



# **FORUM SYSTEMS SENTRY™ VERSION 9**

## **DEVOPS API PIPELINE PROMOTION GUIDE**



#### **Legal Marks**

No portion of this document may be reproduced or copied in any form, or by any means – graphic, electronic, or mechanical, including photocopying, taping, recording, or information retrieval system – without expressed permission from Forum Systems, Inc.

Copyright © 2002-2023 Forum Systems, Inc. – All Rights Reserved.

Forum Systems Sentry™ Version 9 DevOps API Pipeline Promotion Guide, published July 2023.

D-ASF-SE-314330

## Table of Contents

Introduction .....	4
DevOps Products Utilized.....	4
Sentry Instances .....	4
Sentry REST API .....	5
API Promotion Overview.....	5
Overview .....	5
Promotion Flow.....	5
Promotion Roles.....	6
High Level Steps for API Promotion .....	6
Repos and Jenkins Files.....	7
Overview .....	7
Sentry API Repos and Naming Conventions .....	7
Sentry Configuration Naming Conventions .....	7
Jenkins Files in the Git Repo .....	11
API Promotion Peer Review Template.....	12

## Introduction

This document provides a high level example of a DevOps pipeline for Forum Sentry APIs using 3 common DevOps tools – Git, Jenkins, and Cloudbees CD.

The Sentry APIs are developed in a Sandbox instance and eventually promoted through DEV and QA to PROD instances.

The Sentry APIs are developed using Global Variables that allow for the API to be promoted from development instances to production instances with minimal, if any, need for changing the API before or after promotion. For this reason, it is important that Global Variables are used and clearly documented by the developers when building Sentry APIs.

## DevOps Products Utilized

Any variation of similar tools can be used for the DevOps pipeline. This document describes a standard scenario using the following industry tools:

- **GitHub**
- **Jenkins**
- **Cloudbees CD**

## Sentry Instances

For the pipeline promotion, there are several stages of the pipeline which leverage environment instances to facilitate the workflow promotion of APIs to production. Environment instances include Sandbox, Staging, DEV, QA, and PROD. These are defined below.

ENVIRONMENT	DEFINITION
Sandbox	Developer personal sandbox instances used for building the APIs
Staging	Environment instances used for policy compares and validating promotion process
DEV	Environment instances used for functional testing
QA	Environment instances used for performance and User Acceptance Testing (UAT)
PROD	Environment instances used for processing live production traffic

These instances are often named with a convention that enables dev/ops teams to easily identify the environment which the Sentry API promotion process is currently in.

Sandbox – SBX1, SBX2, etc.

Staging – STG1, STG2, etc.

DEV – DEV1, DEV2, etc.

QA – QA1, QA2, etc.

PROD – PRD1, PRD2, etc.

## Sentry REST API

Forum Sentry provides both a graphical interface for policy management as well as a REST API which enables automated provisioning using REST calls to the API. Importantly for the promotion of policies through the pipeline, the REST API is enabled on each instance within each pipeline environment to allow the DevOps tools to access the REST API for FSG import into the next stage pipeline instances. For more information about enabling the Sentry REST API, please refer to the “FS\_Sentry\_REST API” guide in the documentation set.

## API Promotion Overview

### Overview

APIs are developed in the sandbox instances via the Forum Sentry WebAdmin. API policies are defined with the workflow that the specific API should exhibit. This includes the type of API such as REST, JSON, XML, OpenAPI, etc. Additionally, the API policies define the listener and remote protocols, message types, authentication requirements, security policies, task flow processing on request/response, logging, and auditing. These policies artifacts are then exported as single units called FSGs. The FSGs contain all of the main policy and sub-policy dependencies required to handle the API flows in each environment. FSG are single file exports in secure format which enable these files to be used for export and promotion through the DevOps pipeline. Some of the steps are manual (export / import) and others are automated via the DevOps tools in use - such as Jenkins, CD, and the Sentry REST API.

The use of Global Variables enables environment specific values to be abstracted and the global variable mappings can be applied at each environment in the pipeline promotion process.

Since the APIs will likely have a lifecycle of versioning, APIs are often promoted through the pipeline multiple times. The same procedures apply to the initial promotion as for subsequent promotions, though in some cases manual changes by the Platform Admin (i.e. adding new Global Variable mappings) may be necessary with each promotion.

### Promotion Flow

As noted earlier, the Sentry instances are put into environments that facilitate different aspects of each stage in the pipeline promotion process. This includes a step for storing the API configuration FSG file into a version control Repository such as Git. This step provides a central storage and versioning

mechanism for the pipeline process whereby it can leverage standard dev/ops pipeline workflow utilities such as Jenkins.

The Sentry promotion flow is:

SANDBOX → STAGING → **GIT REPO** → DEV → QA → PROD

### Promotion Roles

In order to ensure a review process is implemented, the following roles should be defined for each stage in the pipeline:

- API Developer
- DEV Peer Reviewers
- DEV Lead
- DEV Platform Admin
- QA Platform Admin
- QA Testing Team
- PROD Platform Admin

### High Level Steps for API Promotion

The following sequence is the high level steps that occur from the initial building of a new API to the final publishing of the API to production (assuming all steps complete successfully).

1. Developer builds API in personal Sandbox
2. Developer exports API from Sandbox for Staging
3. Developer imports API into Staging
4. Developer Peer Review in Staging – review API, Global Variables, XSLTs, etc.
5. Developer exports API (FSG file) from Staging
6. Developer uploads API (FSG file) from Staging to the Git Repo
7. Developer runs Jenkins job to promote the API to DEV (FSG import via REST API)
8. Developer submits Pull Request(PR) in Git
9. Code review approval is applied to the PR by Reviewer
10. Dev Lead reviews Pull Request and merges changes
11. CD Pipeline for the API is created with merge
12. Developer requests promotion to QA via CD pipeline
13. Dev Lead and/or Platform Admin promotes API to QA via CD (FSG import via REST API)
14. Developer requests promotion to PROD via CD pipeline
15. CD Pipeline runs at scheduled date/time promoting the API to Prod (FSG import via REST API)
16. Developers and Testers validate the promotion
17. Consumers are on-boarded

## Repos and Jenkins Files

### Overview

Each API should have its own Git Repo. This allows versioning and documentation to be specific to each API. Each Git Repo should include:

1. A README file
2. Jenkins files
3. The API itself (the FSG file)

### Sentry API Repos and Naming Conventions

The process of developing a new API should start with requesting access to a new Git Repo for the API. When requesting a new Repo, naming convention is important to provide clear way of distinguishing the API and version information. This should be the API and Repo name format:

- API name = exdocvirusscan-v1.0
- Repo name = fs\_exdocvirusscan-v1.0

Other Repo names as examples:

- fs\_simoscensus-v1.0
- fs\_exdocmgmt-v1.0
- fs\_bppvdr-v1.0

As the Git Repos are created from templates, the files within need to be reviewed and customized by the developer prior to any reviews or promotions.

### Sentry Configuration Naming Conventions

This section outlines the naming conventions to be used for:

1. Sentry configuration objects (i.e., Content policies, Network Policies, Keys, Task Lists, etc.)
2. Sentry Labeling
3. Global Variables
4. FSX and FSG Files

Important Notes:

- Note that the Content Policy names are the API names – they use a prefix of the type of Content Policy (i.e. WSDL, XML, REST, etc.)
- Labels are only applied to TLGs, TLs, and Documents (not to Content Policies or Network Policies)
- Use underscores rather than hyphens when possible

- We will use camelCase primarily but not everywhere

## Network Policies

HTTP Listeners = <b>HTTP_Port</b>	e.g., HTTP_8080
HTTPS Listeners = <b>HTTPS_Port</b>	e.g., HTTPS_443
HTTP Remotes = <b>HTTP_R_fqdn-or-IP-of-remote</b>	e.g., HTTP_R_1.2.3.4 or HTTP_R_server1.aamc.org
HTTPS Remotes = <b>HTTPS_R_fqdn-or-IP-of-remote</b>	e.g., HTTPS_R_1.2.3.4 or HTTP_R_server1.aamc.org

## Content Policies (APIs)

WSDL Policies = <b>WSDL_ApiName_OperationName</b>	e.g., WSDL_AmsPullServiceService_Change
XML Policies = <b>XML_ApiName_OperationName</b>	e.g., XML_AmcasDataService
REST Policies = <b>REST_ApiName_OperationName</b>	e.g.; REST_ErasDataService
JSON Policies = <b>JSON_ApiName_OperationName</b>	e.g.; JSON_JsonDataService
HTML Policies = <b>HTML_ApiName_OperationName</b>	e.g.; HTML_HireVue

*Notes: Operation name is optional in some cases may be needed to distinguish from other APIs*

## Virtual Directories

Virtual Directory Names = <b>ApiName_lastSegmentofPath</b>	e.g., XML_AmcasDataService_Actions or XML_AmcasDataService_History
--	---

*Notes – there may be more than 1 virtual directory under a Content Policy and each needs to be named uniquely*

*When the virtual directory is at the root level, use \_Root after the API Name*

## PKI Policies

SSL Termination Policies = <b>ssl_term_ApiSpecific_Auth</b>	e.g., ssl_term_Amcas_Auth or ssl_term_WebAdmin
SSL Initiation Policies = <b>ssl_init_ApiSpecific_Auth</b>	e.g., ssl_init_Ldap or ssl_init_DataCommons
Keys & Certs = keep existing names (these can't be renamed)	

*Notes: As these policies already exist, use something specific in the name that indicates usage*

*Add \_Auth if SSL 2 way auth is enabled, leave it off if SSL 2way is disabled*

## Task Lists



Tasks = **FunctionOfTask** e.g., MatchUserAttributeCode or ConvertJsonToXML

*Notes - Camel case task names, use descriptive task names that are unique within the task list*

Task Lists = **tl\_apiname\_function** e.g., tl\_XML\_AmcasDataService\_SslCertExtraction

*Notes - the function is the purpose of the task list, please also list a description for the task list if known*

Task List Groups = **tlg\_apiname\_location** e.g., tlg\_XML\_AmcasDataService\_RequestProcessing

*Notes – the last segment (location) indicates when the TLG runs (i.e., request processing, response processing, error processing)*

## MISC Policies

Error Templates = **et\_apiname** e.g., et\_XML\_AmcasDataService

Request Filters = **rf\_apiname** e.g., rf\_XML\_AmcasDataService

Documents = **doc\_apiname\_DocumentName** e.g., doc\_XML\_AmcasDataService\_AmcasRequest

Pattern Match API Specific = **pm\_apiname\_function** e.g., pm\_XML\_AmcasDataService\_removecomma

*Notes - the function indicates what the pattern match is used for*

Pattern Match Common = **pm\_function** e.g., pm\_removecomma

*Notes - the function indicates what the pattern match is used for, many pattern match policies may be reused across policies*

Custom IDP Groups = **idp\_group\_apiname** e.g., idp\_group\_XML\_AmcasDataService

Custom IDP Rules = **idp\_rule\_RuleType** e.g., idp\_rule\_MaxDocSize50mb

LDAP Policies = **ldap\_usage** e.g. ldap\_AdminAccess

User Groups = **user\_group\_GroupName** e.g., user\_group\_Appian

User ACLS = **user\_acl\_AclName** e.g., user\_acl\_Appian

Data Source Policies = **ds\_apiname or function** e.g., ds\_XML\_AmcasDataService or ds\_backups

## Sentry Labeling

If the policy type is not listed yet supports labels, do not use a label. Refrain from applying more than 1 label to any given policy object.

Task Lists = **apiname** e.g., XML\_AmcasDataService

Task List Groups = **apiname** e.g., XML\_AmcasDataService

Documents = **apiname** e.g., XML\_AmcasDataService

HTTP Listeners = **NO LABEL**

HTTP Group Listeners = **NO LABEL**

HTTP Remotes = **NO LABEL**

Content Policies = **NO LABEL**

Keys / Certs = **NO LABEL**

## Global Variables

Notes – use all lowercase for Global Variable names, watch for extra spaces in the values

Remote Servers = https_r_api/service_host	e.g., https_r_amcas_host
Remote Path = apiname_path	e.g., xml_amcasdataservice_path
Keys = keypair_api/service_function	e.g., keypair_workday_signature or keypair_aamc_sslterm

- *Use a key pair type variable*

Certs = cert_api/service_function	e.g., cert_workday_encryption or cert_aamc_signergroup
-----------------------------------	---

- *Use a certificate type variable*

Client Secrets = apiname_clientsecret	e.g., xml_amcasdataservice_clientsecret
---------------------------------------	---

- *Use password type variables to mask the data*

Client ID = apiname_clientid	e.g., xml_amcasdataservice_clientid
------------------------------	-------------------------------------

Username = username_api/service	e.g., username_workday
---------------------------------	------------------------

Password = password_api/service	e.g., password_workday
---------------------------------	------------------------

- *Use password type variables to mask the data*

## FSG & FSX Files

FSG Files = <b>apiname_ENV.fsg</b>	e.g., XML_AmcasDataService_DEV.fsg or XML_AmcasDataService_PTEST.fsg
------------------------------------	---

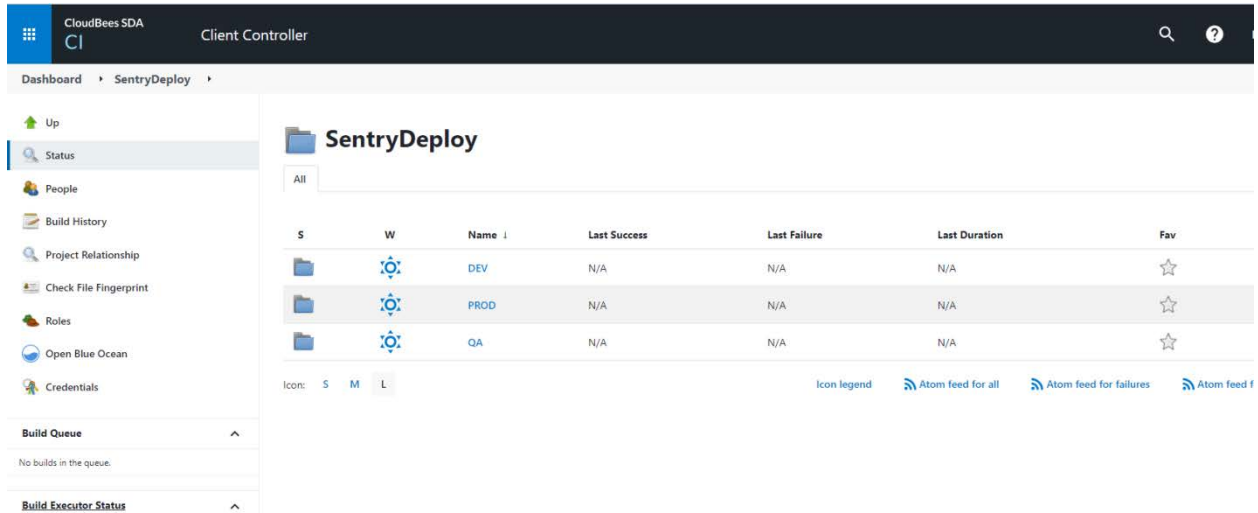
FSX Files = <b>InstanceName_SentryVersion_Date(yyymmdd).fsx</b>	e.g., FTEST1_9.142_220808.fsx or DEV_9.159_221031.fsx
---	--

Password = the password to use with all FSX and FSG files is **[ENTER PASSWORD HERE]**

## Jenkins Files in the Git Repo

The Jenkins files are used for the pipeline promotion. Since there are 3 stages to the promotion pipeline, there will be 3 Jenkins files that are required to be defined in the Repo:

1. Jenkins DEV
2. Jenkins QA
3. Jenkins PROD



Jenkins uses these files to build the scripts needed for the pipeline promotion. Jenkins has its own scripts that perform many tasks such as pulling the data from these files to use for the Sentry import/export process via the Sentry REST API, using the API name and policy type to use in the curl commands for the Sentry REST API calls, etc.

Sample DEV Jenkins File:

```
8 lines (8 sloc) | 206 Bytes

1 @Library('jenkins-shared-library-devops-si') _
2 cd_si_forumsentry(
3     policyname: "exdocvirusscan-v1.0",
4     policytype: "wsdl",
5     environment: "DEV",
6     ie_environment: "IE_DEV",
7     version: "1.0.0"
8 )
```

Developer will access the repo for their API to promote to DEV:

