z/OS 3.1 IBM Education Assistant

Solution Name: IBM SMF Explorer with z/OS Data Gatherer REST services

Solution Element(s): z/OS Data Gatherer

July 2023

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

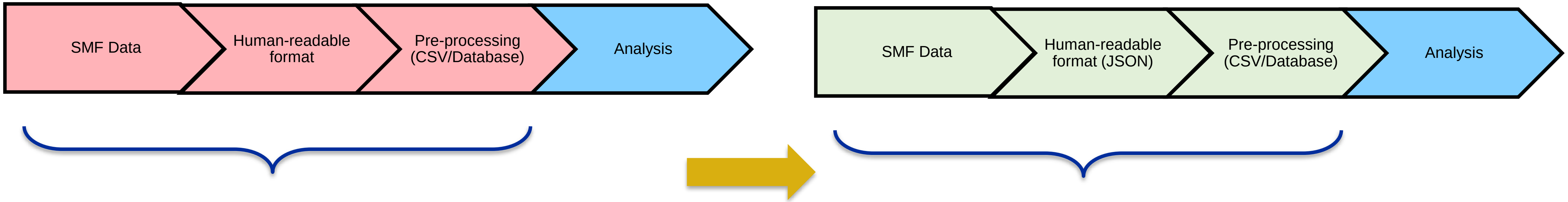
Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - IBM® SMF Explorer with Python

Objectives

- Provide an overview about the SMF Explorer and z/OS Data Gatherer Rest Services for SMF data access
- Motivation:
    - SMF (System management facilities) data:
        - contains valuable data describing activity of the z/OS System
    - Traditional SMF data analysis:
        - Complex, time-consuming
        - Deep z/OS domain knowledge required
        - Hard to do quick-prototyping

Objectives (2)



SMF Data

Human-readable format

Pre-processing (CSV/Database)

Analysis

SMF Data

Human-readable format (JSON)

Pre-processing (CSV/Database)

Analysis

Manually, might take hours or even days

With DG Rest Services + SMF Explorer, only takes minutes

Overview

- Who (Audience)
  - SMF Data Consumers, such as data scientists, system programmers, application developers, performance analysts, capacity planners,…

Application Developer

z/OS performance analysts

z/OS system programmers

Data Scientist

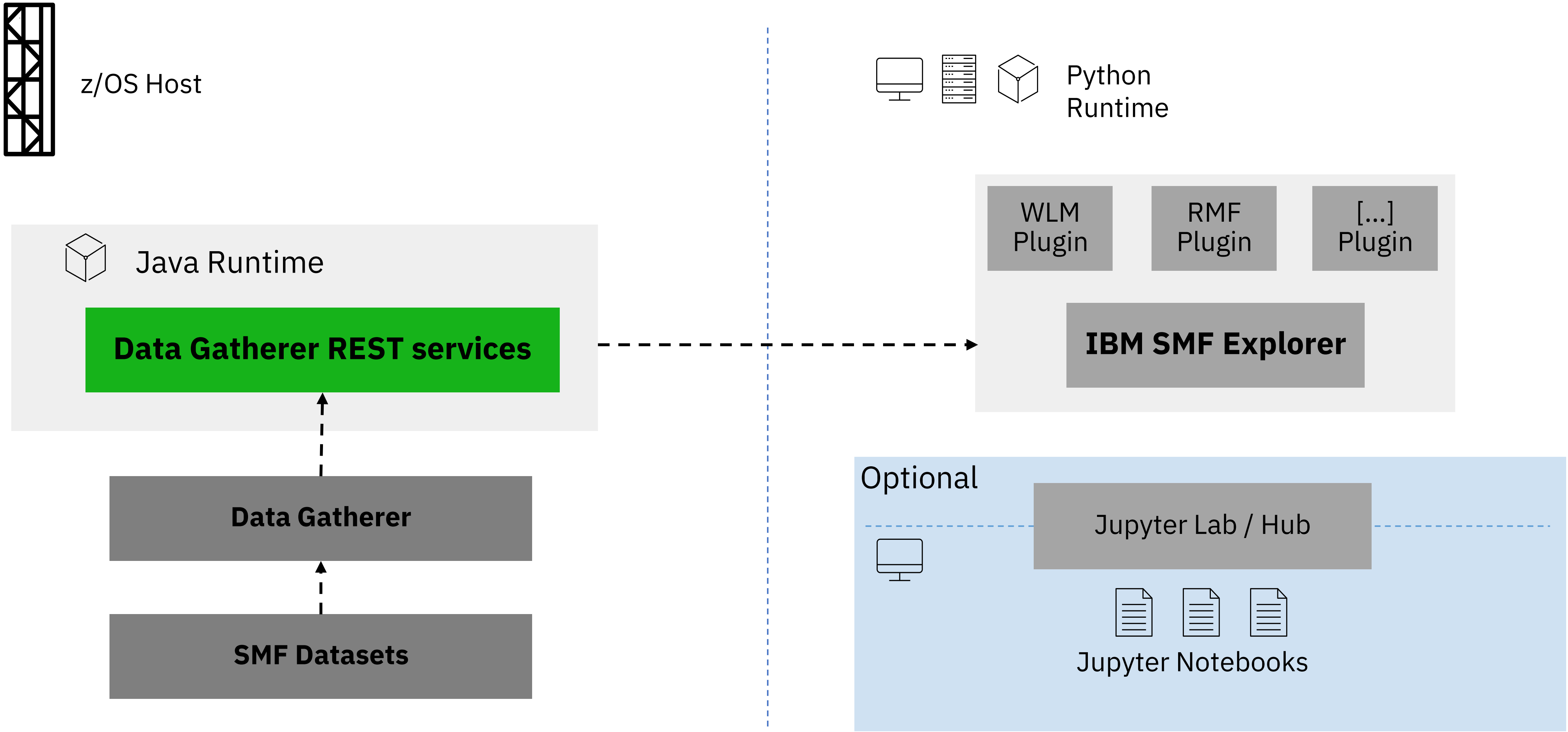
z/OS capacity planners

Overview (2)

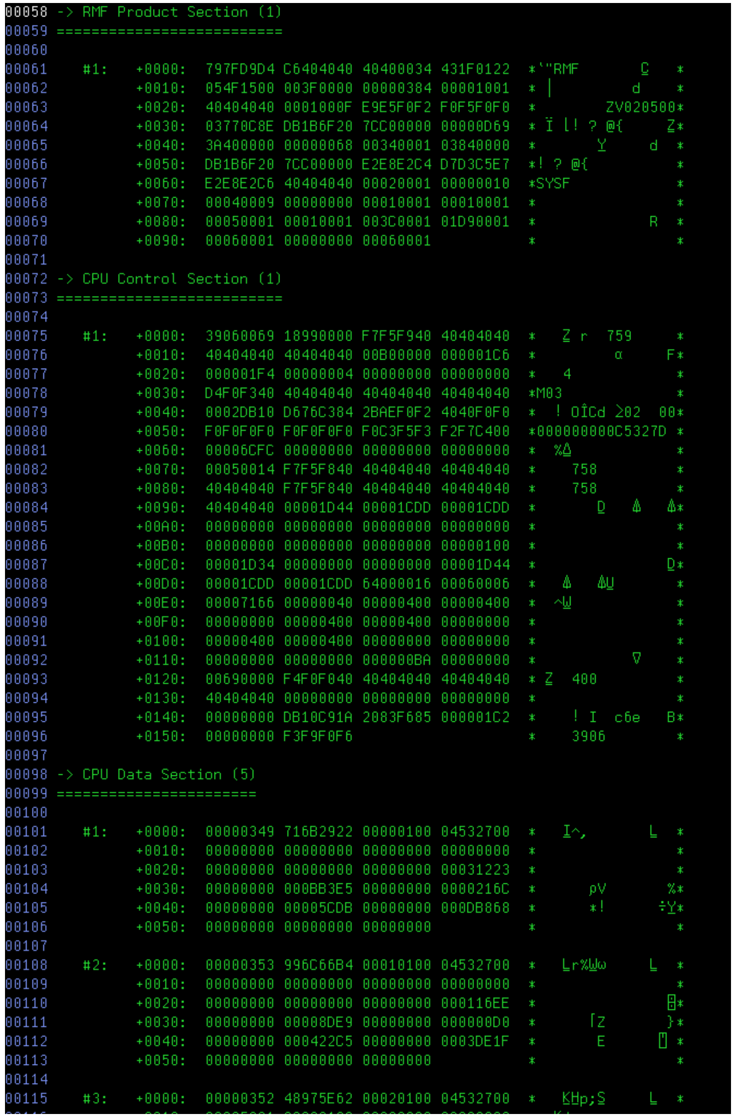
- What (Solution)
  - Data Gather REST Services for SMF data access from the z/OS® host
  - Python framework to fetch SMF data leveraging the REST-API
  - Jupyter Notebook tutorials and support for quick prototyping of reports and analyses.

- Wow (Benefit / Value, Need Addressed)
  - Data Gather REST Services for SMF data access
    - Raw SMF data is modelled and formatted in JSON objects
    - Compliant with industry standard
  - IBM SMF Explorer: modern, convenient way to access and process SMF data with Python
    - SMF data is returned in a table-like format for further analysis.
    - No deep z/OS skills are required to access and process data.

Architecture

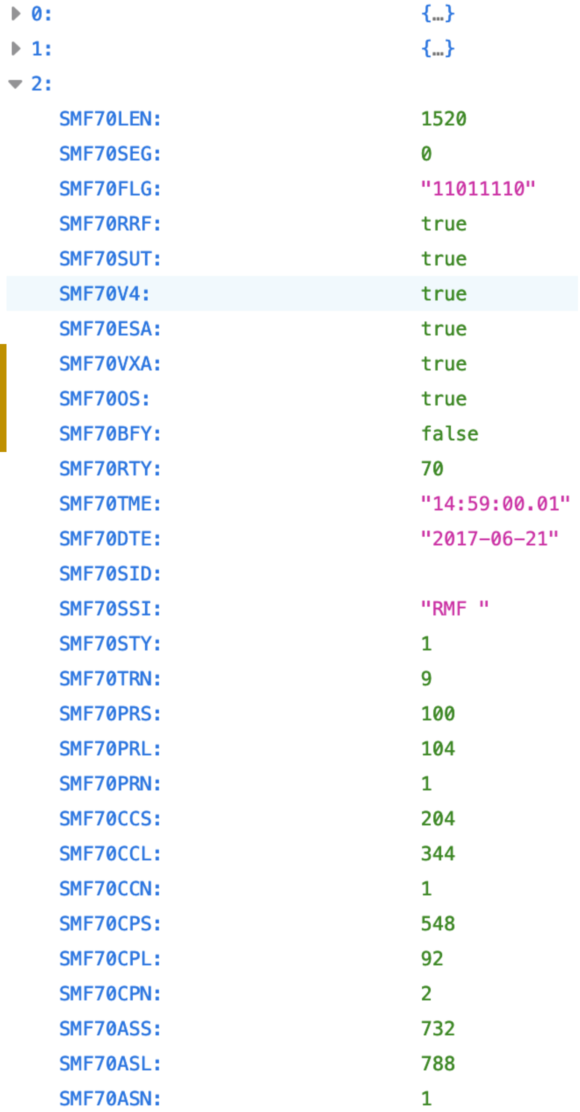


From binaries to …



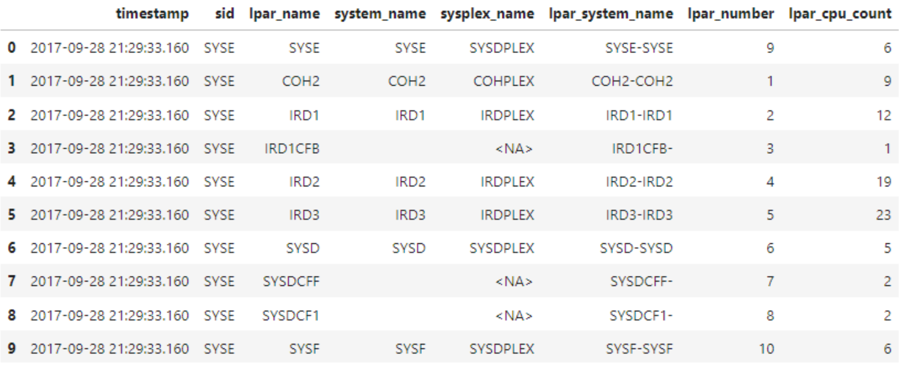
```
-> RMF Product Section (1)
==========================

  #1:   +0000:  797FD9D4 C6404040 40400034 431F0122  *`"RMF       C   *
        +0010:  054F1500 003F0000 00000384 00001001  * |          d   *
        +0020:  40404040 0001000F E9E5F0F2 F0F5F0F0  *        ZV020500*
        +0030:  03770C8E DB1B6F20 7CC00000 00000D69  * Ï l! ? @{     Z*
        +0040:  3A400000 00000068 00340001 03840000  *       Y     d  *
        +0050:  DB1B6F20 7CC00000 E2E8E2C4 D7D3C5E7  *! ? @{          *
        +0060:  E2E8E2C6 40404040 00020001 00000010  *SYSF            *
        +0070:  00040009 00000000 00010001 00010001  *                *
        +0080:  00050001 00010001 003C0001 01D90001  *             R  *
        +0090:  00060001 00000000 00060001           *                *

-> CPU Control Section (1)
==========================

  #1:   +0000:  39060069 18990000 F7F5F940 40404040  *   Z r  759     *
        +0010:  40404040 40404040 00B00000 000001C6  *          α    F*
        +0020:  000001F4 00000004 00000000 00000000  *   4            *
        +0030:  D4F0F340 40404040 40404040 40404040  *M03             *
        +0040:  0002DB10 D676C384 2BAEF0F2 4040F0F0  *  ! OÎCd ≥02  00*
        +0050:  F0F0F0F0 F0F0F0F0 F0C3F5F3 F2F7C400  *000000000C5327D *
        +0060:  00006CFC 00000000 00000000 00000000  *  %⍋            *
        +0070:  00050014 F7F5F840 40404040 40404040  *    758         *
        +0080:  40404040 F7F5F840 40404040 40404040  *    758         *
        +0090:  40404040 00001D44 00001CDD 00001CDD  *       D   ⍋   ⍋*
        +00A0:  00000000 00000000 00000000 00000000  *                *
        +00B0:  00000000 00000000 00000000 00000100  *                *
        +00C0:  00001D34 00000000 00000000 00001D44  *               D*
        +00D0:  00001CDD 00001CDD 64000016 00060006  *   ⍋   ⍋U       *
        +00E0:  00007166 00000040 00000400 00000400  *  ^W            *
        +00F0:  00000000 00000400 00000400 00000000  *                *
        +0100:  00000400 00000400 00000000 00000000  *                *
        +0110:  00000000 00000000 000000BA 00000000  *           ∇    *
        +0120:  00690000 F4F0F040 40404040 40404040  * Z  400         *
        +0130:  40404040 00000000 00000000 00000000  *                *
        +0140:  00000000 DB10C91A 2083F685 000001C2  *    ! I  c6e   B*
        +0150:  00000000 F3F9F0F6                    *    3906        *

-> CPU Data Section (5)
=======================

  #1:   +0000:  00000349 716B2922 00000100 04532700  *   I^,       L  *
        +0010:  00000000 00000000 00000000 00000000  *                *
        +0020:  00000000 00000000 00000000 00031223  *                *
        +0030:  00000000 000BB3E5 00000000 0000216C  *      ρV       %*
        +0040:  00000000 00005CDB 00000000 000DB868  *      *!      ÷Y*
        +0050:  00000000 00000000 00000000           *                *

  #2:   +0000:  00000353 996C66B4 00010100 04532700  *   Lr%Wω     L  *
        +0010:  00000000 00000000 00000000 00000000  *                *
        +0020:  00000000 00000000 00000000 000116EE  *               ⍠*
        +0030:  00000000 00008DE9 00000000 000000D0  *      ⌈Z       }*
        +0040:  00000000 000422C5 00000000 0003DE1F  *       E      ▯ *
        +0050:  00000000 00000000 00000000           *                *

  #3:   +0000:  00000352 48975E62 00020100 04532700  *   KHp;S     L  *
```

Data Gatherer REST services



```
▸ 0:          {…}
▸ 1:          {…}
▾ 2:
    SMF70LEN:     1520
    SMF70SEG:     0
    SMF70FLG:     "11011110"
    SMF70RRF:     true
    SMF70SUT:     true
    SMF70V4:      true
    SMF70ESA:     true
    SMF70VXA:     true
    SMF70OS:      true
    SMF70BFY:     false
    SMF70RTY:     70
    SMF70TME:     "14:59:00.01"
    SMF70DTE:     "2017-06-21"
    SMF70SID:
    SMF70SSI:     "RMF "
    SMF70STY:     1
    SMF70TRN:     9
    SMF70PRS:     100
    SMF70PRL:     104
    SMF70PRN:     1
    SMF70CCS:     204
    SMF70CCL:     344
    SMF70CCN:     1
    SMF70CPS:     548
    SMF70CPL:     92
    SMF70CPN:     2
    SMF70ASS:     732
    SMF70ASL:     788
    SMF70ASN:     1
```

IBM SMF Explorer



	timestamp	sid	lpar_name	system_name	sysplex_name	lpar_system_name	lpar_number	lpar_cpu_count
0	2017-09-28 21:29:33.160	SYSE	SYSE	SYSE	SYSDPLEX	SYSE-SYSE	9	6
1	2017-09-28 21:29:33.160	SYSE	COH2	COH2	COHPLEX	COH2-COH2	1	9
2	2017-09-28 21:29:33.160	SYSE	IRD1	IRD1	IRDPLEX	IRD1-IRD1	2	12
3	2017-09-28 21:29:33.160	SYSE	IRD1CFB		<NA>	IRD1CFB-	3	1
4	2017-09-28 21:29:33.160	SYSE	IRD2	IRD2	IRDPLEX	IRD2-IRD2	4	19
5	2017-09-28 21:29:33.160	SYSE	IRD3	IRD3	IRDPLEX	IRD3-IRD3	5	23
6	2017-09-28 21:29:33.160	SYSE	SYSD	SYSD	SYSDPLEX	SYSD-SYSD	6	5
7	2017-09-28 21:29:33.160	SYSE	SYSDCFF		<NA>	SYSDCFF-	7	2
8	2017-09-28 21:29:33.160	SYSE	SYSDCF1		<NA>	SYSDCF1-	8	2
9	2017-09-28 21:29:33.160	SYSE	SYSF	SYSF	SYSDPLEX	SYSF-SYSF	10	6

z/OS Data Gatherer: SMF REST Services: Overview

- z/OS Data Gatherer: SMF REST Services:
  - Part of z/OS-base infrastructure
  - Access to SMF data
  - Data on demand using client-initiated synchronous Request-Response calls
  - Support all SMF types 70-79, 99.x (subtype 1, 2, 6, 12 and 14), 113.x
  - Modelled and formatted raw data in JSON as they are in the Dataset
    - each Object includes all the hierarchy requested and / or available
    - each Object represents an SMF record of a specific (sub)type
    - each Object includes all the fields requested and / or available

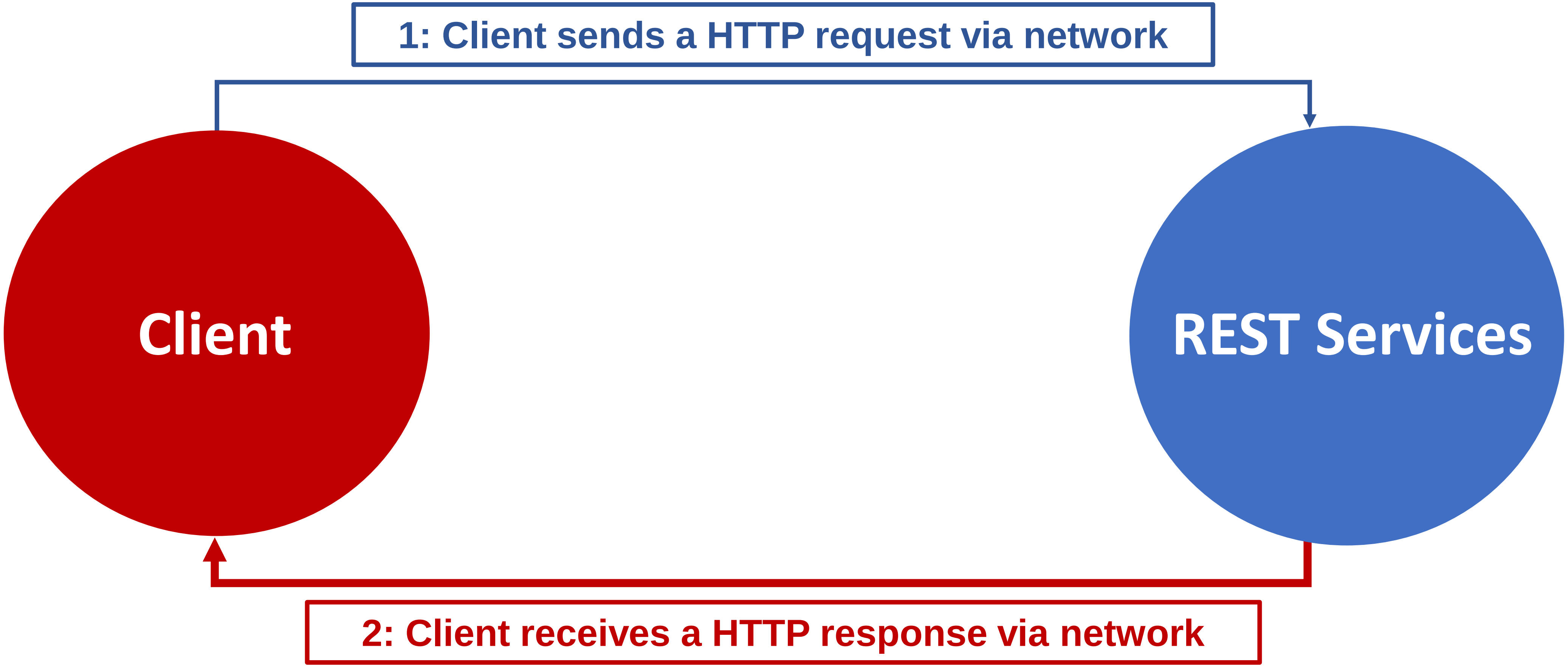
z/OS Data Gatherer: SMF REST Services

Usage & Invocation

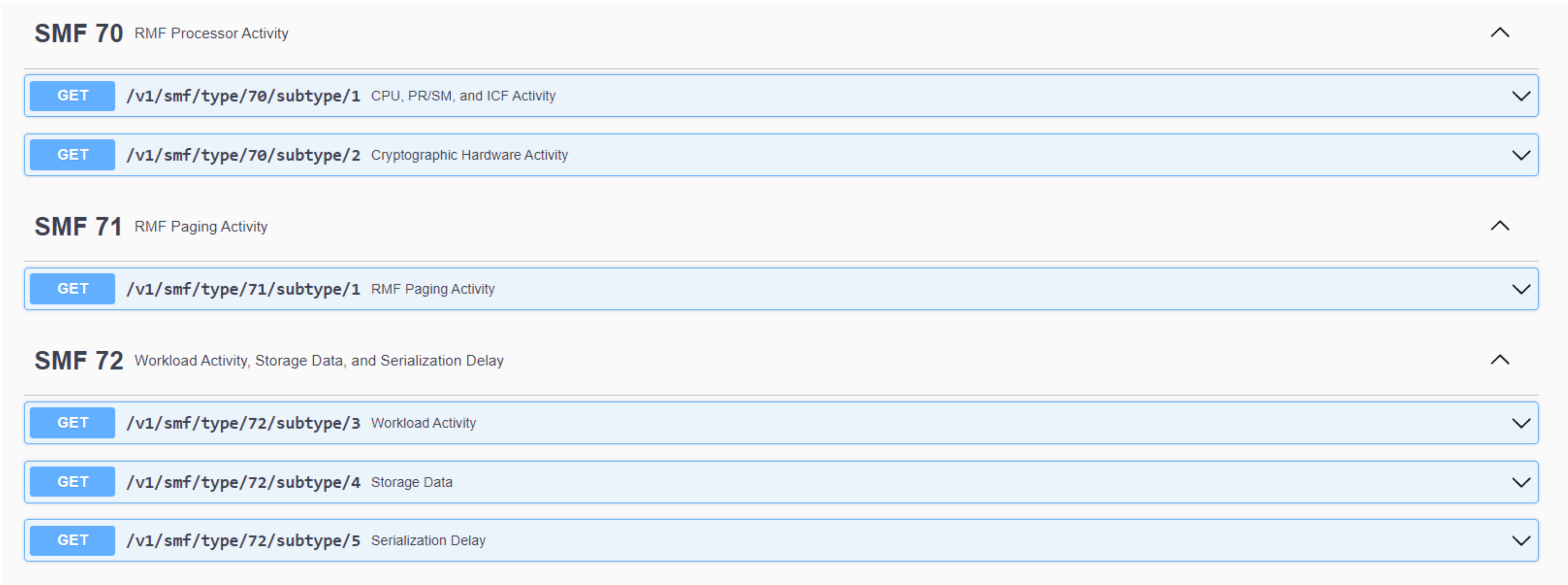
- A caller
  - web-browser
  - front-end application
  - API consuming platform
  - programmatic caller (e.g. IBM SMF Explorer)
  - including generated client, e.g., via OpenAPI Generator
  - command line caller (e.g., CURL)
- Example URL: https://host:port/zosmf/zosdg/smf/

Data Gatherer REST services: Request-Response Model

Usage & Invocation: Client-initiated Request-Response Model,


1: Client sends a HTTP request via network
Client
REST Services
2: Client receives a HTTP response via network

Data Gatherer REST services: Discover SMF Record Data (1)



SMF 70 RMF Processor Activity

GET `/v1/smf/type/70/subtype/1` CPU, PR/SM, and ICF Activity

GET `/v1/smf/type/70/subtype/2` Cryptographic Hardware Activity

SMF 71 RMF Paging Activity

GET `/v1/smf/type/71/subtype/1` RMF Paging Activity

SMF 72 Workload Activity, Storage Data, and Serialization Delay

GET `/v1/smf/type/72/subtype/3` Workload Activity

GET `/v1/smf/type/72/subtype/4` Storage Data

GET `/v1/smf/type/72/subtype/5` Serialization Delay

… up to SMF 79

SMF 99 subtypes 1, 2, 6, 12 and 14

SMF 113 subtypes 1 and 2

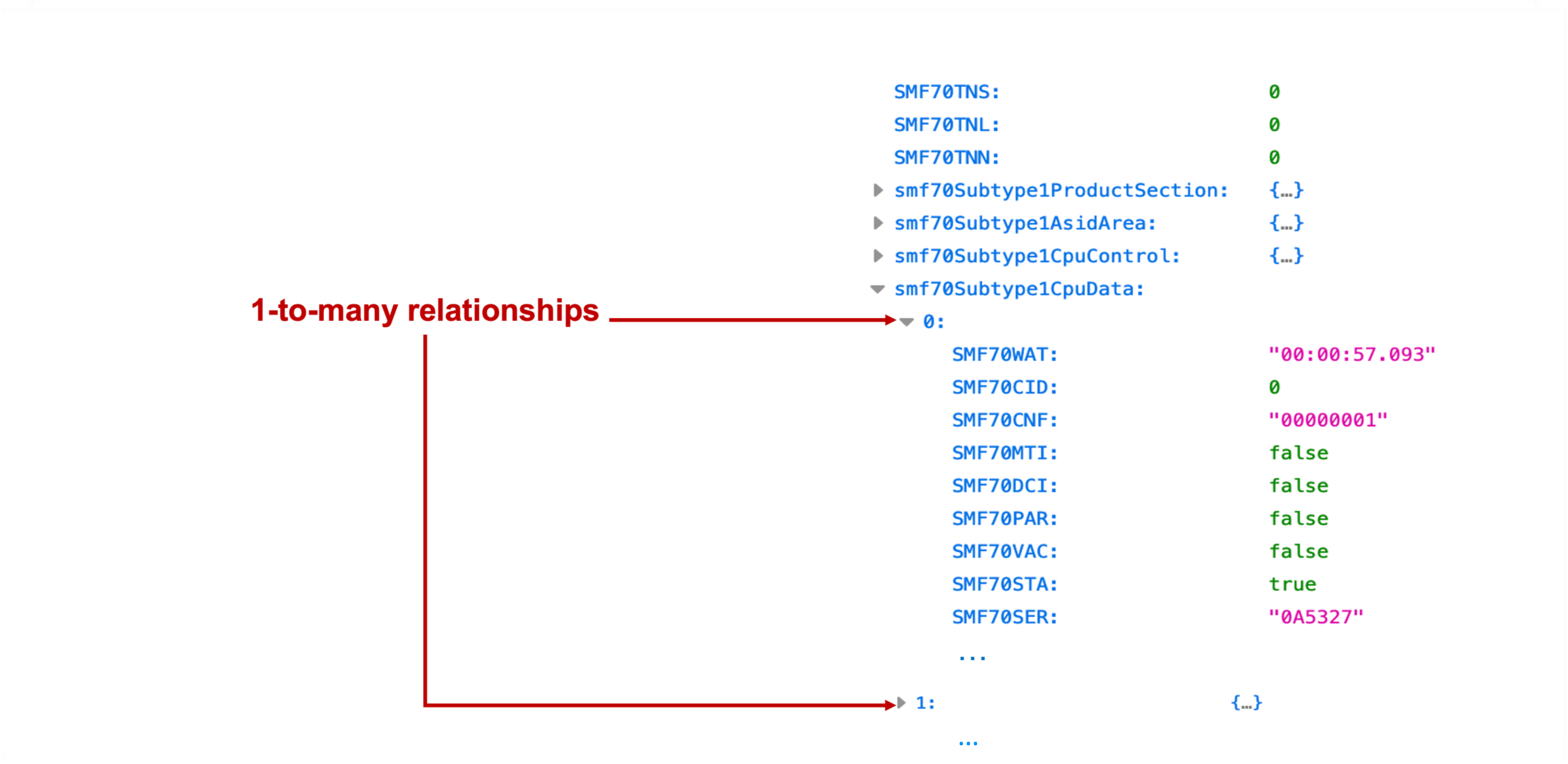
Swagger interface: https://host:port/zosmf/zosdg/smf/

Data Gatherer REST services: Discover SMF Record Data (2)

{**host**}:{**port**}/{**context-path**}/v1/smf/type/{#**type**}/subtype/{#**subtype**}?datasetName={**smf.dataset.name**}&{opt}

Element	Description	Example(s)
Host	Machine’s network identifier	ibm.example.com
Port	The port assigned to the Web Application Server the application is deployed at	444
Context-path	The the prefix of a URL-path	zosmf/zosdg/smf/
#type	The SMF type the application shall process	70 / 71 / 78
#subtype	The subtype of the SMF type the application shall process	1 / 2 / 11
smf.dataset.name	The name of a dataset to process	SOME.VALID.SMF.DATASET

Data Gatherer REST services: Example Results



```
   SMF70TNS:                       0
   SMF70TNL:                       0
   SMF70TNN:                       0
 ▶ smf70Subtype1ProductSection:    {…}
 ▶ smf70Subtype1AsidArea:          {…}
 ▶ smf70Subtype1CpuControl:        {…}
 ▼ smf70Subtype1CpuData:
   ▼ 0:
       SMF70WAT:                   "00:00:57.093"
       SMF70CID:                   0
       SMF70CNF:                   "00000001"
       SMF70MTI:                   false
       SMF70DCI:                   false
       SMF70PAR:                   false
       SMF70VAC:                   false
       SMF70STA:                   true
       SMF70SER:                   "0A5327"
       ...
   ▶ 1:                         {…}
       ...
```

1-to-many relationships

Data Gatherer REST services: Interactions & Dependencies

- Software Dependencies

  None

Data Gatherer REST services: Installation & Configuration

- No hardware configuration required
- No runtime dependencies on other z/OSMF services and the **primary z/OSMF server**
- z/OS Data Gatherer: SMF REST Services must be run on a **separate z/OSMF server instance**
- If z/OS Data Gatherer: SMF REST Services are enabled on a z/OSMF server instance, all other services in that z/OSMF server are disabled
- z/OS Data Gatherer User's Guide (SC31-5703-50)

Data Gatherer REST services: Installation & Configuration (2)

1. Setup a separate instance of z/OSMF with its own user file system (follow IBM z/OSMF Configuration Guide)

2. Enable the z/OSMF server to perform authorisation checks for resources in the DATASET class

3. Log in to z/OSMF and enable z/OS Data Gatherer SMF REST services that are listed under "Optional Services”

4. Restart the z/OSMF server.

5. To verify that the REST services are running, open the Swagger interface at https://host:port/zosmf/zosdg/smf/

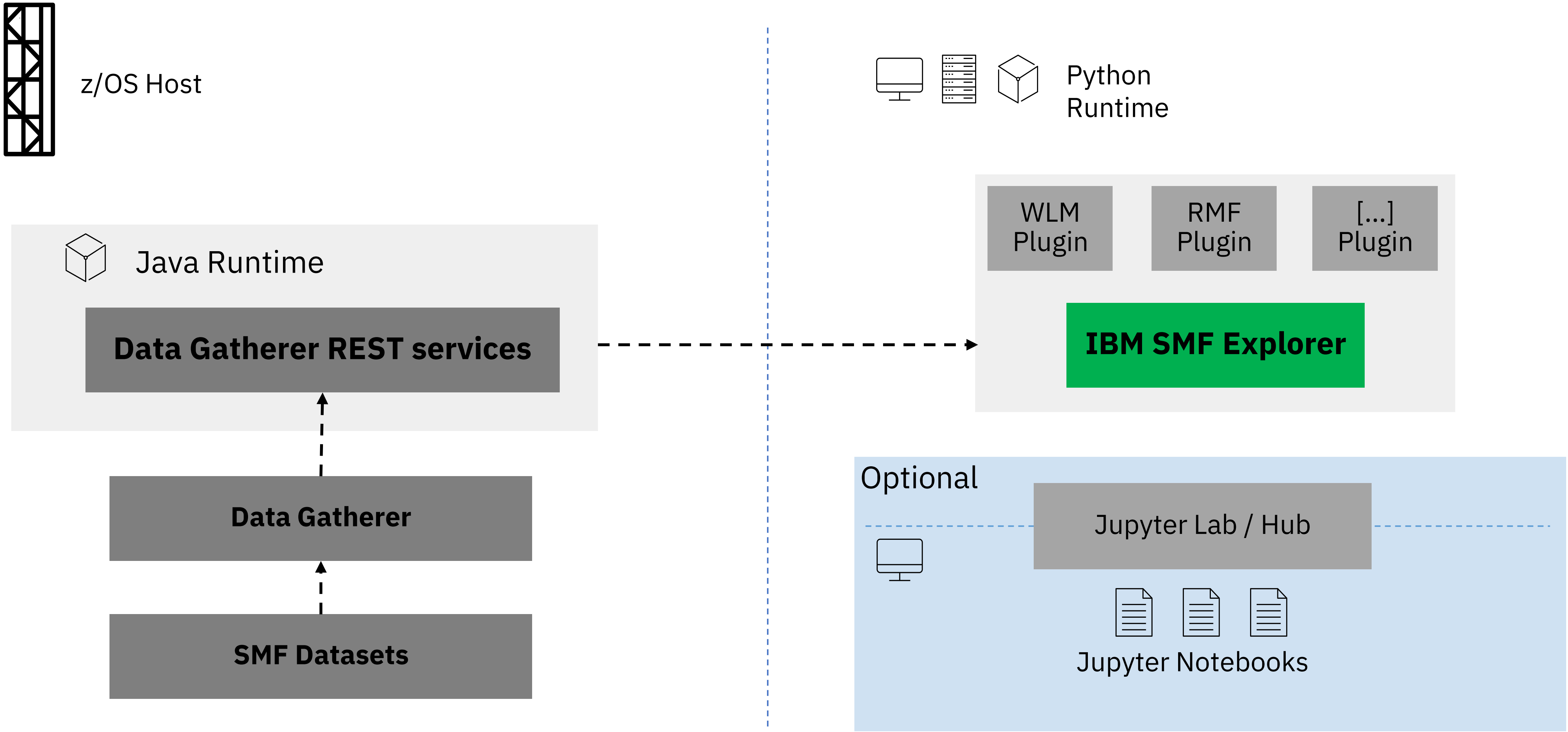
Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex must be at the new z/OS level: No
- List any toleration/coexistence APARs/PTFs.
  - None

Summary

- z/OS Data Gatherer: SMF REST Services:
  - Data on demand using client-initiated synchronous Request-Response calls
  - Full Data Model Composition including parent-child hierarchies and 1-to-1 & 1-to-many relationships
  - Full field-level data representation in common formats

Architecture



z/OS Host

Java Runtime

Data Gatherer REST services

Data Gatherer

SMF Datasets

Python Runtime

WLM Plugin

RMF Plugin

[...] Plugin

IBM SMF Explorer

Optional

Jupyter Lab / Hub

Jupyter Notebooks

IBM SMF Explorer: Introduction

- IBM SMF Explorer is a framework for SMF data access using Python
- Why Python?
  - Python is easy to learn and widely used in the data science community
  - Many packages available for data visualization and analysis
  - Quick prototyping
- Part of the z/OS base. Non-priced, no additional license required

IBM SMF Explorer: Introduction (2)

- Features:
  - SMF data retrieval
  - Filtering, Sorting
  - Multi-Dataset access
- Shipment:
  - IBM SMF Explorer Python package
  - Jupyter Notebook tutorials to simplify the entry into SMF data analysis
- Leverage the REST Services provided by the z/OS Data Gatherer

IBM SMF Explorer: Usage & Invocation

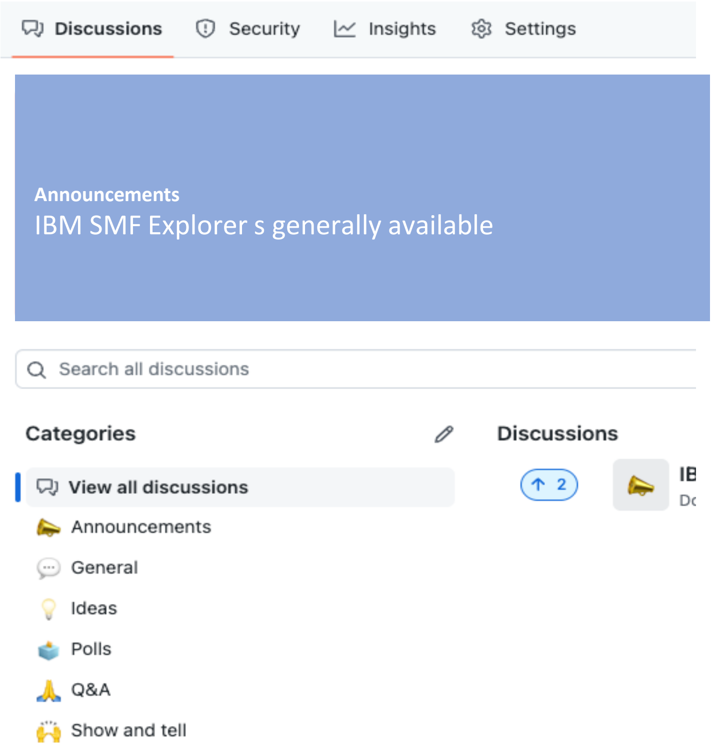
- Prerequisite: z/OS Data Gatherer SMF REST services are running in z/OS Host
- Installation scripts and Jupyter Notebook tutorials are open source

Jupyter Notebooks & GitHub Community

- Notebooks are the main deliverable besides the IBM SMF Explorer Python package
- They give you an easy entry point to IBM SMF Explorer
- Well established tool in the data science world
- Straightforward user experience

We want to build a community around Notebooks

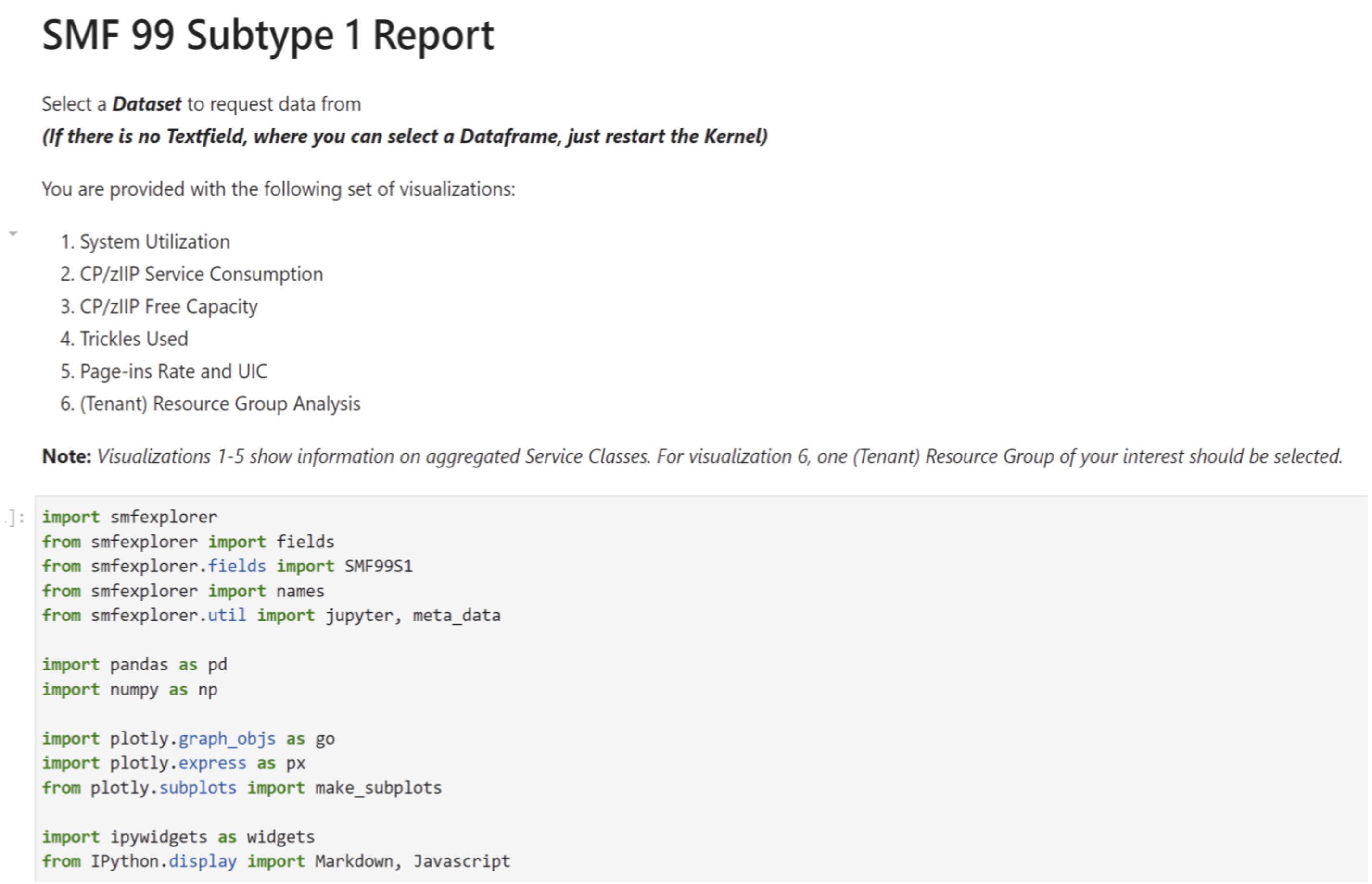
Using GitHub to share our and initial Notebooks and give everyone the opportunity to learn, adapt and contribute



Announcements

IBM SMF Explorer s generally available

Demo



SMF 99 Subtype 1 Report

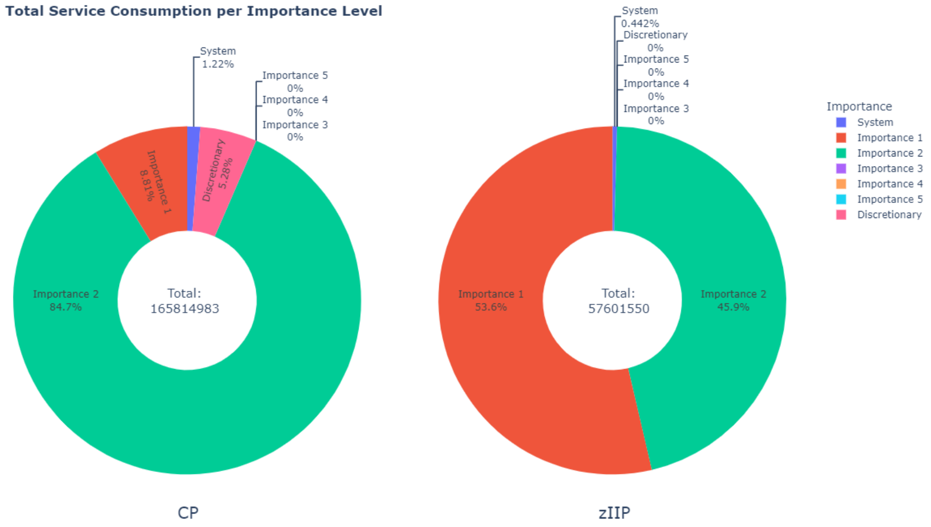
Select a ***Dataset*** to request data from
(If there is no Textfield, where you can select a Dataframe, just restart the Kernel)

You are provided with the following set of visualizations:

1. System Utilization
2. CP/zIIP Service Consumption
3. CP/zIIP Free Capacity
4. Trickles Used
5. Page-ins Rate and UIC
6. (Tenant) Resource Group Analysis

Note: *Visualizations 1-5 show information on aggregated Service Classes. For visualization 6, one (Tenant) Resource Group of your interest should be selected.*

```
import smfexplorer
from smfexplorer import fields
from smfexplorer.fields import SMF99S1
from smfexplorer import names
from smfexplorer.util import jupyter, meta_data

import pandas as pd
import numpy as np

import plotly.graph_objs as go
import plotly.express as px
from plotly.subplots import make_subplots

import ipywidgets as widgets
from IPython.display import Markdown, Javascript
```



Total Service Consumption per Importance Level

IBM SMF Explorer: Installation

- Shipped together with z/OS Data Gatherer: SMF Data REST Services
- The Python package can be found in USS directory *~/usr/lpp/IBM/zdg/smf_explorer~* on your z/OS Host
- The setup scripts with Jupyter Notebook tutorials are in the public Github repository: IBM/IBM-SMF-Explorer (github.com)

IBM SMF Explorer: Interactions & Dependencies

- Software Dependencies
  - z/OS Host: None
  - Workstation:
    - Python 3.8 or later
    - Git (optional)
    - JupyterLab environment
    - Non-priced
- Hardware Dependencies
  - None.
- Exploiters
  - None.

IBM SMF Explorer - Summary

- IBM SMF Explorer is a z/OS non-priced offering for SMF data retrieval.
- Users can conveniently access and process SMF data with Python and Jupyter Notebooks, even with limited z/OS knowledge.
- Users can leverage data analytics skills to gain insights into the data without processing the raw SMF data.

Takeaway

- Data Gatherer REST Services:
  - provides data in JSON format
  - compliant with industry standard
- IBM SMF Explorer:
  - provides data in tabular format for data analysis
- Current support: SMF types 7X, 99 (subtypes 1, 2, 6, 12 and 14) and 113

Appendix

- SMF Explorer:
  - Hot Topics article: How to turn your SMF data into valuable insights without z/OS expertise (zos-hot-topics.com)
  - Documentation: IBM SMF Explorer
  - External github repo with installation scripts and Jupyter Notebooks:
    - IBM/IBM-SMF-Explorer
- Data Gather REST services
  - z/OS Data Gatherer Programmer's Guide
  - z/OS Data Gatherer User's Guide
  - z/OS Management Facility Configuration Guide
  - MVS System Management Facilities (SMF)
  - https://github.com/IBM/IBM-Z-zOS/tree/main/zOS-DataGatherer