GovInfo Components and Technologies

A. Overview

This document is intended to provide a high-level overview and understanding of the GovInfo System architecture. It should be referenced in conjunction with the GovInfo Architecture System Design Document, volume I, particularly Figure 4.1-1 Component and Development Diagram.

There are two main GovInfo subsystems that handle separate areas of major concern. These subsystems have significant interactions to support the ingest, preservation, and public access functions of the system.

1. Content Management and Preservation Subsystem
2. Public Access Subsystem

Both subsystems rely on the same underlying compute and storage infrastructure that are managed by GovInfo system administrators and GPO's enterprise infrastructure support team.

This document also includes a list of technologies currently in use on the GovInfo program to provide a better understanding of the context of the system.

B. Component Descriptions

1. Content Management and Preservation Subsystem

The Content Management and Preservation Subsystem is responsible for the intake of new Government documents and publications into the system, including a variety of checks, processing, and preservation functions to support the needs of GPO to maintain its ISO 16363 Trustworthy Digital Repository status. GPO is committed to ensuring that official Government publications from all three branches are preserved for future generations in the event of technology failure, aging of hardware, or technological change. It serves as the primary management and storage area for content and metadata that are published for public access by the Public Access Subsystem. The following sections describe the major functionality of the Content Management and Preservation Subsystem components including the closely related Content Processing Framework.

Documentum

Documentum is the commercial off the shelf (COTS) software that serves as the content management system and preservation repository for GovInfo content and metadata. It has been configured to support GPO's implementation of the Open Archival Information System reference model (OAIS, ISO 14721) including but not limited to aspects of ingest, data management, and archival storage, preservation planning, administration, and access functions. GovInfo's Documentum implementation currently enables user interaction via WebTop. It also works with several monitoring jobs to ingest content from a variety of sources (e.g. SFTP, hot folders for internal content, Congress.gov API, and Salesforce). Documentum interfaces with the publisher component within the Public Access Subsystem to make content available to the public.

Databases

The GovInfo Documentum repository currently operates on an Oracle database. The database schemas are defined, and tables populated, by Documentum applications. The database maintenance is therefore not about managing schemas, tables, et cetera, unlike standalone or custom database applications. For GovInfo, the database maintenance must be performed in the context of the Documentum repository. Thorough understanding of the interactions between Documentum applications with the underlying database is critical to manage the database properly and effectively. Repository performance issues may find root causes in the underlying database; proactive monitoring, careful tuning of the database, and quarterly security patching, is the key to maintain a healthy repository. Database availability and uptime is also a key consideration; daily backup, clustered operation and synchronization between nodes (RAC, Oracle Clustered database configured on NetApp iSCSI Luns) /sites (Oracle Data Guard is implemented between primary and secondary production sites to ensure data protection and disaster recovery) are critical to daily operations, and knowledge of Oracle-specific hardware and software constraints and setup are necessary for continued service availability. Note, the GovInfo Drupal content management system for content curation activities and Matomo for analytics metrics both operate on MySQL databases and require similar considerations. All the Oracle and MySQL database servers run on VMware virtual machines (VM).

Processor

The GovInfo processor interacts with the Documentum system and other Content Management and Preservation Subsystem components to perform several important workflow steps as part of the GovInfo ingest process. This includes parsing content files to generate granules, combine files, generating PREMIS preservation and custom GovInfo descriptive/technical metadata, creating derived public-access renditions of content and metadata, signing/optimizing/OCRing/certifying of PDF content, validating packages, and other tasks.

Other Applications

OAIS Packaging

This functionality takes files provided from content originators and organizes them in a model that fits the ISO 14721 Open Archival Information System model, as implemented by the GovInfo program.

Preprocessing and Hotfolders

This functionality provides for any preprocessing jobs that are required prior to processing within Documentum. Examples include interfacing with the Salesforce API for the Congressionally Mandated Reports collection, PDF file validation, and Submission Information Package generation.

Internal and external users upload files or packages to hotfolders through SFTP services. Services consumes these files through direct mount and upload.

Virus Scanning

The virus scanning component is used as a service to validate that content files being ingested into the Content Management and Preservation Subsystem are free of viruses and malware. This is currently implemented via a COTS Malware protection tool via API call.

PII Scanning

The PII scanning component is implemented as a service to identify content being ingested that may contain PII. This check is requested via an API call to software provided by a COTS PII scanning tool and returned to the ingest processing workflow.

PDF Signing and Optimization

This functionality integrates an Adobe AEM service backed by a hardware security module that allows for PDF signing and other functionality.

Publication Linking

This component provides the ability to link multiple GovInfo content packages based on certain metadata fields and defined relationships. It uses Apache Jena and Fuseki with a custom OWL ontology. As an example, Publication Linking provides linkages between various versions of a bill text, or from a bill text to the equivalent public law, and various other relationships among documents. For more information, see https://www.govinfo.gov/related-documents.

Metadata Editors

GovInfo uses customized metadata editors based on Orbeon Forms to allow editing of descriptive and preservation metadata associated with GovInfo content. This is integrated with the Documentum component to allow internal authorized users to build content packages, make edits to existing published packages, and update preservation-related metadata for content archived in the system.

USLM Web Services

The USLM Web Services component is used to perform conversion of submitted content files into United States Legislative Markup-compliant XML. For reference see https://github.com/usgpo/uslm.

Schemas

GovInfo uses custom XML schemas to manage and persist descriptive and technical metadata for the content ingested and preserved within the system. Schemas are built, updated, and extended to fit the needs of new and existing content.

Content Processing Framework

The Content Processing Framework is a set of standalone Java applications that allows internal users to execute several jobs thru batch processing on content outside the system prior to submission on an ad hoc basis, including validation of certain file characteristics using tools like FITS or Jpylyzer. Additionally, it can perform conversions of TIFF files to JP2 files and perform MARC XML conversion as well as PDF signing for other Federal agencies by having their respective FTP site linked to CPF hot folders.

Analytics

GovInfo uses an on-premise installation of the open-source Matomo web analytics platform to help track overall usage of the public GovInfo system through integration of access logs as well as a JavaScript tracker. Data is anonymized and aggregated to help track retrievals of GovInfo content and metadata as well as inform the type of devices and browsers used to access the system. GovInfo also uses an on-premise installation of Splunk for log aggregation for analytics research and log monitoring.

2. Public Access Subsystem

The Public Access Subsystem is responsible for providing public access to copies of Government documents and publications that have been ingested into GovInfo. The following sections describe the major functionality of the Public Access Subsystem components.

Search Applications

These applications are key interfaces between the Public Access Subsystem and the Content Management and Preservation Subsystem.

Solr Search Engine

GovInfo's open-source Solr search engine implementation is the heart of the Public Access Subsystem, providing the indexed content and metadata used by other Public Access components to generate the public user interfaces, including the various browse and search use cases as well as API access for the developer community.

A custom Solr Search Engine core includes support for custom fields, dynamic field definitions, custom fieldType definitions, as well as custom analyzers, tokenizers and filters.

Additionally, custom plug-ins are implemented to provide support for Lemmatizers, Query Digesters, Custom query operations, and custom Tokenization

The Solr Search Engine installation is designed for both high-availability and scalability. There are two rows of Solr servers, with one row dedicated as the primary indexer. Indexed content is automatically replicated to the second row. A software load balancer automatically distributes the query load across the Solr servers. Scalability is achieved by adding one (or more) servers to each row.

Publisher

The publisher component interacts with the Content Management and Preservation Subsystem's Documentum component to make processed content and metadata publicly available, allowing indexing by the search engine and sending information to the Notification App to update RSS and Sitemaps.

The publisher automates pulling new and updated content from the Content Management and Preservation Subsystem. It uses XSL to transform that content into Solr documents with custom metadata. XSL transforms also used to generate public preservation (PREMIS XML) and descriptive (MODS XML) metadata files.

The publisher uses Apache Tika to extract text from binary content. That text can be used to populate indexed metadata as well as the main body and teaser.

The publisher uses JSoup to support automated crawling and indexing of curated content.

The publisher also includes an API interface for maintenance of remotely hosted video content.

Consistency Check

This custom Java component is used to perform regular validation of consistency between content available to the public in the search engine and the Documentum system.

Parsers

This custom java component is used to extract metadata from submitted files during ingest processing, which are then propagated into a variety of different forms within both subsystems. The parsers integrate with the Processor component of the Content Management and Preservation Subsystem.

Search API

The Search API component uses dynamic groovy templates and custom solr interface to provide decorated results. These include navigators and hierarchical navigators, hierarchical browse tree navigation, custom drop-down data population, and custom search results displays including computed fields and teasers.

The Search API implements custom search fields that simplify common and complex Solr queries.

The Search API implements a custom software load balancer to distribute the load across all available Solr servers. The load balancing also implements automated error detection and blacklisting.

The Search API implements custom query mapping to convert common search phrases into specific Solr query syntax.

Query Parser Language

This is a custom layer to convert Solr queries into an efficient Lucene syntax. This layer implements custom query operators (before, between, boost, starts_with, ends_with, near, phrase, scope, etc.). The Query Parser Language supports search within XML blobs.

Web Applications and Services

These Services are responsible for providing public access to GovInfo content and metadata.

GovInfo UI Web Services

GovInfo web services are the core of the GovInfo website. They provide interfaces to the backend services that powers most of the functionality of the website. These web services are separated into multiple submodules to support GovInfo functionality for end users, including Related Documents, Content Details, Browse, and Search functionality. The web services interfaces with three forms of storage - Search indices, RDF store, and file system metadata files - to perform most of the processing and present data to the front-end application in JSON format.

The web service offloads most of the heavy-duty processing like doing XSL transformation on the metadata, Zip file creation, and caching the file to another application by creating an event job file before delegating it to the target Content Delivery and Caching Services application. The web service uses the delegate pattern to ensure it is loosely coupled with the services layer. It also uses commonly used REST standards to handle requests from the web front end.

Bulk Data Repository

GPO makes select GovInfo collections available in a machine-readable format (i.e., XML) via the GovInfo Bulk Data Repository. The top-level directory for select collections, such as the Federal Register and Code of Federal Regulations, also includes a Resources directory that contains the XML schema, XSL stylesheet, and user guide necessary to make the XML to be transformed to HTML for the browsers to present in a human-readable format. The Bulk Data repository provides access this data through XSLT to make a human- and machine-readable interface.

Public Functionality: https://www.govinfo.gov/bulkdata

Redirect Dispatcher

This component is designed to support the users who had bookmarked URLs or links of the predecessor FDsys application before its retirement. Dispatcher is a drop-in replacement for the legacy FDsys application that receives all the traffic that FDsys was supposed to receive and redirects the requests to equivalent GovInfo pages. Dispatcher handles all the links that FDsys used to handle, not just the content links but also search, browse, details, and document in context links.

CGI Redirects

This component provides backwards compatibility for multiple legacy patterns of content links to access documents available on GovInfo. This component supports users who have historical links referencing GPO Access, a legacy application and precursor to FDsys and GovInfo. These historical links appear on some government websites as well as in print. GPO Access was retired in 2011. When FDsys took over its functionality, redirects were put in place at that time. During the development of GovInfo, GPO determined that there was an ongoing need to maintain these legacy links. This interface continues to honor legacy GPO Access content retrieval patterns using the latest GovInfo infrastructure.

Sample link handled by CGI redirects: http://frwebgate.access.gpo.gov/cgi-bin/getdoc.cgi?dbname=110_cong_public_laws&docid=f:publ085.110

Link Service

The link service application provides a standard and predictable way to reference content or metadata that is available on GovInfo. These references often take the form of embedded links in third-party content or services. Published content on GovInfo is uniquely identified using an "accessid" metadata field. The same value manifests itself as packageid and granuleid if the content is package or granule, respectively. In addition, the accessid is a Solr attribute for a document to uniquely identify it. The Link Service provides a way to identify content by using terms and identifiers that are germane to select GovInfo content, such as a combination of Congress, bill type, and bill number.

The Link service adheres to the Open API 3 specification and the user interface is a customization of the Swagger-ui.

Public Functionality: https://www.govinfo.gov/link-docs

Public API

The GovInfo public API is the main means to programmatically expose GovInfo content and metadata to third party developers and other business partners or Federal agencies. Anyone who is interested in consuming these APIs should enroll and have an api.data.gov key. The GovInfo Public API endpoints are diverse and far-reaching in their functionality and usefulness, as the design is heavily influenced by the input of the stakeholders. The API endpoints follow REST and HATEOS paradigms, so a consumer of the API can directly follow up to related resource links. These APIs differ from the GovInfo UI Web Services in that they are intended to serve as a stable, contracted interface for external developers rather than the specific evolving needs of the GovInfo user interface. The Public API adheres to the Open API 3 specification and the user interface is a customization of the Swagger-ui.

Public Functionality: https://api.govinfo.gov/docs

See also:

- Developer Hub: https://www.govinfo.gov/developers
- GitHub Repository: https://www.github.com/usgpo/api
- GovInfo Feature Articles:
    - Overview
    - Related Documents Service
    - Search Service

Content Curation (Drupal)

In addition to GovInfo content and metadata, GPO staff author and publish "curated content" that serves as finding aids and resources for users. This content includes feature articles, help pages, custom landing pages, and other content. This includes functionality to upload and manage media files for use within articles. This functionality is implemented via a custom implementation of the Drupal Content Management System using custom modules and Twig templates that integrates with the look, feel, and functionality of the GovInfo custom applications, including the search widget available on every page. Drupal sign-in uses Two- factor authentication using Authenticated AES encryption method for Encrypt through the Defuse PHP Encryption library and is seamlessly integrated with enterprise LDAP for authenticating the users. Drupal is accessed only internally by authorized users, with custom static publishing of curation content and media to the public. The custom site crawlers crawl the site and publish the content to an intermediate (Preview) static site which will then be published to the public site after review. The static crawled pages and sections are seamlessly integrated and embedded with the dynamic sections of the site using micro front end architecture. Content curation also triggers events for the crawled static pages to be consumed by the publisher to index the pages in Solr for curation search. Drupal is backed by a MySQL database that stores content and metadata.

Content Curation Search

The content curation search component provides the web service interface and React-based front-end for the public curation search functionality for about, feature articles, and help pages. This integrates the curation functionality provided by Drupal with the GovInfo Solr search engine, where the curated content has its own distinct collection, separate from GovInfo's official publications.

Public Functionality: https://www.govinfo.gov/app/search/curated/API/

Notification App

The Notification App component is a custom Java application that receives events from the Publisher component to update RSS feeds and Sitemaps based on new or updated public content. The component creates or updates the sitemap index files and the sitemap yearly file for every collection of the packages received in the event files. After the process is done, the same event is forwarded to the RSS event queue to generate the RSS feed. The robots.txt file deployed at the root of the application containing an entry for GovInfo and Bulk Data sitemap index files for every collection. Heavy use of xml parsing is done using SAX, StAX and JAXB parsers. Sitemap consistency check is performed from time to time using several custom jobs which provide custom reports for any inconsistencies in the sitemaps and RSS feeds. External crawlers start by requesting and parsing the index files. Then for each sitemap element, the last modified date can be compared against the internal saved value and the sitemaps are crawled and indexed accordingly depending on the last seen modified value.

Caching Services

The Caching Services component asynchronously processes and streams GovInfo collection resources like XML, CSS, XSL, DTD, image files, metadata, and content files to end users. This multi-threaded application component receives the input data requests in the form of events for different granules for complex collections. When users access them via the website or Public API for metadata and Zip file content, the component checks for their existence in the file system cache, processes them, and caches them in the file system before passing to the Delivery service, which streams requested files to the end users.

Delivery Service

The Delivery Service is a lightweight spring-boot-based java application that maps the sanitized input from user request for content and metadata to the backend Md5 hash distributed file system on the GovInfo filers. The final computed paths are then handed over to the web servers which delivers the file to the end client by adhering to the request headers and uses fast optimization and caching techniques

The delivery service uses md5 hashing to identify the paths of the content files and its resources in ACP-cache and dmz-cache. The delivery service also interacts with the caching services to create on demand event files for large workloads such as ZIP and large granule mods file. The event file is later consumed by the caching services application and the content is generated and finally delivered back to the client on a retry-after basis.

The Delivery Service services content and metadata URL requests for the entire Public Access subsystem, supporting the following components:

- GovInfo UI Web Services
- Bulk Data Repository
- Public API
- Redirect Dispatcher
- CGI Redirects
- Caching Services

Pre-Render Service

The Pre-Render Service uses an Open-source prerender server running on nodejs technology and expressjs in a load balanced configuration on on-premise servers for partner crawlers. There are customizations to the underlying source for adding log files, timeout customizations, and there are customizations for reporting metrics to

Prometheus (pull). This allows certain data partners to crawl prerendered HTML for the content details pages, enabling non-Javascript aware tools to collection information.

Due to these customizations, it requires hands on expertise on prerender.io codebase. It is also configured to be in load balanced environment with 2 instances. There is a Jenkins job associated to monitor the health of the configuration and send notifications daily

Apache Server Virtual Domains and Customization

GovInfo's Public Access subsystem relies heavily on Apache Web Servers to respond to requests from end users. Configurations for these servers are templated using standard processes, allowing for consistent functionality and performance across environments.

UI Stylesheets and Style Guide

GovInfo's CSS files are compiled from source Sass SCSS files that are collected into a single project to maintain a single source of design truth. Bootstrap, FontAwesome, and React Calendar SCSS files are compiled as part of the SCSS build.

Gulp and Gulp plugins are used to transpile and build the source SCSS files into CSS. Gulp is also used to build the style guide and CSS documentation. Autoprefixer adds browser and vendor prefixes to styles, where needed, at build time. PostCSS is also used for adding additional prefixes. Stylelint is used to check for errors and enforce conventions.

Style and UI Documentation

Style and UI documentation is maintained in three sets: via in-file comments, a style guide built using FractalJS, and the UI component library workshop tool StorybookJS.

Visual Regression Testing

Visual design continuity and consistency is checked using visual regression testing by comparing reference and test screen shots for differences. Visual regression testing is automated using BackstopJS.

C. Technologies

The following is a list of the standards and technologies that are currently in use within the GovInfo program.

1. Content Management and Preservation Subsystem

- Documentum Content Server, Documentum Composer, Documentum Administrator, Process Engine, Forms Builder, Media Workspace / Digital Asset Manager, Process Builder, Process Integrator Core, Trusted Content Services Core
    - Documentum Content Server deployed on Linux

    - Documentum WebTop Custom Web Application deployed on Apache Tomcat
    - Documentum xPlore and Interactive Delivery Service deployed on Linux
- Various descriptive, administrative, structural, preservation metadata schemas, data dictionaries, and technical registry, implementation tools, including but not limited to METS, MODS, PREMIS, PRONOM
- Various file format identification tools, including but not limited to Droid, PRONOM, JHOVE, FITS (File Information Tool Set), Exiftool
- Various data schemas structures, transforms, and formats including JSON, XML, XSLT, XSL, XSD
    - USLM schema
    - Custom internal GovInfo XML schema
- Data interchange models and frameworks, web ontologies such as RDF, OWL, SPARQL, Apache Fuseki SPARQL Server, Apache Jena RDF Store
- Custom Applications in Java EE
- Data analytics tools such as Matomo
- COTS PII scanning tools with OCR on embedded images
- ImageMagick Converter, Redact-it, Jpylyzer, FITS, VeraPDF
- Adobe standard technology for digital signing, including Adobe Experience Manager Forms, PDF Generator, Adobe Acrobat, Hardware Security Module
- iText, iText Pdf Optimizer, iText PdfOCR, Tesseract Engine
- HSQLDB used to Manage Jobs for Content Processing Framework
- Custom Web Application using Orbeon XForms server
- Standard COTS AV and AV API interface
- Active Directory, LDAP compliant application directories, identity management
- Common open-source application servers (e.g., Wildfly, Tomcat)
- Application Gateway & high availability using Apache Webserver mod_proxy_balancer
- API usage including api.data.gov
- Salesforce Rest API
- Salesforce JWT authentication
- Git, GitLab EE, Azure DevOps (ADO)
- Common and standard open-source software build automation tools and IDEs including Jenkins, Maven, Eclipse, and Nexus Artifact Repository
- Common open-source administration automation tools including Ansible
- Common open-source products for analyzing data in real time (e.g., Splunk, Zabbix, Prometheus, Grafana)
- Red Hat Enterprise Linux

- MySQL Database
- Oracle Database used by Documentum including DataGuard
- Oracle Enterprise Manager Cloud Control

2. Public Access Subsystem Technologies

- JSON, XML, XSLT, XSL, XPath, XSD
- XML parsing and Processing using Jakarta API; JAXP, JAXB, Xalan XSLT Parser, SAX, STAX2 APIs, XOM, and XStream
- Spring MVC, Spring boot, Restful WS, SpringDoc for OpenAPI-3.x, Templating engine Thymeleaf, Micrometer and Servlet API & Custom XSS filters
- Embedded Servlet Container Tomcat and Jetty
- Monitoring Service using Prometheus, Grafana, Alert Manager, Node Exporter, Solr Exporter and Custom QPS exporter
- JavaScript, TypeScript HTML, AJAX, nodejs, Prerender, Swagger-ui-react
- HTML, CSS, SASS including media queries
- Open-source libraries: httpclient, slf4j, gson, gauva, Logback, micrometer, sitemapgen4j, OpenCSV, Tika, SolrJ, JSoup
- Automatic testing using JUnit, Mockito, rest-assured, Cucumber, Selenium, Jest and Cypress
- Performance testing using: JMeter, Custom Model using production log files, Grafana and timeseries database InfluxDB
- PHP, Xampp, MySQL Database, Toad Edge for MySQL
- UI architecture: Micro-frontend, Single Page Application and Responsive UI
- Application Architecture: Multi-layer Architecture, Micro-service and Event-Driven architecture
- Scripted deployment using Ansible, Jinja2 template and Bash
- CI/CD using GitLab, Jenkins, Webhook, Nexus, Maven, npm, vite and Azure DevOps (ADO)
- Jenkins Pipeline as code and multi-branch pipelines.
- Custom Applications in Java EE
- Common and standard open-source software build automation tools and IDEs. including Jenkins, Maven, Eclipse, and Nexus Artifact Repository
- Common open-source administration automation tools including Ansible
- Common open-source products for analyzing data in real time (e.g., Splunk, Zabbix, Prometheus, Grafana)
- MODS, PREMIS, METS, USLM

- Groovy Dynamic Language, Template Engine and Shell
- Message queue systems using RabbitMQ and String Stream API
- API usage including api.data.gov and api.congress.gov
- Custom Java Content and Metadata Parsers
- Core Java and RegEx
- Custom Query Parser Language (QPL), Lucene, Velocity Template Engine
- Solr deployed on common open-source application servers
- Spring MVC-Based Custom Web Applications
- Custom Spring, Spring MVC, and Spring Boot RESTful Web Services
- High Visibility, High Availability, Public Facing Web Applications using Bootstrap, Backbone.js, React, Redux, JQuery, Vitejs, rollup, webpack, npm packaging and publishing, Enterprise component library (building, versioning and packaging for non-react environments like Drupal and third-party sites), End to End tests using Cypress and Selenium web driver
- Apache webserver Configuration as Code using Ansible, Jinja2 template engine and Python
- Application Gateway & high availability using Apache Webserver mod_proxy_balancer and mod_proxy_hcheck
- Maintenance and c gnu compilation for the custom apache module mod_xsend
- Drupal deployed on Apache Web Server, PHP, Custom Drupal modules written in PHP deployed on Apache Web Server, Twig 2.0 custom templates, TFA authentication, Custom Site crawlers, Composer dependency management for PHP, Drush shell interface for Drupal, Bootstrap custom theme, Custom Configurations for DB interfaces, Custom Sitemap and RSS generators
- Containerization and Orchestration tools like Docker Containers, Docker-Compose, Kubernetes, minikube, podman
- Web metrics, analytics, Matomo
- Linux, RHEL8/9
- SEO, Crawlers, Sitemaps
- CloudFlare including Cloudflare Stream
- Accessible Standards including Sections 508 and 255, WCA, WAI-ARIA
- Active Directory, LDAP 3 Compliant Application Directories, Red Hat Identity Management (IdM)
- Swagger and Open API Documentation