}  
}

**4-Highlighting a web object with some colour-**

Selenium driver will highlight the web object, we only need to call below method and same will highlight the web object in return

public void WebElementHighlight(WebElement element, WebDriver driver)  
{  
if (driver instanceof JavascriptExecutor) {  
((JavascriptExecutor)driver).executeScript("arguments[0].style.border='6px groove green'", element);  
}  
// return element;  
}

**5-Whether a webelement exist or not-**

To identify whether a web element exists or not, below code is an effective way to identify the same.  
In below code size of the webobject is being checked, where not equal to zero condition is being checked.

driver.findElements([By.id](http://By.id)("element-id")).size()!=0

**6-Working on frame-**

While working on frames, first of all, identify the count of a frame and then use switch to method to switch over respective frame.

int i;  
i=driver.findElements(By.xpath("//iframe")).size();  
driver.switchTo().frame(i-1);

**7-How to declare webdriver and use the same in another class-**

Best way to declare a web driver in a class and then to get that driver.

Find below sample code

public class Superclass  
{  
public WebDriver driver;  
public Superclass(){  
driver = new FirefoxDriver();  
}  
public WebDriver getdriver(){  
if (driver == null){  
driver = new FirefoxDriver();  
return driver;  
}else{  
return driver;  
}  
}  
}

In other class, call like below syntax  
getdriver().findelement(by.xpath("xpath").click());

Instantiate Specific Browser Or Client

You may need this code in every selenium script. So I thought of keeping it here handy for the popular ones.  
  
**8-Firefox Driver**

WebDriver driver = new FirefoxDriver();

**9-Chrome Driver**

WebDriver driver = new ChromeDriver();

**10-Safari Driver**

WebDriver driver = new SafariDriver();

**11-Internet Explorer Driver**  
WebDriver driver = new InternetExplorerDriver();

**12-Android Driver**

WebDriver driver = new AndroidDriver()

**13-iPhone Driver**

WebDriver driver = new IPhoneDriver();

**14-HTML Unit**

WebDriver driver = new HtmlUnitDriver()

**15-Check If An Element Exists**

You may need to perform a action based on a specific web element being present on the web page. You can use below code snippet to check if a element with id “element-id” exists on web page.

driver.findElements([By.id](http://By.id)("element-id")).size()!=0

**16-How To Check If An Element Is Visible With WebDriver**

The above mentioned method may be good to check if a elemet exists on page. However sometimes a element may be not visible, therefore you can not perform any action on it. You can check whether an element is visible or not using below code.

WebElement element  = driver.findElement([By.id](http://By.id)("element-id"));

if(element instanceof RenderedWebElement) {

System.out.println("Element visible");

} else {

System.out.println("Element Not visible");

}

**17-Refresh Page**

This may be required often. Just a simple refresh of the page equivalent to a browser refresh.

driver.navigate().refresh();

**18-Navigate Back And Forward On The Browse**r

Navigating the history of borwser can be easily done using below two methods. The names are self explanatory.

**19-Go back to the last visited page**

driver.navigate().back();

**20-Go forward to the next page**

driver.navigate().forward();

**21-Wait For Element To Be Available**

The application may load some elements late and your script needs to stop for the element to be available for next action. You can perform this check using below code.  
In below code, the script is going to wait maximum 30 seconds for the element to be available. Feel free to change the maximum number per your application needs.

WebDriverWait wait = new WebDriverWait(driver, 30);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable([By.id](http://By.id)("id123")));

**22-Focus On A Input Element On Page**

Doing focus on any element can be easily done by clicking the mouse on the required element. However when you are using selenium you may need to use this workaround instead of mouse click you can send some empty keys to a element you want to focus.

WebElement element  = driver.findElement([By.id](http://By.id)("element-id"));

//Send empty message to element for setting focus on it.

element.sendKeys("");

**23-Overwrite Current Input Value On Page**

The sendKeys method on WebElement class will append the value to existing value of element. If you want to clear the old value. You can use clear() method.

WebElement element = driver.findElement([By.id](http://By.id)("element-id"));

element.clear();

element.sendKeys("new input value");

**24-Mouseover Action To Make Element Visible Then Click**

When you are dealing with highly interactive multi layered menu on a page you may find this useful. In this scenario, an element is not visible unless you click on the menu bar. So below code snippet will accomplish two steps of opening a menu and selecting a menu item easily.

Actions actions = new Actions(driver);

WebElement menuElement = driver.findElement([By.id](http://By.id)("menu-element-id"));

actions.moveToElement(menuElement).moveToElement(driver.findElement(By.xpath("xpath-of-menu-item-element"))).click();

**25-Extract CSS Attribute Of An Element**

This can be really helpful for getting any CSS property of a web element.  
For example, to get background color of an element use below snippet

String bgcolor = driver.findElement([By.id](http://By.id)("id123")).getCssValue("background-color");

**26-And to get text color of an element use below snippet**

String textColor = driver.findElement([By.id](http://By.id)("id123")).getCssValue("color");

**27-Find All Links On The Page**

A simple way to extract all links from a web page.

List link = driver.findElements(By.tagName("a"));

**28-Take A Screenshot On The Page**

The most useful one. Selenium can capture the screenshot of any error you want to record. You may want to add this code in your exception handling logic.

File snapshot =((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

**29-Execute A JavaScript Statement On Page**

If you love JavaScript, you are going to love this. This simple JavascriptExecutor can run any javascript code snippet on browser during your testing. In case you are not able to find a way to do something using web driver, you can do that using JS easily.  
Below code snippet demonstrates how you can run a alert statement on the page you are testing.

JavascriptExecutor jsx = (JavascriptExecutor) driver;

jsx.executeScript("alert('hi')");

**30-Upload A File On A Page**

Uploading a file is a common use case. As of now there is not webdriver way to do this, however this can be easily done with the help of JavascriptExecutor and little bit of JS code.

String filePath = "path\\to\\file\for\\upload";

JavascriptExecutor jsx = (JavascriptExecutor) driver;

jsx.executeScript("document.getElementById('fileName').value='" + filePath + "';");

**31-Scroll Up, Down Or Anywhere On A Page**

Scrolling on any web page is required almost always. You may use below snippets to do scrolling in any direction you need.

JavascriptExecutor jsx = (JavascriptExecutor) driver;

//Vertical scroll - down by 100 pixels

jsx.executeScript("window.scrollBy(0,100)", "");

//Vertical scroll - up by 55 pixels (note the number is minus 55)

jsx.executeScript("window.scrollBy(0,-55)", "");

//Horizontal scroll - right by 20 pixels

jsx.executeScript("window.scrollBy(20,0)", "");

//Horizontal scroll - left by 95 pixels (note the number is minus 95)

jsx.executeScript("window.scrollBy(-95,0)", "");

**32-Get HTML Source Of A Element On Page**

If you want to extract the HTML source of any element, you can do this by some simple Javascript code.

JavascriptExecutor jsx = (JavascriptExecutor) driver;

String elementId = "element-id";

String html =(String) jsx.executeScript("return document.getElementById('" + elementId + "').innerHTML;");

**33-Switch Between Frames In Java Using Webdriver**

Multiple iframes are very common in recent web applications. You can have your webdriver script switch between different iframes easily by below code sample

WebElement frameElement = driver.findElement([By.id](http://By.id)("id-of-frame"));

driver.switchTo().frame(frameElement);

**34-Capturing full page screenshot, with vertical scrollbar-**

Using shutterbug jar full page screen capture can be done,It is capable of handling vertical, horizontal scrollbar effectively and captures a full screenshot.

String timeStamp = new SimpleDateFormat("HH\_mm\_ss").format(new Date());  
String screenshot\_timestamp = null  ;  
screen\_timestamp = screenshotname+"\_"+timeStamp;  
Shutterbug.shootPage(driver,ScrollStrategy.VERTICALLY).withName(screen\_timestamp).save("Locl drive location");

**35-Linking screenshot in TestNG reports-**

To link captured a screenshot to testNg report below line of code can be called, this code link screenshot with click to open become hyperlink which opens captured screenshot.

Reporter.log("<a href="+"E:\\screenshot\\"+screenshot\_timestamp+".png"+">click to open

**36-Method to handle Sync Issues in web automation-**

This is the best way to handle sync issues, by using a for loop with an upper limit of time

public WebElement Element\_Finder(WebDriver driver, By locator, int T) throws InterruptedException  
{  
WebElement myElement = null;  
//driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(0));  
for (int i = 1; i < T; i++)  
{  
try  
{  
System.out.println("Iteration -->"+ i);  
myElement = driver.findElement(locator);  
//This method is written in #4  
WebElementHighlight(myElement, driver);  
break;  
}  
catch (Exception ex)  
{  
Thread.sleep(1000);  
System.out.println("Did not found element with locator"+locator);  
}  
}

**37--Highlighting a web object with some colour-**

Selenium driver will highlight the web object, we only need to call below method and same will highlight the web object in return

public void WebElementHighlight(WebElement element, WebDriver driver)  
{  
if (driver instanceof JavascriptExecutor) {  
((JavascriptExecutor)driver).executeScript("arguments[0].style.border='6px groove green'", element);  
}  
// return element;  
}

**38-Whether a webelement exist or not-**

To identify whether a web element exists or not, below code is an effective way to identify the same.  
In below code size of the webobject is being checked, where not equal to zero condition is being checked.

driver.findElements([By.id](http://By.id)("element-id")).size()!=0

**39-Working on frame-**

While working on frames, first of all, identify the count of a frame and then use switch to method to switch over respective frame.

int i;  
i=driver.findElements(By.xpath("//iframe")).size();  
driver.switchTo().frame(i-1);

**40-How to declare webdriver and use the same in another class-**

Best way to declare a web driver in a class and then to get that driver.

Find below sample code

public class Superclass  
{  
public WebDriver driver;  
public Superclass(){  
driver = new FirefoxDriver();  
}  
public WebDriver getdriver(){  
if (driver == null){  
driver = new FirefoxDriver();  
return driver;  
}else{  
return driver;  
}  
}  
}

//In other class, call like below syntax  
getdriver().findelement(by.xpath("xpath").click());