Table of Contents

Sample 1: Documentation Work Samples

- **CA Process Automation Solution Suite (CA PASS)**
  I have authored the content for the Solution Suite from the scratch:
  http://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-enterprise-software/intelligent-automation/automic-process-automation/04-3-02/solution-suite.html

- **CA Process Automation (CA PA)**
  The documentation space of CA Process Automation includes the legacy content.

  I have authored almost 30% of the documentation space and maintained the entire documentation space of CA Process Automation:
  http://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-enterprise-software/intelligent-automation/automic-process-automation/04-3-02/release-notes.html

- **CA Process Automation Open API (REST API)**
  The documentation covers the REST APIs for CA Process Automation. The API documentation is provided with an interface which is similar to Swagger user interface:
  https://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-enterprise-software/intelligent-automation/automic-process-automation/04-3-02/reference/web-services/ca-process-automation-open-rest-apis.html

- **CA Process Automation Open API (SOAP API)**
  The documentation covers the SOAP APIs for CA Process Automation:
  https://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-enterprise-software/intelligent-automation/automic-process-automation/04-3-02/reference/web-services/soap-api-reference.html

- **DX Application Performance Management 11.1 (DX APM 11.1) On-premise**
  The documentation space of DX APM is provided as an on-premise solution and includes 30% of the legacy content:
  https://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-enterprise-software/it-operations-management/application-performance-management/11-1.html

- **DX Application Performance Management SaaS (DX APM SaaS)**
  The documentation space of DX APM is applicable for the Solution as a Service (SaaS) platform and includes 30% of the legacy content:
  http://techdocs.broadcom.com/content/broadcom/techdocs/us/en/caenterprise-software/it-operations-management/dx-apm-saas/SaaS.html

  On-going updates are made to 70% of the DX APM SaaS documentation space with the latest SaaS features such as Amazon Web Services, Microsoft Azure Services, NGINX, Elasticsearch, Couchbase, and so on.

  Apart from maintaining the complete documentation space of DX APM SaaS, I have authored SaaS related content from the scratch for the cloud platforms such as Amazon Web Services, Microsoft Azure Services, Google Cloud Platform, NGINX Monitoring, Elasticsearch Monitoring, Couchbase Monitoring, MongoDB Server, IIB Monitoring, and so on.

Sample 2: Enhancements in the Existing Documentation

I have performed the following tasks to revamp the existing agent documentation significantly by improving the documentation of the agent extensions of DX APM SaaS.

Performed the following tasks towards improving the quality of the NGINX Monitoring Extension documentation:

1. Restructured the entire section of the following Agent Extensions:
   a. NGINX Monitoring Extension
   b. Oracle EBS Monitoring Extension
   c. Oracle RAC Extension
   d. Oracle Database Extension
   e. Elasticsearch Monitoring
   f. Couchbase Monitoring
   g. MySQL Database Monitoring
   h. Amazon Web Services Monitoring
   i. Azure Monitoring
   j. Apache Flume Monitoring
   k. SQL Server Database Monitoring
2. Discussed with the Product Manager (PM) and arrived at a logical flow for Installation and Configuration section of the following Agent Extensions:
   a. Install and Configure NGINX Monitoring Extension
   b. Install PHP Agent in Docker
   c. Install and Configure Oracle EBS Monitoring Extension.
   d. Install and Configure Oracle RAC Extension
   e. Install and Configure Oracle Database Extension
   f. Install and Configure Elasticsearch Monitoring Extension
   g. Install and Configure Couchbase Monitoring Extension
   h. MySQL Database Monitoring
   i. Install and Use Amazon Web Services Monitoring
   j. Install and Configure Azure Monitoring
   k. Install the Apache Flume Monitoring Extension
   l. Install and Configure SQL Server Monitoring Extension
3. Restructured the content and validated the procedure in the Install and Configure section by performing the testing of procedure of all Agent Extensions that are listed in **Step 2**.
4. While validating the Install and Configure section, **identified the technical gaps in the information that the development team had provided**, and addressed the gaps in the section Install and Configure section of all Agent Extensions that are listed in **Step 2**.

5. Installed the Agent Extension to verify that the metrics and attributes pertaining to a particular Agent Extension is populated in the user interface of DX Application Performance Management.
   I had identified the **technical errors** in the information that the development team had provided. I shared my findings with the respective stakeholders. The gaps that I had identified helped to furnish the required technical content with technical accuracy.

   Validated the description for the metrics and the attributes for the following Agent Extensions:

   a. NGINX Monitoring Metrics
      Verified and validated **54 Metrics** and **7 Attributes** of the NGINX Monitoring Extension.

   b. Oracle EBS Monitoring Metrics
      Verified and validated **54 Metrics** and **7 Attributes** of the Oracle EBS Extension.

   c. Oracle RAC Monitoring Metrics
      Verified and validated **120 Metrics** and **7 Attributes** of the Oracle RAC Extension.

   d. Oracle Database Monitoring Metrics
      Verified and validated **90 Metrics** and **7 Attributes** of the Oracle Database Extension.

   e. Elasticsearch Metrics
      Verified and validated **226 Metrics** of the Elasticsearch Monitoring Extension.

   f. Couchbase Metrics
      Verified and validated **390 Metrics** of the Couchbase Monitoring Extension.

   g. MySQL Metrics
      Verified and validated **93 Metrics** and **10 Attributes** of the MySQL Monitoring Extension.

   h. Amazon Web Services Monitoring Metrics
      Verified and validated **270 Metrics (approximately)** and **70 attributes** of the Amazon Web Service Monitoring Extension.

   i. Azure Metrics
      Verified and validated **120 Metrics (approximately)** and **50 attributes** of the Azure Monitoring Extension.

   j. Apache Flume Metrics
      Verified and validated **34 Metrics** of the Apache Flume Monitoring Extension.

   k. SQL Server Monitoring Metrics
      Verified and validated **140 Metrics** and **12 Attributes** of the SQL Server Monitoring Extension.

Sample 3: GitHub Related Work

To showcase my work, used a sample use-case of “How to Withdraw Money from ATM Machine?”. I have created a public repository in GitHub and authored the procedure using Markdown language.

Uploaded the following relevant documents to the GitHub repository:

- An image of the ATM machine
- Source code of the procedure, which is written in Markdown language

Sample Work

The link of the GitHub page site with the simple procedure of *“How to Withdraw Money from ATM Machine?”* is as follows:
https://github.com/ManasiPimprikar/WorkSamples/wiki

I have provided the link to my GitHub repository containing the source code of the simple procedure of *“How to Withdraw Money from ATM Machine?”.*

You can access the source code of the simple procedure of *“How to Withdraw Money from ATM Machine?”* in the word document, *StretchTask_in_Markdown_Language.docx* from the following location:
https://github.com/ManasiPimprikar/WorkSamples

Sample 4: API Provide Example

Question # A basic example based on the echo-service quickstart, demonstrating how to configure an API provider and consumer, and test a defined usage policy.

Approach

To come up with a sample use case of using a sample echo service API request of Apiman quickstart, I used Google to search for a “sample echo service API request”:
https://github.com/apiman/apiman-quickstarts/tree/master/echo-service/

I had observed that the following reference link did NOT follow a proper chronology of events on how to configure an API provider and consumer:
https://github.com/apiman/apiman-quickstarts/tree/master/echo-service/

Therefore, I came up with the instructions on how to configure an API provider and consumer for a user, who is starting from the scratch.

In my procedure, I provided details related to **pre-requisites** and ensured to maintain the chronology of tasks to be performed to configure an API provider and consumer.

Answer # How to Configure an API Provider and Consumer

You can configure an API provider by using an application, *Apiman-quickstarts*. To configure an API provider, you need to build, deploy, and run the application. After you deploy and run the application, you can access and invoke the APIs of the application to execute a defined usage policy.

Prerequisites

1. You have the required access to a Github repository.
2. WildFly 8.x server is up and running.
3. You have access to a REST Client

Clone the Repository

As a prerequisite to configuring an API provider and consumer, ensure that you clone or download a copy of the repository to a local host by executing the following command:

```
git clone https://github.com/apiman/apiman-quickstarts.git
```

Build and Deploy the Repository

You can build and deploy the repository, *apiman-quickstarts.git* on WildFly 8.x server by executing the following command:

```
cd apiman-quickstarts/echo-service mvn clean install mvn
wildfly:deploy
```

Now, your Apiman-echo application is deployed and running in the WildFly server. For more information, see WildFly documentation.

Verify that you can access the REST Echo Service at the following address:
http://losangeles-localhost:8080/Apiman-echo/sample/path

Invoke the Echo Service

When you invoke an Echo service, this project produces a simple WAR that implements an "echo" REST web service, which is useful for testing and demo-ing Apiman.

When you make a request to the **echo** service. A successful API request made to the echo service will respond with a *200* response code and a JSON response body that includes all the meta-data that is sent, along with the echo service in the request. Eventually, the body of the response that is returned includes the request method, path, and headers wrapped into a JSON object.

For example, if you invoke the end-point of the echo service as follows:
http://losangeles-localhost:8080/Apiman-echo/sample/path

Then the response payload may be as follows:

```
{"method" : "GET", "resource" : "/Apiman-echo/sample/path", "length" : -1,
"uri" :
"/Apiman-echo/sample/path", "headers" : { "Accept-Language" :
"en-US,en;q=0.5", "Host" :
"losangeles-localhost:8080", "Accept-Encoding" : "gzip, deflate",
"User-Agent" :
"Mozilla/5.0 (Windows NT 6.3; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0",
"Accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8",
"Connection" : "keepalive" } }
```

Test the API Provider

You can verify the API provider using any third-party Client API testing tools such as RESTClient, Postman, and so on.