IBM CICS Performance Analyzer for z/OS, V5.4

Output to JSON Lines

User documentation

Document information

Last modified	24 September 2021
Copyright	© Rocket Software 2019, 2020, 2021
Author	Graham Hannington (ghannington@rocketsoftware.com) Daniel John Lalwet (dlalwet@rocketsoftware.com)
Jira issue	CPA-2605, CPA-3055, CPA-3090 (Rocket Perth)

The latest version of this document is available on the web in CICS SupportPac CA10.

Change history

Most recent changes first:

Publication date *yyyy-mm-dd*	Summary of changes (with Jira issue number, where applicable)
2021-09-24	Include the Enqueue Manager statistics summary dashboard information.
2021-08-24	Added documentation for z/OS Connect EE dashboards in the Splunk app.
2020-10-30	Updated example Logstash config with workarounds to convert code property value to lowercase and to ensure large input values in scientific notation are serialized to output as integer values.
2020-03-04	Added links to newly available web addresses: the app on Splunkbase and CICS SupportPac CA10.
2020-02-28	First edition for V5.4 APAR PH16158 GA (general availability).
2020-01-10	Update for 1.0.0-beta.2 of the IBM-supplied sample Splunk app. Added more statistics data and corresponding charts (CPA-2721). Removed performance-list-based "Transaction overview detail" dashboard (CPA-2728). Changed default time range of dashboards from "All time" to "Last 24 hours" (CPA-2748). Various enhancements to dashboards (CPA-2745, CPA-2748).
2019-12-13	First edition for V5.4 APAR PH16158 beta (CPA-2695).

Contents

Overview

This document describes enhancements introduced in IBM CICS Performance Analyzer for z/OS, V5.4, APARs PH16158 and PH37264 to output CICS performance, statistics and z/OS Connect EE data in JSON Lines format.

Topics covered in this document include:

- Using CICS Performance Analyzer to output data in JSON Lines format

- Installing the IBM-supplied sample Splunk app and sample data that demonstrate using JSON Lines from CICS Performance Analyzer

- Getting JSON Lines from CICS Performance Analyzer into:

    - Splunk

    - Elastic Stack

The information in this document that is specifically about CICS Performance Analyzer, rather than about Splunk or the Elastic Stack, is also available in the CICS Performance Analyzer product documentation in IBM Knowledge Center.

Who this document is for

This document is for the following users:

- **CICS Performance Analyzer users** who want to output CICS performance and statistics data in JSON Lines format

- **Splunk users, or users of other analytics platforms,** who need to work with JSON Lines from CICS Performance Analyzer

What you need to know

This document assumes that you already understand the following topics:

CICS Performance Analyzer report sets, report forms, and statistics alerts
You only need to understand these topics if you are a CICS Performance Analyzer user who wants to output JSON Lines. You don't need to understand these topics if, for example, you are a Splunk app developer who wants to *use* that JSON Lines.

The enhancements for output to JSON Lines extend the existing report set, report form, and statistics alert features in CICS Performance Analyzer. This document describes only the *changes* to these features; it does not step through the complete process of defining report forms or statistics alerts and then using them in report sets.

Splunk, Elastic Stack, or another analytics platform of your choice

This document describes the JSON Lines output by CICS Performance Analyzer and how to configure Splunk or Elastic to ingest that data.

This document does *not* describe how to use Splunk or Elastic in general.

To configure other analytics platforms, read the documentation provided here about the JSON Lines output by CICS Performance Analyzer, and then see the documentation for those analytics platforms.

JSON Lines

If you are a developer who needs to use data from CICS Performance Analyzer in an analytics platform, then you should understand the JSON Lines format before reading about the specific characteristics of JSON Lines from CICS Performance Analyzer.

Terminology

CSV

This document uses the term CSV to mean "*delimiter*-separated values (DSV), including *comma*-separated values (CSV)". In CICS Performance Analyzer, you can use the `DELIMIT` parameter to specify which character to use as a field separator.

Extracting

In CICS Performance Analyzer, *extract* is sometimes used as a synonym for *CSV file*; and now, also *JSON Lines file*. *Extracting* is sometimes used to mean *writing CSV or JSON Lines to a file*. The file could be an MVS data set or a z/OS UNIX file.

Forwarding

In CICS Performance Analyzer, writing JSON Lines or CSV data over a TCP network directly to an analytics platform is referred to as **forwarding**. In this sense, *forwarding* contrasts with *extracting*.

More generally, extracting data to a file is often the first step in a multi-step forwarding process. The use of the term *forwarding* in this document, and in CICS Performance Analyzer, emphasizes that writing data over a network offers a single-step forwarding process that contrasts with extracting, transferring, and then loading.

Requests for enhancement (RFEs)

The enhancements described in this document address the following public RFEs:

CICS PA Web solution (RFE 83233)

"web based analytics to CICS PA, where the PA reports ... can be sent to a web server ... accessed through browser ... dashboard / charts"

"trend analysis easily without running multiple reports ... store data for a very long period ... over a year+ ... queries ... easier than ... from DB2 / UDB"

JSON format output (RFE 120510)

"Can we have a switch ... to produce a JSON format file"

Enable Splunk as a destination for CICS PA (RFE 132212)

"We generate many CICS PA threshold reports ... difficult to view singly if you are trying to understand multiple days and/or multiple regions. If the raw data for the reports are sent to Splunk, we can generate Splunk reports that show a larger view of the data."

Enhance CICS PA to process zOS Connect EE SMF 123 records (RFE 104690)

"We use CICS PA here and are beginning to exploit zOS Connect to access some of our CICS applications. Since PA can process SMF data from other subsystems that interact with CICS such as CTG, MQ, and DB2, we would like to be able to process the SMF 123 subtype 1s from zOS Connect similarly for reporting purposes."

Existing functionality

To understand the enhancements for JSON Lines, it's useful to understand how CICS Performance Analyzer can already extract data for use in analytics platforms. The enhancements for JSON Lines extend that existing functionality.

CICS Performance Analyzer can already extract data in CSV format for use in analytics platforms.
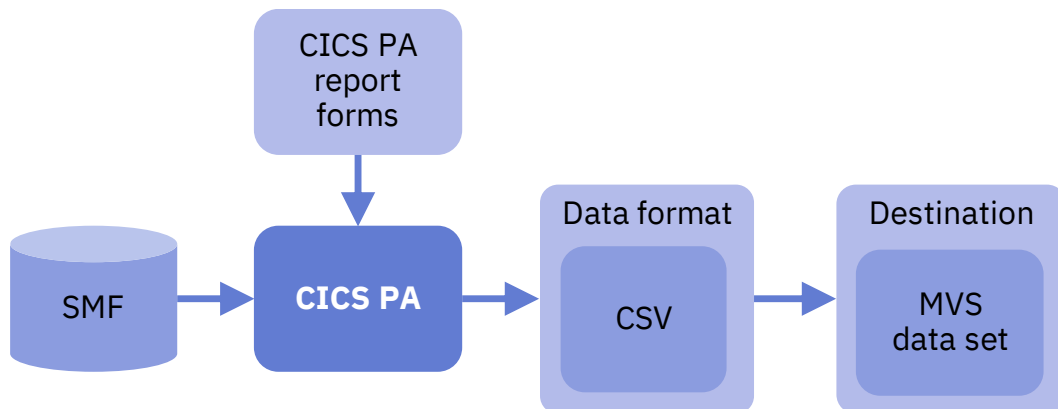
Figure 1. Existing functionality: CSV to MVS data set

To output CSV from CICS Performance Analyzer:

1. Create a report form, or use a supplied sample form, to describe the data that you want to extract.

2. Run a report set that writes CSV to an MVS data set.

Then, outside of CICS Performance Analyzer:

3. If required, transfer the data off z/OS. For example, using FTP.

4. Load the data into your choice of analytics platform.

What's new

As before, in CICS Performance Analyzer:

1. Use report forms to describe the data that you want to extract. You might use forms that you already use today.

2. Optionally, specify and activate z/OS Connect EE extract or Forwarding definitions in the report set

3. Run a report set.
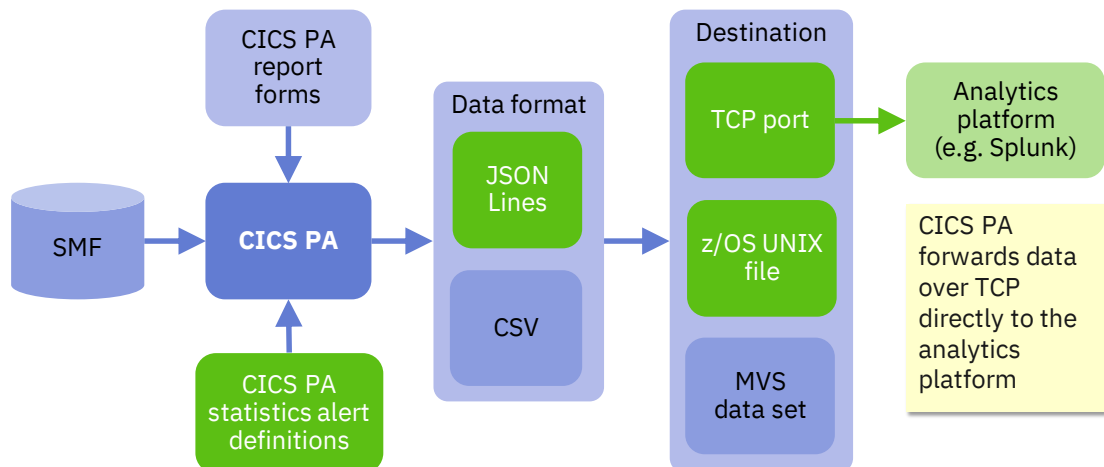
Figure 2. What's new: outputting JSON Lines, forwarding over TCP, writing to z/OS UNIX

New:

- CICS Performance Analyzer writes the data in JSON Lines format to the destination specified by the report set:
  - TCP port. For example, Splunk configured to listen for incoming data on a TCP port.
  - z/OS UNIX file.

The CICS Performance Analyzer ISPF dialog now allows you to specify a z/OS UNIX file path as the output destination, and then generates the appropriate JCL DD statement for you.

- MVS data set.

- In addition to writing *form-based output* to JSON Lines, the enhancements also enable you to write *statistics alerts* to JSON Lines. Previously, you could only write statistics alerts to a report.

- You can also write z/OS Connect EE SMF 123 records together with their associated CMF record to JSON Lines.

- Time values can be output as a single ISO 8601 date and time of day representation, optionally with a zone designator.

What's new for CSV and report output

Some enhancements introduced for JSON Lines also apply to CSV output. For example:

- You can write statistics alerts to CSV

- You can write z/OS Connect EE SMF 123 records to CSV

- You can extract CSV output to a z/OS UNIX file

- You can forward CSV output to a TCP port

Some enhancements introduced for JSON Lines also apply to both CSV *and* report output. For example:

- Time stamps can be output as a single ISO 8601 date and time of day representation, optionally with a zone designator

This document focusses on output to JSON Lines rather than CSV or report output.

Splunk app that visualizes data from CICS Performance Analyzer

IBM supplies a sample Splunk app for CICS Performance Analyzer that:

- Configures Splunk to ingest JSON Lines data from CICS Performance Analyzer
- Contains sample dashboards that demonstrate how to visualize that data

Here is a screenshot (this might not match the latest version of the app):
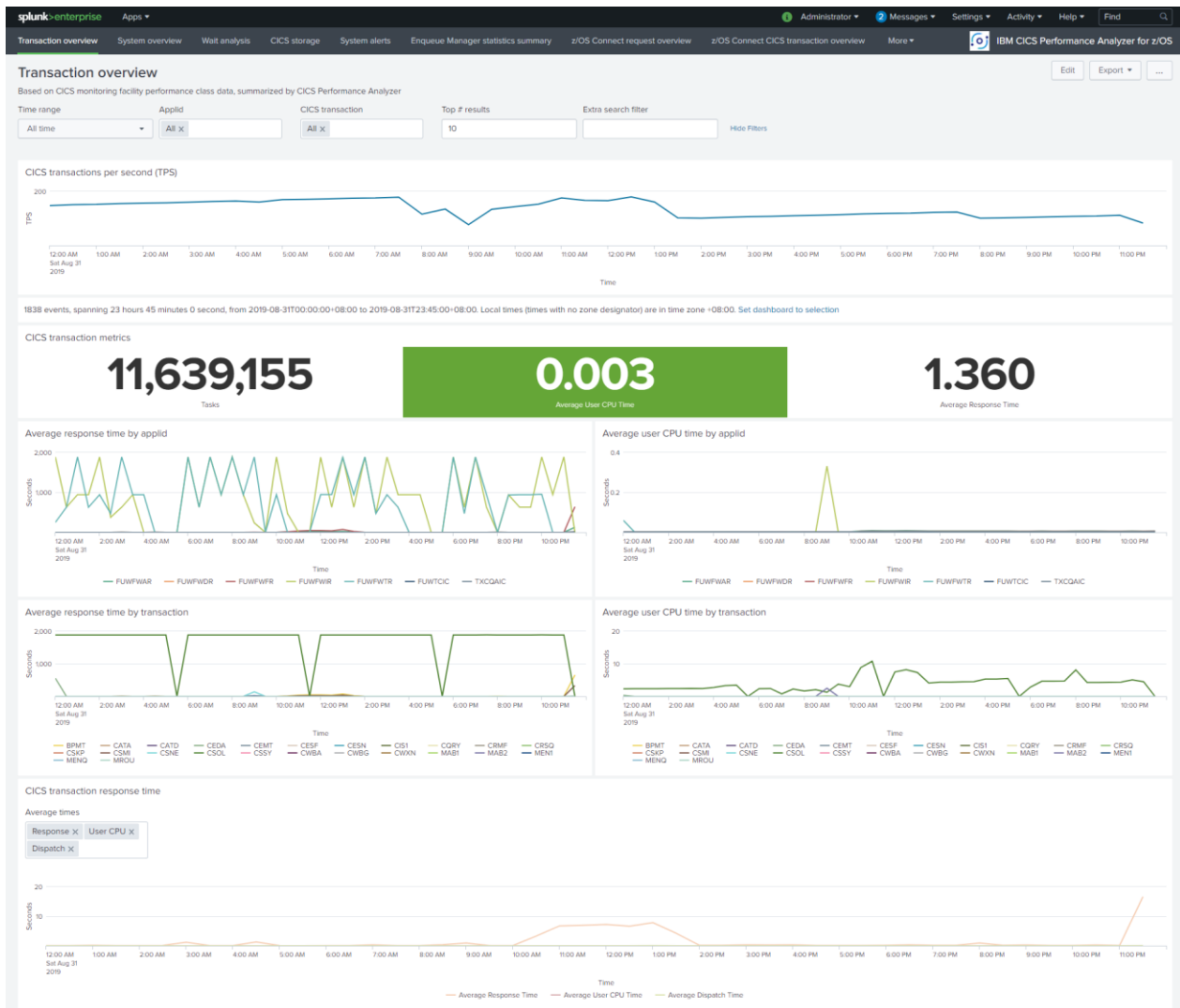
Figure 3. IBM-supplied sample Splunk app for CICS Performance Analyzer

You do not need a z/OS system to try out this app. IBM supplies sample JSON Lines data for use with the app, so that you can use the app without forwarding your own data from CICS Performance Analyzer.

If you want to understand the app in more detail before you install it, see "Splunk configuration by the IBM-supplied app" on page 58.

Scope and intended use of the app

The enhancements for output to JSON Lines respond to customer requests for enhancements (RFEs). These RFEs request JSON. They do *not* request a fully-fledged Splunk app for analyzing CICS performance.

The app is not intended to be a fully-fledged out-of-the-box solution for analyzing CICS performance in Splunk. Instead, the app contains sample dashboards that demonstrate some example use cases for visualizing data from CICS Performance Analyzer.

The developers of the IBM-supplied app anticipate that customers will examine these sample dashboards, and then perhaps copy and adapt selected visualizations into their own bespoke Splunk apps to match their own specific requirements.

Getting the app and sample data

You can download the app from the Splunkbase website: see the "IBM CICS Performance Analyzer for z/OS Sample App for Splunk".

Tip: You can use Splunk Web to find and install the app without downloading it first.

The sample data is available on the web in CICS SupportPac "CA10: CICS Performance Analyzer for z/OS - Output to JSON Lines". The SupportPac also contains this documentation.

Alternatively, contact your IBM Software representative for the app and sample data. You should receive a .tar.gz file and several .jsonl files, possibly packaged together in a single `.zip` file.

- `.tar.gz`
  Splunk app, packaged in a gzipped tarball. You might receive this file with an extension such as `.spl` or `.tgz`.

- `.jsonl`
  Sample data in JSON Lines format for use with the app.

Prerequisite: Splunk

To install the Splunk app, **you need Splunk**.

The app and this documentation were developed using Splunk **7.3.0**.

By convention, minor and patch versions should not introduce breaking changes, so the app *should* work in any Splunk 7.*minor.patch* version from Splunk 7.3.0 onwards.

The app *might* also work in earlier versions. Using the app in earlier versions is your choice.

It is recommended that you install the app in a "sandbox" Splunk environment, not your organization's production Splunk environment.

For instructions on getting and installing Splunk, see the Splunk website.

For example, you could:

- Install Splunk directly on your own personal computer.

- If you have Docker, start a container running Splunk.

Splunk Sankey Diagram – Custom Visualization app

The Sankey Diagram app is built by the Splunk developers (Splunk Inc.). The sample dashboard described in this document includes a sankey diagram visualization that relies on this app.

You can install this app directly from Splunk Web. Select Apps > Manage Apps > Click the Browse more apps button, and then search for "Sankey".

The sample dashboard was developed using Sankey Diagram app 1.5.0.

Important: After installing the Sankey Diagram app, set its sharing permissions to Global, to other apps to include sankey diagram visualizations. Select Apps > Manage Apps > Sankey Diagram > Change sharing permission to Global (not App).

Running Splunk in a Docker container

If you have Docker, then you can use the following `docker run` command to start a container running Splunk:

```
docker run -d -e "SPLUNK_PASSWORD=changeme" -e "SPLUNK_START_ARGS=--accept-
license" -p 18000:8000 -p 18089:8089 -p 11516:1516 --name cicspa-splunk-
demo splunk/splunk:7.3.0
```

The `-p` command options map ports inside the container to the following ports on your Docker host:

- Splunk listens on port 11516 for incoming JSON Lines from CICS Performance Analyzer.

    Inside the container, the corresponding TCP input definition in the Splunk `inputs.conf` file is set to listen on port 1516.

- Splunk Web is on port 18000.

- The splunkd management port is 18089.

- 7.3.0 is the Splunk version.

Installing the app

To install the Splunk app:

1. On the Splunk Web home page, click the cog icon ("Manage Apps") next to the **Apps** heading at the top left of the page:
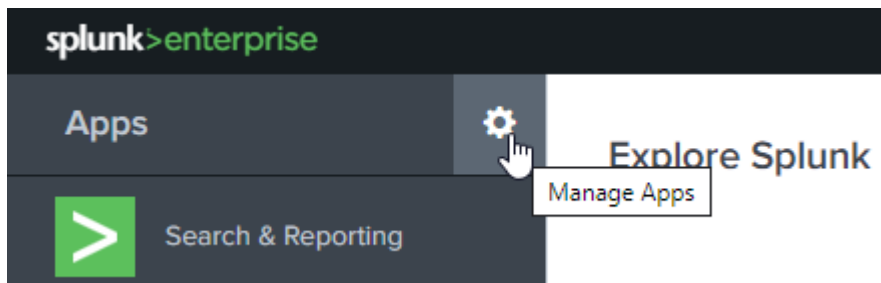
2. At the top right of the Apps page:

   – If you want to install the app from a `.tar.gz` or `.spl` file that is on your computer, click **Install app from file**. Follow the on-screen instructions to upload the app.

   – Otherwise, click **Browse more apps**, and find the "IBM CICS Performance Analyzer for z/OS Sample App for Splunk".

After installing the app, either:

- Upload the IBM-supplied sample data
- Forward or upload your own data

Uploading the sample data

To upload the IBM-supplied sample data to Splunk from a JSON Lines (`.jsonl`) file that is on your computer:

1. You must have already installed the corresponding IBM-supplied Splunk app. This app configures Splunk to correctly ingest the sample data.

2. If you have previously uploaded sample data for the app, and you have now received a newer version of the app with updated sample data: remove all data from ("clean") the `cicspa` index. Cleaning the index avoids uploading duplicate data. Note that, if you have added your own data to this index, cleaning the index will also remove that data.

For example, with admin-level privileges, change to the Splunk `bin` folder ("C:\Program Files\Splunk\bin" default location for windows) and enter the following commands:

```
splunk stop
splunk clean eventdata -index cicspa
splunk start
```

For more information on this procedure, see the Splunk docs heading "Remove all data from one or all indexes".

3. If you received the `.jsonl` file of sample data compressed inside a `.zip` file, extract the `.jsonl` file.

4. From the Splunk Web **Settings** menu (at the top right of the window), select **Add Data** (in the left-hand side bar of the menu dropdown).

5. Click **Upload** ("files from my computer").

6. Click **Select File**.

7. Navigate to the folder on your computer (or LAN) containing the `.jsonl` file, and then select the file.

8. Click the **Next** button at the top of the window.

9. Click **Source type** > **Application** > **cicspa**.

10. Click **Next**.

11. On the **Input Settings** panel:

    a. Set the **Host field value** to whatever you want. Suggestion: select **Constant value**, and then specify the name of your computer, or any string value ("`CICS-DEMO`").

    b. Set the **Index** to **cicspa**.

12. Click **Review**.

13. Click **Submit**.

Splunk displays a message to indicate the file has been uploaded successfully.

If you click **Start Searching**, then the search that Splunk generates should find *no results*. This is because the generated search includes the condition `sourcetype="cicspa"`. The configuration in the IBM-supplied app transforms that original "cicspa" `sourcetype` value into a value that includes the `code` property from the sample JSON Lines data. If that configuration worked, then none of the indexed events should match the condition

`sourcetype="cicspa"`. To find results, append an asterisk (*) to the condition value (`sourcetype="cicspa*"`), and then press Enter to perform the search.

Using the app

After installing the app and uploading the sample data, you can view the sample data in the app.

In the **App** dropdown selection list at the top-left of the window, select **IBM CICS Performance Analyzer for z/OS**:
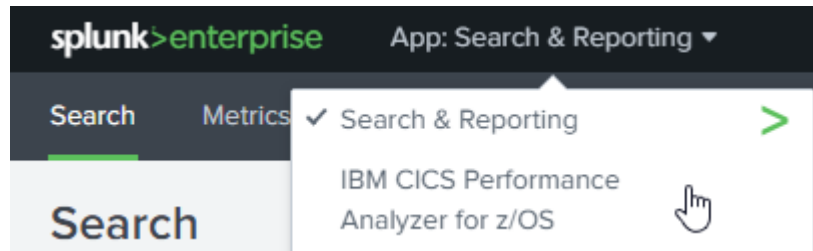
The app nav bar displays links to each of the sample dashboards.

The default time range of each dashboard is "Last 24 hours". **To view the sample data, set the time range to "Presets > Other > All time":**
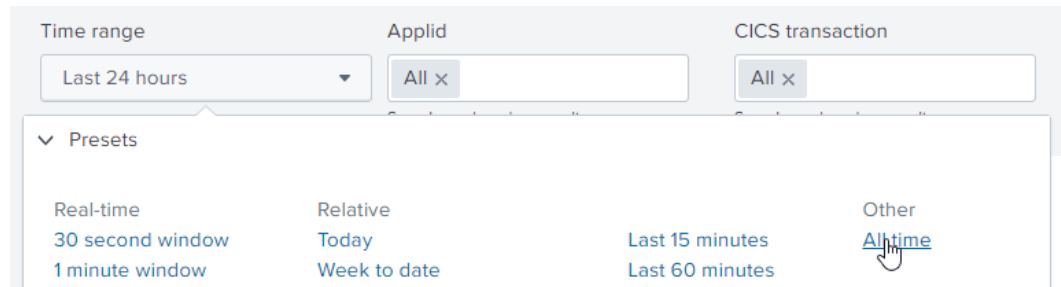
Dashboards

The following table lists the sample dashboards in the app (at the time of writing: later versions of the app might have more or different dashboards) and the types of data they visualize:

Table 1. Sample dashboards in the IBM-supplied Splunk app

Dashboard title	Data type
Transaction overview	Performance summary
System overview	Statistics list

Dashboard title	Data type
Wait analysis	Performance summary (using fields specific to wait analysis)
CICS storage	Statistics list
System alerts	Statistics alerts
Enqueue Manager statistics summary	Performance summary
z/OS Connect request overview	z/OS Connect request summary
z/OS Connect CICS transaction overview	Performance summary

While it's straightforward to mix multiple data types in a single dashboard, each of the current sample dashboards typically focusses on a single data type.

To understand the dashboards in detail, review the underlying Splunk Search Processing Language (SPL) and Simple XML. For more information about developing dashboards in your own app, see the Splunk docs.

Zooming and panning within a dashboard time range

At the top of each dashboard is a set of filter controls, including a time picker. The time picker sets the dashboard time range. The default range is "All time".

Below the filters, the first visualization in each dashboard is a time-based chart. The time range of this chart is set by the time picker in the filter controls. The data in this chart depends on the dashboard.

These time-based charts at the top of each dashboard have a common feature: selecting a period of time within the chart sets the time range for all other visualizations.

To select a period of time within the chart, "marquee-select" it: hold down your mouse button at the start time, drag to the end time, and then release the mouse button:
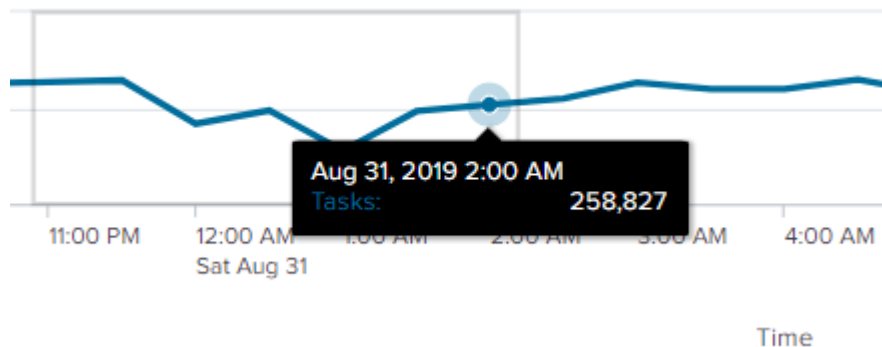
Figure 4. Marquee-selecting a time period in a chart

The time outside the selection is grayed-out and handles appear at the selection start and end times:
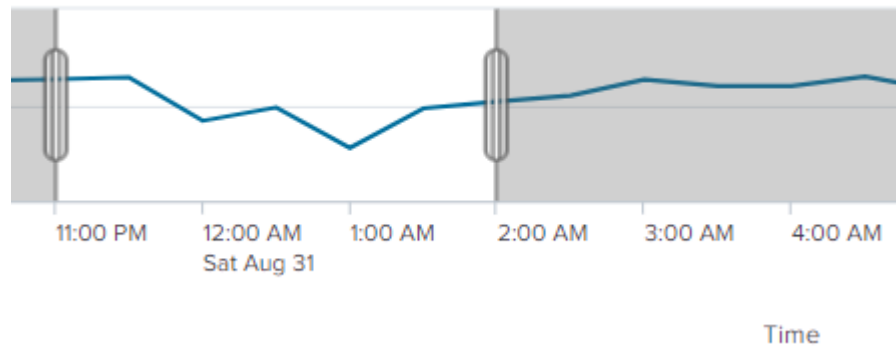
Figure 5. A selected time period in a chart

The visualizations below the chart update to reflect the selection.

To clear the selection, click the **Reset** button in the chart.

To change the start or end time of the selection, drag the selection handles.

To pan (move) the selection, drag the selection left or right.

To set the time range of the dashboard time picker to the selection (zoom in), click the **Set dashboard to selection** link below the chart.

To return to a previous time range (zoom out), you might need to click your web browser's **Back** button more than once.

Filtering groups of CICS applids and transactions

The filter controls at the top of each dashboard include multiselect dropdowns for CICS applid and, where applicable, transaction. The default value for these filters is "All" (show all values):

You can select multiple values in each control:

Filter values are reflected in the URL, so you can bookmark them in your web browser. Such bookmarks are especially useful if you also set a relative time range in the time picker, such as "Week to date".

Filtering to improve app performance

By default, the dashboards show all applids and transactions. If your data contains many applids and transactions, then, to improve performance, consider specifying filters to show only a selection of applids and transactions.

Extra search filter

To specify more complex filters for grouping CICS applids or transactions, use the **Extra search filter** text box in the filter controls. This text box accepts SPL syntax that gets appended—ANDed—to the search generated by the other filter controls. This text box enables you to specify more complex filters without taking the next step of developing your own dashboards with custom filters.

For example, if you have specified "All" in the CICS transaction dropdown selection box, then you can restrict the transactions shown by entering the following SPL in the **Extra search filter** text box:

```
NOT Tran IN (CSOL, CEMT, CEDA, CSNE)
```

You can use wildcards to specify values:

```
APPLID=FUW*
```

or lists of values:

```
Tran IN (M*, B*, CSMI)
```

Tip: In Splunk, field names are case-sensitive.

For filters that require long lists of values, consider using a lookup table. The app contains an example macro, `cicspa_exclude_trans`, that uses a lookup table to exclude CICS-supplied transactions. To refer to a macro in the **Extra search filter** text box, enclose the macro name in back ticks (`` ` ``):

Extra search filter

`cicspa_exclude_trans`

Here is the macro definition:

```
NOT [|inputlookup cics_trans.csv]
```

The `cics_trans.csv` file contains the field name `Tran` followed by a list of CICS transactions, including CICS-supplied transactions. This file is supplied in the app in the following folder:

Path to your Splunk directory`/etc/apps/cicspa/lookups`

Sankey diagram of applids vs enqueue pool IDs

Each event in the summarized CICS Enqueue Manager statistics contains the text fields "APPLID" and "ENQ Pool ID".

The Enqueue Manager statistics summary dashboard contains a sankey diagram that shows combinations of applids and enqueue pool IDs:

Figure 6. Sankey diagram (source: applid, target: enqueue pool ID)

Use the **Measure** dropdown to select which numeric field the diagram shows. The diagram color-codes values: green indicates low values down to zero, while red indicates the highest values.

The **Zero values** radio buttons control how the diagram treats zero values:

Exclude

> Excludes zero values. This is the default behavior. This option excludes combinations of applid and enqueue pool ID for which the value of the selected measure is zero. For example, if you select the measure "ENQs Issued" with this option, then the diagram shows only those combinations for which enqueues *were* issued.

Include

> Includes zero values. This option shows *all* combinations of applid and enqueue pool ID, *including* combinations for which the value of the selected measure is zero.

Only

> Shows *only* zero values. This shows only those combinations of applid and enqueue pool ID for which the value of the selected measure is zero. You can use this option to answer questions such as "Which combinations of applid and enqueue pool ID had no enqueue waits?"

For the **Exclude** and **Only** options, the fixed legend label ">= 0" is inaccurate; the label should be "> 0" or "= 0". For details, see the Splunk Answers discussion topic "Override ">= 0" legend label in sankey diagram?". (Related discussion topic also prompted by the development of this dashboard: "Sankey diagram drilldown to set form tokens?".)

Drilldown in z/OS Connect dashboards

The z/OS Connect dashboards have a table that enables the user to move between the two dashboards. Clicking on any row of the table in the "z/OS Connect request overview" dashboard links the user to the "z/OS Connect CICS transaction overview" dashboard. The transactions are filtered based on the tracking token of the selected row. This behavior also applies vice versa.

z/OS Connect URL Parameter

`form.zctoken=<value>`
> Parameter used by z/OS Connect dashboards. Filters the data based on the provided z/OS Connect tracking token value.

Persisting filters between dashboards and sessions

By default, Splunk Web does not preserve your filter values when you switch between dashboards in an app. For example, the time range that you set in one dashboard is not kept when you switch to another dashboard.

However, the IBM-supplied Splunk app for CICS Performance Analyzer contains custom JavaScript that persists filters when you switch between most dashboards in the app, and also in subsequent browser sessions, excluding z/OS Connect dashboards.

URL parameters to override saved filters

The custom JavaScript introduces two new parameters to the query string of dashboard URLs:

`form.set_all_filters=y`

> Sets all filter values from the URL, rather than from previous values saved in your browser's local storage.

`form.set_time_filter=y`

> Sets only the time range from the URL; restores all other filters from previous values saved in your browser's local storage. The app uses this parameter when "zooming in" to a narrower selected time range for the same dashboard.

In typical use, you don't need to know about these parameters. The app simply remembers your filters. However, if you want to understand your Splunk Web URLs in detail, then it's useful to understand these parameters.

Tip: To reset filters to the default values defined by a dashboard, enter the dashboard URL in your browser with the query string `?form.set_all_filters=y` and no other parameters. For example:

```
http://my.splunk.host:8000/en-US/app/cicspa/
overview_tran?form.set_all_filters=y
```

Tip for viewing the sample data: To set the time range to "All time", but reset all other filters, include `earliest=0` in the query string of the URL. For example:

```
http://my.splunk.host:8000/en-US/app/cicspa/
overview_tran?earliest=0&form.set_all_filters=y
```

Custom JavaScript for persisting filters

The custom JavaScript for persisting filters is supplied in the following file:

Path to your Splunk directory`/etc/apps/cicspa/appserver/static/persist_filters.js`

About the sample data

The IBM-supplied sample data was generated in-house by the CICS Performance Analyzer development team using simulated workloads on test CICS systems.

Any similarity between identifiers in the sample data, such as CICS transaction identifiers, and identifiers in real-life production data are entirely coincidental.

The sample data does not include any personal data.

Duration

The sample data covers a period of 24 hours, from midnight to midnight, UTC+8.

Summarization interval

For the sample data for the Enqueue manager statistics summary dashboard, the CICS Performance Analyzer summarization reporting interval was set to **3 hours**.

For other sample data, the following intervals were both set to 15 minutes:

- CICS statistics collection interval

- CICS Performance Analyzer summarization reporting interval, used in performance summary report forms and statistics summary report forms

Sample data types, `code` property values, and report form names

The sample data consists of multiple data types: data generated using multiple report forms, and statistics alert definitions. Each data type has a unique `code` property value. The `code` property is the first property in each line of JSON Lines.

The sample data uses `code` property values that are compatible with the IBM-supplied Splunk app. The app refers to source types that correspond to those `code` property values.

The following table lists the `code` property values in the sample data:

Table 2. `code` property values of the IBM-supplied sample data

Value (see Note)	Description	Data type
`TRANLIST`	General-purpose transaction performance list data	Form-based performance list
`TRANSUM`	General-purpose transaction performance summary data	Form-based performance summary
`TRANWAIT`	As per `TRANSUM`, with additional fields specific to wait analysis	
`CONECTIO`	ISC/MRO connections statistics, ID 052A	Form-based statistics list
`DB2CONNS`	Db2 connections statistics, ID 102A	
`DB2ENTRY`	Db2 entries statistics, ID 103A	
`DISPOVER`	Dispatcher overview statistics, ID 062A	
`DISPTCBM`	Dispatcher TCB modes statistics, 062B	
`DISPTCBP`	Dispatcher TCB pools statistics, ID 062C	

Value (see Note)	Description	Data type
DSA	Dynamic storage areas (DSAs) statistics, ID 029B	
ENQMANAG	Enqueue manager statistics, ID 097A	
ENQMGSUM	Enqueue manager statistics summary	
LOGSTREM	Log stream overview statistics, ID 092A	
MVSLOGST	MVS log streams statistics, ID 094A	
STORAGE	Storage manager statistics, ID 029A	
SYSDUMPS	System dump overview statistics, ID 090A	
TDQUEUES	Transient data overview statistics, ID 045A	
TRANCLAS	Transaction classes statistics, ID 012A	
TRANDUMP	Transaction dump overview statistics, ID 087A	
TRNMANAG	Transaction manager statistics, ID 010A	
TSQUEUES	Temporary storage overview statistics, ID 048A	
SYSALERT	From the IBM-supplied sample system alerts definitions named CTSKEY	Statistics alerts
ZCON_REQWAIT	z/OS Connect EE requests statistics.	z/OS Connect request summary
ZCON_TRANWAIT	General-purpose CICS transaction performance data related to a z/OS Connect EE request.	Form-based performance summary

Note: The code property values in the IBM-supplied sample data correspond with IBM-supplied sample report forms and statistics alert and z/OS Connect EE definitions:

- For form-based output, the code property value matches the name of a supplied sample report form.

- The code property value SYSALERT corresponds to the supplied sample statistics alert definition CTSKEY.

- For more details, see "Sample report set for using your own data in the Splunk app" on page 26.

To review the code property values in the sample data, open the sample data .jsonl file in a text editor.

To review the corresponding source types in Splunk Web using the default "Search & Reporting" app, after uploading the sample data:

1. Set the time range picker to the preset time range **Other** > **All time**.

2. Submit the following search:

```
index=cicspa | stats count by sourcetype
```

For a description of the relationship between code property values in JSON Lines and Splunk source types, see "Configuring Splunk for JSON Lines with or without a code property" on page 62.

Viewing the sample data in the default Splunk Search app

You can view most of the sample data in dashboards in the IBM-supplied sample Splunk app. You can also choose to view the sample data in the default Splunk "Search & Reporting" app.

For example, to view the sample CICS transaction performance list data (there is currently no dashboard in the IBM-supplied app for this data), enter the following search:

```
index=cicspa sourcetype=cicspa_TRANLIST "Response Time">1.5
```

Remember to set an appropriate time range, such as **Other** > **All time**.

Using the sample data in analytics platforms other than Splunk

The IBM-supplied sample data works with the IBM-supplied Splunk app for CICS Performance Analyzer. The app *requires* the particular data types—code property values—in the sample data.

However, there's nothing Splunk-specific about the sample data. You can configure other analytics platforms to ingest the same data. For example, see "Configuring the Elastic Stack to use JSON Lines from CICS Performance Analyzer" on page 91.

Sample report set for using your own data in the Splunk app

The CICS Performance Analyzer sample library member CPARSSAM contains a sample report set named ANALYTIC, and corresponding report forms, statistics alert and z/OS Connect EE definition, to output JSON Lines that works with the IBM-supplied Splunk app. The names of the report forms and statistics alert definition are listed in "Sample data types, code property values, and report form names" on page 24.

For instructions on extracting the sample report set, report forms, and statistics alert definition from that member, see the following topic in IBM Knowledge Center:

Before submitting the ANALYTIC report set:

- If you have added the IBM-supplied sample data to the Splunk instance that is running the app, consider removing the sample data from the `cicspa` index before adding your own data. For details, see the Splunk docs heading "Remove all data from one or all indexes".

- You need to edit the corresponding items under the Extracts and Forwarding categories in the report set to specify your site-specific details. For details, see "Using the ISPF dialog to generate and submit report set JCL" on page 28.

- It is expected that you will use either Extracts or Forwarding to create the JSON Lines. However, if you choose to activate the corresponding definitions in both Extract and Forwarding, please ensure that the Recap DDName in each corresponding definition is unique, otherwise errors may occur.

In case you notice that fields requested by a form are missing from the JSON Lines output: the ANALYTIC report set specifies the parameter `MISSING(EXCLUDE)`, which excludes from the JSON Lines output any fields that are missing from the input SMF records.

The sample report forms for output to JSON Lines are based on the fields in CICS Transaction Server 5.5. If you have an earlier CICS TS release, then some of the fields in the forms might be missing from the JSON Lines that you generate. These missing fields might result in empty visualizations or errors in the Splunk dashboards. If you have a *later* CICS TS release, then you might want to update the sample searches to add new fields; for example, new wait analysis components.

Using CICS Performance Analyzer to output JSON Lines

You can use the CICS Performance Analyzer ISPF dialog to generate JCL that outputs JSON Lines.

To output JSON Lines that works with the IBM-supplied Splunk app, use the IBM-supplied sample report set ANALYTIC. For details on installing that sample report set, see "Sample report set for using your own data in the Splunk app" on page 26.

Using the ISPF dialog to generate and submit report set JCL

To **extract** JSON Lines to an MVS data set or a z/OS UNIX file, select the Performance, Statistics or z/OS Connect EE report under the Extracts category in a report set.

To **forward** JSON Lines over a TCP network, select the Performance, Statistics or z/OS Connect EE report under the new Forwarding category.

You can choose to use only one of these two methods—extract or forward—and activate only those reports in a report set. Or you can choose to extract *and* forward in the same report set. For example, you might want to extract some JSON Lines to a z/OS UNIX file for local processing *and* forward JSON Lines off z/OS to a remote analytics platform. **Note:** if you choose to activate both extract and forwarding of corresponding definitions, please ensure that the Recap DDName in each corresponding definition is unique, otherwise errors may occur.

Summary of changes to the ISPF dialog

New items shown highlighted:

- New: ISPF dialog option 0.6, **Connection Settings**, specifies the details of a destination for forwarding data over TCP.

  These settings are only required if you intend to *forward* data over TCP. If you intend to only *extract* data to an MVS data set or a z/OS UNIX file, then you do not need to specify these settings.

- New: Time stamp fields in performance list and performance summary report forms (under ISPF dialog option 3, Report Forms) now support the new format ISO8601.

- Report set tree (under ISPF dialog option 2, Report Sets):

  – **Extracts** category:

    - **Performance   Statistics** and **z/OS Connect** reports:

      – New: write to either JSON Lines or CSV (was CSV only)

- New: write to either z/OS UNIX files or MVS data sets (was MVS data sets only)

- New: The **Statistics** report panels now support writing either statistics alerts or form-based statistics extracts (was form-based statistics extracts only)

- New: The **Forwarding** category enables you to *forward* JSON Lines or CSV output over TCP, as opposed to *extracting* to an MVS data set or a z/OS UNIX file. The Performance, Statistics and z/OS Connect report panels under this new category are similar to the Performance and Statistics panels under the Extracts category, except that the Forwarding panels use the destination details specified by the new Connection Settings (option 0.6).

Extracts and Forwarding categories in a report set

The following report set highlights the reports that output to JSON Lines: the existing reports, and the new Extract and Forwarding reports.

```
EDIT                    Report Set - ANALYTIC                    Row 1 of 18
Command ===> _____     Scroll ===> CSR

Description . . . Analytics Sample Report Set

Enter "/" to select action.


___         ** Reports **                          Active
   -  ___   Extracts                                Yes
      ___      Cross-System Work                    No
      ___      Performance                          Yes
      ___      Record Selection                     No
      ___      HDB Load                             No
      ___      System Logger                        No
      ___      Statistics                           Yes
      __  z/OS Connect EE                           Yes
   -  ___   Forwarding                              No
      ___      Performance                          No
      ___      Statistics                           No
      __  z/OS Connect EE                           No


         ** End of Reports **
```

Figure 7. CICS Performance Analyzer ISPF dialog: report set categories and reports that output to JSON Lines

Extracting JSON Lines to an MVS data set or a z/OS UNIX file

Generating JCL that writes JSON Lines to an MVS data set or a z/OS UNIX file is similar to the existing procedure for writing CSV extracts: you select Performance, Statistics or z/OS Connect under the Extracts category of a report set.

The changes:

- Previously, you could only use these Extracts panels to write CSV to an MVS data set. You can now write either CSV or JSON Lines to either an MVS data set or a z/OS UNIX file.

- There is now a choice of extract format: **JSON** (specifically, JSON Lines) or **CSV**.

- Select **Options** to show the new pop-up window for CSV and JSON options.

The following screen captures highlight these changes.

Changes to the Performance Extract panel:

```
                         ANALYTIC - Performance Extract
 Command ===> _____

 System Selection:                      Extract Recap:
  APPLID . . _____    +                DDname   . . . _____
  Image  . . _____    +
  Group  . . _____    +

 Data Output:
  Data Set Name  . . _____
  Disposition  . . . _   1. OLD   2. MOD   (If cataloged)
  z/OS UNIX File . .    _____    +

 Extract Focus:                         Summary Processing Options:
  Form  . . . . _____    +             Interval  . . . _____   (hh:mm:ss)
  Alert . . . . _____    +             Override Form   _____   +
   Severity . . _____    +             Timestamp . . . _____   +

 Extract Format:
  _  1. CSV   2. JSON                 _     Options

 Selection Criteria:                    Execution Options:
  _    Performance                      _   Use External Sort

 Repository  . . : CPA.JSON.RSSAMP.REPOSTRY
```

Figure 8. CICS Performance Analyzer ISPF dialog: Performance Extract panel

Changes to the Statistics Extract panel:

```
                        ANALYTIC - Statistics Extract
Command ===> _____

System Selection:                        Extract Recap:
 APPLID . . _____  +                       DDname   . . . _____
 Image  . . _____  +
 Group  . . _____  +

Data Output:
 Data Set Name Prefix . . _____
 Disposition  . . . . . . _   1. OLD   2. MOD   (If cataloged)
 z/OS UNIX File . .  _____       +

Extract Focus:
 _   Statistics Reports              Form . . . _____   +

Selection Criteria or Alert Extract:   Summary Options:
 Alert  . . _____  +                   Interval . . _____
  Severity _____   +
  _   Include Severity column
 Type . . /  EOD  /  INT  /  USS  /  REQ  /  RRT

Extract Format:                          Execution Option:
 _   1. CSV   2. JSON                      _   Use External Sort
 _   Options

Repository  . . : CPA.JSON.RSSAMP.REPOSTRY
```

Figure 9. CICS Performance Analyzer ISPF dialog: Statistics Extract panel

On the Statistics Extract panel:

- The MVS data set name is characterized as a *prefix*. However, when writing to a z/OS UNIX file path, CICS Performance Analyzer uses the path as-is: no suffix is appended.

- To extract statistics alerts instead of form-based statistics data, see "Extracting or forwarding statistics alerts" on page 36.

```
                         ANALYTIC – z/OS Connect EE Extract
  Command ===>  _____

  System Selection:                       Extract Recap:
   APPLID . .  _____   +                 DDname   . . .  _____
   Image   . .  _____   +
   Group   . .  _____   +

  ZCON System Selection:                  Extract Options:
   SSID . . .  _____   +                _  1. CICS Service Providers only
   Image   . .  _____   +                  2. All Service Providers
   Group   . .  _____   +                / Include ZCON request CICS tasks

  Data Output:
   Data Set Name Prefix . .  _____
   Disposition  . . . . . .  _  1. OLD   2. MOD   (If cataloged)
   z/OS UNIX File . .  _____     +

  Extract Required:                       Extract Focus:
   _  1. List   2. Summary                 Form . . .  _____   +

  Extract Format:                         List Option:
   _  1. CSV   2. JSON                      _  Include ZC Header fields
   _  Options

  Summary Options:
   ZC Key  . .1  . .  _____   +  2  . .  _____   + 3  . .  _____   +
   Interval   . . .  _____  (hh:mm:ss)
   _  Include Maximum value Request details

  Selection Criteria:
   _  Performance
```

Figure 10. CICS Performance Analyzer ISPF dialog: New z/OS Connect EE Extract panel

On the z/OS Connect Extract panel:

- The MVS data set name and z/OS UNIX File is characterized as a *prefix.* Depending on the *Extract Option* selected, 1 or 2 files are created, one containing z/OS Connect request records and appended with "ZCN", and the other containing the related Performance records and appended with "CMF".

The new Options pop-up window includes several new options, most of which apply to both JSON and CSV:

```
                                  Options
Command ===> _____

Code . . . . . . _____

/   Time Stamp Fields in ISO8601 Format
OutZone   . . . _____      (Z for UTC or -12:00 to +12:00)
Encoding . . . 1  1. EBCDIC    2. ASCII
EOL   . . . . . _  1. LF        2. CR     3. NL     4. CRLF
Label Format   1  1. Standard  2. CICS   3. Short
Label Case . . 1  1. As is     2. Upper  3. Lower

CSV only:
 Delimiter . . ;
 /  Include Field Labels

JSON only:
 /   Exclude Missing Fields
```

Figure 11. CICS Performance Analyzer ISPF dialog: Pop-up window for CSV and JSON options

The new options correspond to new `CICSPA` batch parameters:

New CSV and JSON options in the ISPF dialog	CICSPA batch parameter
Code	`CODE`
Time Stamp Fields in ISO8601 Format	`TIMEFORMAT(ISO8601)`
OutZone	`OUTZONE`
Encoding	`EBCDIC|ASCII`
EOL	`EOL`
Label Format (statistics only) Label Case	`LABELS`
Exclude Missing Fields (JSON only)	`MISSING`
CMF Code	`CODECMF`
ZCON Code	`CODEZCON`

For details, see the documentation of the corresponding batch syntax in "New `CICSPA` batch command syntax" on page 66.

Forwarding JSON Lines over a TCP network

The Performance, Statistics and z/OS Connect EE panels under the new Forwarding report set category are similar to the existing Performance and Statistics panels under the Extracts category.

For performance:

```
                    ANALYTIC - Performance Forwarding
  Command ===> _____

  System Selection:                      Forwarding Recap:
   APPLID . . _____    +                 DDname  . . . _____
   Image  . . _____    +
   Group  . . _____    +

  Forwarding Focus:                      Summary Processing Options:
   Form  . . . . _____    +             Interval  . . . _____    (hh:mm:ss)
   Alert . . . . _____    +             Override Form   _____    +
    Severity . . _____    +             Timestamp . . . _____    +

  Forwarding Format:                     Connection Settings Override
   _   1. CSV    2. JSON                   Port  . . . . . . _____
   _   Options

  Selection Criteria:
   _   Performance

  Repository  . . : CPA.JSON.RSSAMP.REPOSTRY
```

Figure 12. CICS Performance Analyzer ISPF dialog: Statistics Extract panel

For statistics:

```
                    ANALYTIC - Statistics Forwarding
  Command ===> _____

  System Selection:                      Forwarding Recap:
   APPLID . . _____    +                 DDname  . . . _____
   Image  . . _____    +
   Group  . . _____    +

  Forwarding Focus:                      Connection Settings Override
   Form . . . _____    +                 Port  . . . . . . _____

  Selection Criteria or                  Summary Options:
  Alert Forwarding:                        Interval . . _____
   Alert  . . _____    +
    Severity _____    +
    _   Include Severity column
   Type . . /  EOD  /  INT  /  USS  /  REQ  /  RRT

  Forwarding Format:
   _   1. CSV    2. JSON                       _   Options

  Repository  . . : CPA.JSON.RSSAMP.REPOSTRY
```

Figure 13. CICS Performance Analyzer ISPF dialog: Statistics Extract

```
                      ANALYTIC – z/OS Connect EE Forwarding
  Command ===> _____

  System Selection:                     Forwarding Recap:
   APPLID . . _____   +               DDname  . . . _____
   Image  . . _____   +
   Group  . . _____   +

  ZCON System Selection:                Forwarding Options:
   SSID . . . _____   +              _    1. CICS Service Providers only
   Image  . . _____   +                   2. All Service Providers
   Group  . . _____   +              / Include ZCON request CICS tasks

  Forwarding  Required:                 Forwarding  Focus:
   _  1. List   2. Summary                Form . . . _____   +

  Forwarding Format:                    Connection Settings Override
   _  1. CSV   2. JSON                    Port . . . . . . . _____
      Options

  List Options:
   _   Include ZC Header fields

  Summary Options:
   ZC Key  . .1  . . _____   + 2  . . _____   + 3 . . _____   +
   Interval    . . . _____ (hh:mm:ss)
   _   Include Maximum value Request details

   Selection Criteria:
   _   Performance
```

Figure 14. CICS Performance Analyzer ISPF dialog: New z/OS Connect EE Forwarding panel

However, instead of specifying an output MVS data set name or a z/OS UNIX file path, the forwarding panels implicitly refer to your **Connection Settings** (option 0.6):

```
                      Connection Settings
  Command ===> _____


  Host . . . . . . . _____
  Port . . . . . . . _____  (1-65535)
  Timeout  . . . . . _____  (seconds)
  Security . . . . . _  1. None  2. TLS
    FIPS . . . . . . _  (Y/N)
    Key ring . . . . _____  +
    Stash file . . . _____  +
    Password . . . . _____  +
    Ciphers  . . . . _____  +
    Label  . . . . . _____
```

Figure 15. CICS Performance Analyzer ISPF dialog: new option 0.6, Connection Settings

These settings specify the host, port, and other details for forwarding over TCP.

These settings are not shared with other users; they are stored in each user's ISPF profile.

Depending on how you configure your listening analytics platform, you might want to forward each type of data—for example, data based on different reports—to different ports. The **Port** field on the forwarding panels overrides the port specified in your connection settings.

For more information about each connection setting, see the documentation of the corresponding batch syntax in "New `CICSPA` control operand: `CONNECTION`" on page 66.

Extracting or forwarding statistics alerts

Previously, you could only write statistics alerts to a report. That report is still available, unchanged, under the Statistics Reports category in the report set tree.

Now, you can output statistics alerts in JSON Lines or CSV format to an MVS data set, a z/OS UNIX file, or over a network to a TCP port.

To extract statistics alerts to an MVS data set or a z/OS UNIX file, select **Statistics** under the **Extracts** report set category.

To forward statistics alerts over a network to a TCP port, select **Statistics** under the **Forwarding** report set category.

In either case, on the Statistics Extract or Statistics Forwarding panel, to output statistics alerts instead of form-based statistics data:

1.   Leave the **Form** field blank.

2.   Specify the statistics alert definition in the **Alert** field under the **Selection Criteria or Alert...** heading.

Note: When extracting statistics alerts, no suffix is appended to either the MVS data set name or the z/OS UNIX file path.

If you specify *both* a form and an alert, then, as before, the alert is used as selection criteria for the form-based output.

Extracting or forwarding z/OS Connect requests statistics and related Performance data

Now, you can output z/OS Connect requests statistics and their related Performance data in JSON Lines or CSV format to an MVS data set, a z/OS UNIX file, or over a network to a TCP port.

To extract z/OS Connect requests statistics and their related Performance data to an MVS data set or a z/OS UNIX file, select **z/OS Connect EE** under the **Extracts** report set category.

To forward statistics alerts over a network to a TCP port, select **z/OS Connect EE** under the **Forwarding** report set category.

Tips for designing report forms for output to JSON Lines

When designing report forms for output to JSON Lines, consider the following tips:

Take care to generate JSON properties with unique names
See "Uniqueness of property names" on page 54.

Consider making the event time stamp the first time stamp field in the form
See "Event time stamps" on page 51.

Consider using one statistics summary report form instead of multiple statistics list report forms
A statistics summary report form can refer to fields from multiple CICS statistics record identifiers. By contrast, a statistics list report form can only refer to fields from a single CICS statistics record identifier.

Rather than defining separate statistics list report forms for output to JSON Lines, consider defining one, or a limited number, of statistics summary report forms that contain all of the statistics fields that you are interested in. Combining fields from multiple CICS statistics identifiers into a single line of JSON Lines avoids the effort of correlating and merging separate events in your analytics platform.

When designing a statistics summary report form that refers to fields from multiple CICS statistics identifiers, limit the summary key to fields that are common across those identifiers. For example: Collection Time, APPLID, Image.

Note: The optional `code` property is specified by the report *set*, not the *form*. This enables you to reuse report forms, unchanged, in situations where you do and do not want a `code` property.

Tips for defining statistics alerts for output to JSON Lines

Statistics alert descriptions become the value of the `Alert` property in the JSON Lines output. When describing an alert:

- Take care to only use characters that are successfully output to JSON Lines. For example, do not include quote (") characters in alert descriptions.

- Be concise. Especially if the pricing model of your analytics platform is linked to the number of bytes ingested.

In most cases, you deliberately define alerts so that the volume of alerts is low: that is, an alert is generated only when there is a real problem. However, you might choose to define some information-level alerts that are relatively high volume.

If the volume of alerts and the length of your alert descriptions becomes an issue, consider specifying a brief identifier as the alert description in CICS Performance Analyzer. When presenting data in your analytics platform, use a lookup table to replace the brief identifier with a longer description.

See:

-
-

Concatenating JSON Lines files for easier ingestion

A single CICS Performance Analyzer batch job can write JSON Lines to multiple MVS data sets or z/OS UNIX files. In fact, different `CICSPA` report operands in the same job step *cannot* write to the same MVS data set or z/OS UNIX file.

If you have multiple JSON Lines files with various `code` property values, then you can concatenate them into a single file for easier distribution and easier uploading in Splunk Web. Uploading a single large file in Splunk Web involves fewer mouse clicks than uploading several smaller files.

For example, suppose you have the following collection of MVS data sets, with a common naming pattern:

```
CPA.EXTRACT.JSONL.TRANLIST
CPA.EXTRACT.JSONL.TRANSUM
CPA.EXTRACT.JSONL.TRNMANAG
CPA.EXTRACT.JSONL.CTSKEY
…
```

You can transfer those data sets from z/OS to your personal computer with the following FTP client commands:

```
asc
prompt
cd 'CPA.EXTRACT.JSONL'
mget *
```

You can choose to append an appropriate file extension to the transferred files. For example, on Windows:

```
ren * *.jsonl
```

To concatenate the files (assuming there is no existing `all.jsonl` file in the current directory):

```
copy /b *.jsonl all.jsonl
```

Tip: If you omit the `/b` switch on the Windows `copy` command, the output file will end in a substitution character (X'1A'), which, when you upload the file to Splunk, will result in a bogus "empty" event.

Creating the corresponding report form in CICS Performance Analyzer for Enqueue Manager statistics

To create the report form in CICS Performance Analyzer for forwarding your own summarized Enqueue Manager statistics:

1. On the CICS Performance Analyzer primary option menu, select option 3, Report Forms:

```
V5R4M0          CICS Performance Analyzer - Primary Option Menu
Option ===> 3

0  CICS PA Profile      Customize your CICS PA dialog profile
1  Systems              Specify Systems, SMF Files, and Groups
2  Report Sets          Request and submit reports and extracts
3  Report Forms         Define and submit Report Forms
4  Object Lists         Define Object Lists
5  Historical Database  Collect and process historical data
6  Statistics           Report CICS Statistics
7  Profiling            Request Transaction Profiling
8  Resource Definitions Define CICS PA resources
X  Exit                 Terminate CICS PA
```

2. Enter NEW on the command line:

```
                              Report Forms              Row 1 to 32 of 69
Command ===> NEW                                        Scroll ===> PAGE
```

3. Select option 2, Statistics:

```
             New Report Form

Select an option then press Enter.
2  1. Performance
   2. Statistics
```

4. Specify the form name ENQMGSUM and the form type **Summary**:

```
                          New Statistics Report Form
Command ===>

Specify new Statistics Report Form options.

Name . . .  ENQMGSUM   Version (VRM) . . TS: 720  +  TG: 910  +

Form Type or Model:
2  1. List                    3. Model (Report Form)
   2. Summary

Model . . . . . . . . . .  _____   +
Report Forms Data Set . . 'CPA.JSON.FORM'                          +
```

5. On the statistics report tree, enter S next to **Regions** > **Enqueue Pools**:

```
EDIT                      EDIT Statistics Reports Form - ENQMG    Line 1 of 117
Command ===>                                               Scroll ===> CSR

Description   Statistics Summary Report Form        VRM TS: 720  TG: 910

____          ** Report **
   -   ___    Regions
         ___    Transaction Manager
         ___    Monitoring
      -  ___    CICS Dispatcher
            ___    Dispatcher Overview
            ___    Dispatcher TCB Modes
            ___    Dispatcher TCB Pools
            ___    MVS TCB Overview
            ___    MVS TCBs
      -  ___    CICS Storage
            ___    Storage Overview
            ___    DSAs
            ___    Domain Subpools
            ___    Task Subpools
      -  ___    CICS Dumps
            ___    Transaction Dump Overview
            ___    Transaction Dumps
            ___    System Dump Overview
            ___    System Dumps
         ___    Asynchronous Service
      S  ___    Enqueue Pools
         ___    BUNDLE Resources
         ___    POLICY Resources
```

6. Edit the default report form:

   –   Set the Description to "Analytics - Enqueues Summary".
   –   Change the O (Order) column value of the top four (key) fields to A (ascending).

```
                    EDIT Enqueue Pools - ENQMGSUM              Row 1 of 17 More: >
   Command ===> _____      Scroll ===> PAGE

   Description  . . . Analytics - Enqueues Summary        VRM TS: 720  TG: 910

                                                          Page width . : 211


       Field                                         Sort
   /   Heading +                                     K  O Fn   Limit
   ___ APPLID_____ K  A ___  _____
   ___ IMAGE_____ K  A ___  _____
   ___ COLLECTION TIME_____ K  A ___  _____
   ___ ENQ POOL ID_____ K  A ___  _____
   ___ ENQS ISSUED_____ _  TOT   _____
   ___ ENQS WAITED_____ _  TOT   _____
   ___ ENQS WAITED TIME_____ _  TOT   _____
   ___ SYSPLEX ENQS WAITED_____ _  TOT   _____
   ___ SYSPLEX ENQS WAITED TIME_____ _  TOT   _____
   ___ TOTAL ENQS RETAINED_____ _  TOT   _____
   ___ TOTAL ENQ RETENTION_____ _  TOT   _____
   ___ TOTAL REJECTED ENQBUSY_____ _  TOT   _____
   ___ TOTAL REJECTED ENQ RETAINED_____ _  TOT   _____
   ___ WAITING ENQS REJECTED RETAINED_____ _  TOT   _____
   ___ WAITING ENQS PURGED BY OPERATOR_____ _  TOT   _____
   ___ WAITING ENQS PURGED BY TIMEOUT_____ _  TOT   _____
```

7.   Back at the primary option menu, select option 2, Report Sets:

```
V5R4M0            CICS Performance Analyzer - Primary Option Menu
Option ===> 2_____

0  CICS PA Profile      Customize your CICS PA dialog profile
1  Systems              Specify Systems, SMF Files, and Groups
2  Report Sets          Request and submit reports and extracts
3  Report Forms         Define and submit Report Forms
4  Object Lists         Define Object Lists
5  Historical Database  Collect and process historical data
6  Statistics           Report CICS Statistics
7  Profiling            Request Transaction Profiling
8  Resource Definitions Define CICS PA resources
X  Exit                 Terminate CICS PA
```

8.   Enter S next to the ANALYTIC sample report set:

```
                              Report Sets                    Row 1 to 11 of 11
   Command ===> _____        Scroll ===> PAGE

   Report Sets Data Set . . : CPA.JSON.RSSAMP.RSET

   /    Name           Description                  Changed          ID
   S    ANALYTIC Analytics Sample Report Set    2020/11/17 13:41 TS8013
   ___  BFORAFTR Before and after comparison     2015/10/22 09:18 CICSPA
   ___  CONSOLDT Region consolidation analysis   2015/10/20 13:49 CICSPA
   ___  CPUINCRS Unexpected increase in CPU      2015/10/20 13:49 CICSPA
   ___  FILEACCS File access problem analysis    2015/10/20 13:49 CICSPA
   ___  MAXTASK  MXT exceeded analysis           2015/10/28 15:52 CICSPA
   ___  PLUGIN   CICS PA Plugin HDB Load & Export 2015/10/30 09:21 CICSPA
   ___  RESPINCR Unexpected increase in Response 2015/10/20 13:49 CICSPA
   ___  SOS      Short-on-Storage analysis       2015/10/28 15:53 CICSPA
   ___  THRDSAFE Transaction threadsafe analysis 2015/10/20 13:49 CICSPA
   ___  TSQUEUE  Temporary Storage analysis      2015/10/20 13:49 CICSPA
   ****************************** Bottom of data ******************************
```

9. Enter S next to either **Extracts** > **Statistics** or **Forwarding** > **Statistics**, depending on your site-specific practices:

```
   EDIT                    Report Set - ANALYTIC                  Row 1 of 18
   Command ===> _____        Scroll ===> CSR

   Description . . . Analytics Sample Report Set

   Enter "/" to select action.

   ___          ** Reports **                        Active
      +  ___    Options                              Yes
      +  ___    Selection Criteria                   No
      +  ___    Performance Reports                  No
      +  ___    Exception Reports                    No
      +  ___    Transaction Resource Usage Reports   No
      +  ___    Statistics Reports                   No
      +  ___    Subsystem Reports                    No
      +  ___    System Reports                       No
      -  ___    Extracts                             Yes
         ___       Cross-System Work                 No
         ___       Performance                       Yes
         ___       Record Selection                  No
         ___       HDB Load                          No
         ___       System Logger                     No
         ___       Statistics                        No
      -  ___    Forwarding                           Yes
         ___       Performance                       No
         S         Statistics                        Yes
                ** End of Reports **
```

10. Enter R (Repeat) next to the existing entry for ENQMANAG (for Enqueue Manager statistics *list* data).

```
                    ANALYTIC - Statistics Forwarding          Row 7 from 18
Command ===> _____     Scroll ===> CSR

        ——— System Selection ———
/ Exc APPLID +  Image +  Group +    Recap       Form +      Alert +
R       _____  _____  _____   ENQMANAG    ENQMANAG    _____
-----------------------------------------------------------------------------
```

11. On the newly created entry, replace the Recap and Form values with ENQMGSUM, and then enter line action S:

```
                    ANALYTIC - Statistics Forwarding          Row 7 from 19
Command ===> _____     Scroll ===> CSR

        ——— System Selection ———
/ Exc APPLID +  Image +  Group +    Recap       Form +      Alert +
_       _____  _____  _____   ENQMANAG    ENQMANAG    _____
-----------------------------------------------------------------------------
S       _____  _____  _____   ENQMGSUM    ENQMGSUM    _____
-----------------------------------------------------------------------------
```

12. Update other values according to your preferences, such as the summary interval, and then enter S next to **Options**:

```
                    ANALYTIC - Statistics Forwarding
Command ===> _____

System Selection:                        Forwarding Recap:
 APPLID . . _____  +                    DDname  . . . ENQMGSUM
 Image  . . _____  +
 Group  . . _____  +

Forwarding Focus:                        Connection Settings Override:
 Form . . . ENQMGSUM  +                    Port . . . . . . _____

Selection Criteria or                    Summary Options:
Alert Forwarding:                         Interval . . 03:00
 Alert  . . _____  +
  Severity _____   +
  _   Include Severity column
 Type . . / EOD  / INT  / USS  / REQ  / RRT

Forwarding Format:
 2  1. CSV   2. JSON                      S  Options

Repository  . . : CPA.JSON.RSSAMP.REPOSTRY
```

13. Set the **Code** to ENQMGSUM, and update other values, such as **OutZone**, according to your preferences:

```
                              Options
Command ===> _____

Code . . . . . ENQMGSUM_____

/  Time Stamp Fields in ISO8601 Format
OutZone  . . . _____    (Z for UTC or -12:00 to +14:00)
Encoding . . . 1  1. ASCII    2. EBCDIC
EOL  . . . . . 1  1. LF       2. CR     3. Newline  4. CRLF
Label Format   1  1. Standard 2. CICS   3. Short
Label Case . . 1  1. As is    2. Upper  3. Lower

CSV only:
 Delimiter . . ;
 /  Include Field Labels

JSON only:
 /  Exclude Missing Fields
```

14. Optional: To disable forwarding (or extracting) Enqueue Manager *list* statistics (in favor of the *summary* statistics that you have just defined), enter X next to the entry for ENQMANAG:

```
                    ANALYTIC - Statistics Forwarding       Row 7 from 19
Command ===> _____  Scroll ===> CSR

      ——— System Selection ———
/ Exc APPLID +  Image +  Group +    Recap      Form +     Alert +
X     _____  _____ _____   ENQMANAG   ENQMANAG   _____
-------------------------------------------------------------------------------
_     _____  _____ _____   ENQMGSUM   ENQMGSUM   _____
-------------------------------------------------------------------------------
```

15. Run the report set to forward (or extract) the required records.

Here is an excerpt from the report set JCL:

```
              STATISTICSSUMMARY(OUTPUT(ENQMGSUM),
                  DDNAME(CPAOSX01),
                  JSON,
                  CODE('ENQMGSUM'),
                  LABELS(STANDARD,ASIS),
                  TIMEFORMAT(ISO8601),
                  MISSING(EXCLUDE),
                  EBCDIC,
                  NOEOL,
                  INTERVAL(03:00),
                  TYPE(EOD,INT,USS,RRT),
FIELDS(APPLID(ASCEND),
       IMAGE(ASCEND),
       COLLECTTIME(ASCEND),
       NQGPOOL(ASCEND),
       NQGTNQSI(TOT),
       NQGTNQSW(TOT),
       NQGTNQWT(TOT),
       NQGGNQSW(TOT),
       NQGGNQWT(TOT),
       NQGTNQSR(TOT),
       NQGTNQRT(TOT),
       NQGTIRJB(TOT),
       NQGTIRJR(TOT),
       NQGTWRJR(TOT),
       NQGTWPOP(TOT),
       NQGTWPTO(TOT)))
```

Figure 16. Excerpt from the report set JCL to forward (or extract) summarized Enqueue Manager statistics

Characteristics of JSON Lines output by CICS Performance Analyzer

If you are a developer who needs to work with the JSON Lines output by CICS Performance Analyzer, then it's useful to know about the characteristics of this data that are specific to CICS Performance Analyzer, such as its structure, property names, and property values.

Flat: no nested objects

Each line of JSON Lines output by CICS Performance Analyzer is a JSON object consisting of a collection of name/value pairs ("properties").

The structure is flat: there are no nested objects.

No whitespace between tokens

The JSON standard (ECMA-404) allows insignificant whitespace before or after any token.

The JSON Lines output by CICS Performance Analyzer is deliberately compact and omits such whitespace.

For example:

```
{"code":"cicspa_stat_list_010a","APPLID":"FUWTCIC",
```

not:

```
{ "code" : "cicspa_stat_list_010a" , "APPLID" : "FUWTCIC" ,
```

Properties

The properties (key/value pairs) in each line of JSON Lines output by CICS Performance Analyzer depends on the following conditions:

- Whether the data is based on a report form or statistics alerts.
- Whether the corresponding `CICSPA` batch command specifies:
  - The `CODE` parameter, which inserts a `code` property as the first property of each line.
  - `MISSING(INCLUDE)` (default) or `MISSING(EXCLUDE)`. For details, see "Including or excluding fields that are missing from the original SMF records" on page 56.

Properties based on report forms

You can use CICS Performance Analyzer to output JSON Lines based on the following types of report form:

- Performance list
- Performance summary
- Statistics list
- Statistics summary

Each line of JSON Lines contains the fields specified by the form: that is, as specified by the `FIELDS` parameter of the `CICSPA` batch command.

The property order matches the order specified by the `FIELDS` parameter.

Statistics list only: CICS Performance Analyzer inserts the following fixed set of properties before the properties specified by the form:

- CICS application identifier (APPLID)
- MVS image (LPAR name)
- Statistics interval type (for example, "EOD" for "end of day")
- Statistics collection time

For example (shown with line breaks inserted for readability):

```
"APPLID":"FUWTCIC",
"Image":"FTS1",
"Type":"EOD",
"Collection Time":"2019-08-31T00:00:00+08:00"
```

Figure 17. Fixed JSON properties inserted before fields in a statistics list report form

Properties for statistics alerts

Each line of JSON Lines for statistics alerts contains a fixed set of properties. For example (shown with line breaks inserted for readability):

```
"Collection Time":"2019-08-31T00.00.00+08:00",
"Sev":"C",
"Alert":"File string waits",
"APPLID":"FUWFWFR",
"Image":"FTS1",
"System Type":"TS",
"Interval Type":"EOD",
"Threshold":">10",
"Actual":467,
"Resource":"File_Name",
"Resource value":"MBKACCT1",
"stid":"067A"
```

Figure 68. Fixed JSON properties for statistics alerts

If there is no applicable resource, then the `Resource` and `Resource value` property values are empty strings (`""`).

Optional `code` property at the start of each line

If you specify a `CODE` parameter on the `CICSPA` batch command that generates JSON Lines, then each line of output contains `code` as the first property.

For example, if you specify the parameter `CODE('stat_list_010a')` on the `CICSPA` batch command, then each line of JSON Lines output begins:

```
{"code":"stat_list_010a",…
```

It is your decision whether to include a `code` property in the JSON Lines and what value to specify. These choices depend on how you plan to use the JSON Lines.

Including the `code` property enables you to tag a set of properties—for example, the set of fields defined by a CICS Performance Analyzer report form—with a unique identifier. In the previous example, `"stat_list_010a"` refers to a Transaction Manager statistics list report form.

Some analytics platforms encourage or require the use of such an identifier, to make it easier to define searches that generate a homogeneous set of results: results with a known common set of properties that you can visualize in charts. In Splunk, this identifier is known as the *source type*, with a corresponding field named `sourcetype`.

If you choose to forward the JSON Lines output from each report form and statistics alerts to a different "input" of an analytics platform—for example, to different TCP ports—then you might not need to include a `code` property in the JSON Lines. Each input definition might specify its own identifier—in Splunk, its own source type—for the lines arriving at that input.

However, if you choose to forward all JSON Lines output to a single input of an analytics platform, then—again, depending on the analytics platform—you can use the `code` property to assign each incoming event its appropriate identifier.

CICS Performance Analyzer does not prescribe `code` property values. The values are your choice. However, the IBM-supplied Splunk app requires particular source types, corresponding to the `code` property values in the IBM-supplied sample data. For details, see "Sample data types, `code` property values, and report form names" on page 24.

For more information about the `code` property in the context of Splunk, see "Configuring Splunk for JSON Lines with or without a `code` property" on page 62.

Statistics ID (`stid`) property at the end of each line of statistics

Each line of JSON Lines that is based on statistics data—from statistics list or summary report forms, or statistics alerts—ends with a property named `stid`. The `stid` property identifies the source CICS statistics record.

For example:

```
{"code":"cicspa_stat_list_029a", … "Current DSA
Limit":5242880,"Current EDSA Limit":536870912, … "stid":"029A"}
```

The `stid` property helps you to correlate related statistics events. For example, you can use the `stid` property value in a statistics *alert* event to find the related statistics *list* event—in the corresponding time period, for the same applid—containing details of the statistics field values that triggered the alert.

A statistics summary report form can refer to fields from *multiple* types of statistics records. If a form refers to fields from only a *single* type of statistics record, then the `stid` property in the corresponding JSON Lines contains that identifier. Otherwise, if a form refers to fields from *multiple* types of statistics records, the `stid` property value is an empty string (`""`).

ZC Tracking Token property at the end of each line of z/OS Connect request line and related Performance line

Each line of z/OS Connect and related Performance JSON Lines ends with a property named `ZC Tracking Token`.

For example:

```
{
     "code": "ZCON_REQWAIT",
     "JOBNAME": "ZOSCON2A",
     "SORID": "ROCKNET1.FUWFWAR",
     "ZCENTRY": "2021-06-08T09:58:00",
     "Request": 255674,….
     "ZC Tracking Token": "RSPLEXL6RSD6     D9D47C8B8A39C6C0"
}
```

The `ZC Tracking Token` property links the z/OS Connect request lines with the related Performance lines and is used to navigate from the *z/OS Connect requests overview* dashboard to the *z/OS Connect CICS transaction overview* dashboard.

Event time stamps

Analytics platforms typically assign a time stamp to each incoming event.

When you design a report form in CICS Performance Analyzer for exporting data to an analytics platform, you should consider which field you want the analytics platform to use as the event time stamp.

The sample report forms supplied with CICS Performance Analyzer for output to JSON Lines follow a convention: the *first* time-format field in the report form is the event time stamp. The sample forms use the following fields:

Table 3. Fields used as event time stamps in sample report forms for output to JSON Lines

Report form type	Field name specified in `CICSPA FIELDS` parameter	Default JSON property name	Description
Performance list	`START`	`Start`	CICS transaction start time
Performance summary	`START`	`Start Interval`	Start of CICS Performance Analyzer summary reporting interval
Statistics	`COLLECTTIME`	`Collection Time`	CICS Statistics interval collection time
z/OS Connect request	`SMF123S1_TIME_ZC_ENTRY`	`ZCENTRY`	The time the request was received by the z/OS Connect EE server

Statistics list report forms and statistics alert definitions do not support explicitly specifying a leading time-format field; for these data types, CICS Performance Analyzer inserts a `Collection Time` property that represents the statistics interval collection time.

Property names

The property names (keys) in JSON Lines output by CICS Performance Analyzer typically match, or are closely based on, the column headings in CICS Performance Analyzer reports. These property names might not match the field names specified in the `FIELDS` parameter of the `CICSPA` batch command.

For example, here is a snippet of JSON Lines generated using a CICS Performance Analyzer performance list report form (with line breaks inserted for readability):

```
"APPLID":"FUWTCIC",
"Tran":"CWXN",
"Start":"2019-08-31T00:00:00.885283+08:00",
"Dispatch Time":0.0687,
"User CPU Time":0.0011,
"Suspend Time":0.7523,
"TaskNo":"54272"
```

Figure 79. JSON keys are based on report column headings, not field names

Property name components in form-based output

Property names in JSON Lines based on report forms consist of several components, depending on the type of report form.

Performance list or performance summary

Property names in JSON Lines based on performance report forms consist of the following components, in order:

1. **Name**
   A "base" (or "stem") name that corresponds to the field in the report form. For example, `User CPU`.

   Application group names are followed by the qualifier `Group`.

2. **Type**
   For clock fields, the selected component type: `Time` or `Count`.

3. **Units**
   For numeric fields, such as count fields, that can be reported in different units: `K`, `M`, `KB`, or `MB`.

4. **Function**
   For summarized fields in performance summary report forms: the summarizing function name, such as `Avg` or `Total`.

   For range functions: the range value, such as `0.0-0.1`.

For peak percentile functions: the percentage value followed by the percent sign. For example, 80%.

For performance alerts: the severity (`Critical`, `Warning`, or `Info`).

Example property names for performance list data:

```
APPLID
Start
Dispatch Time
Dispatch Count
```

Example property names for performance summary data:

```
APPLID
Start Interval
Dispatch Time Avg
Dispatch Count Total
PCLOADWt Time S Dev
User CPU Time 0.0-0.1
User CPU Time 80%
```

Statistics list or statistics summary

Property names in JSON Lines based on statistics report forms consist of the following components, in order:

1. **Function**
   For summarized fields (in summary report forms), the summarizing function, such as `Fin` or `Tot`.

2. **Name**
   A "base" (or "stem") name that corresponds to the field name in the original SMF records. For example, `Current MAXTASK`.

Notable differences between property names for performance summary data and statistics summary data:

- In statistics property names, the summarizing function appears at the *start* of the property name. In performance property names, the summarizing function appears at the *end*.

- In statistics summary property names, the Total function is represented by the 3-character abbreviation `Tot`. Performance summary property names contain the full word `Total`. This is an existing difference in CICS Performance Analyzer extracts, unchanged for these JSON Lines enhancements.

Uniqueness of property names

You can create report forms in CICS Performance Analyzer that generate non-unique property names in JSON Lines.

For example:

- You can create report forms that contain identical rows, such as the same field summarized in the same way. This will result in JSON Lines output containing multiple instances of the same property name.

- "Percentage" and "count" performance alerts of the same severity for the same field generate the same property name. That is, the following two `CICSPA` batch command parameters both generate the property name `TS Wait Time Info`:

    ```
    TSWAIT(TIME(SEV(INFO,PERCENT))),
    TSWAIT(TIME(SEV(INFO,COUNT))),
    ```

    The only noticeable different in the JSON Lines output is in the property values: the count value is an integer, whereas the percentage value has a decimal point followed by a single digit.

Whether these identically named properties cause an error, or which value takes precedence, depends on what you use to process the JSON Lines.

The JSON syntax (described in the ECMA-404 standard) does not *require* that property names are unique, but it's typically good practice.

Controlling property name length and case

The `LABELS` parameter of the `CICSPA` report operands for output to JSON Lines controls the length and case of property names. For details, see "New parameters of `CICSPA` report operands" on page 70.

Property values

Property values in JSON Lines output by CICS Performance Analyzer depend on the field type in CICS Performance Analyzer:

Table 4. How CICS Performance Analyzer represents values of each field type in JSON

Field type	Value	Example JSON key/value pair
Count	Integer	`"#Tasks":17876` `"FCAMCT ":17376`
Elapsed time	Floating-point number of seconds.	`"User CPU Time`

Field type	Value	Example JSON key/value pair
	A maximum precision of microseconds (up to 6 decimal places), depending on the property.	`Total":67.7207` `"ENQDelay Time Total":0.0`
Percentage	Floating-point percentage value generated by the `RNGPERCENT` summarization function.	`"Suspend Time 0.0-0.1":6.51` `"Suspend Time 0.0-0.1":100.0`
Storage	Number of *bytes,* unless the property name contains a qualifier that explicitly specifies other units.	`"Peak DSA Size":262144` `"PC31AHWM KB":29`
Character	String. No trailing whitespace. For example, if a field can have up to 8 characters, but a value has only 3 characters, `"ABC"`, then, in the JSON Lines, the property value is `"ABC"`, with no trailing blanks.	`"Tran":"BPMT"`

Clock	For output that is based on a report form, the value of clock (time stamp) fields in JSON Lines output depends on either:
	• The time format specified for each field in the report form
	• The `TIMEFORMAT` parameter of the `CICSPA` batch command, which overrides per-field time formats specified in the report form
	
	Consider specifying `TIMEFORMAT(ISO8601)`, so that all time stamps are represented in ISO 8601 format. Also consider specifying the `OUTZONE` parameter, to output time stamps with a zone designator.
	
	If a clock field is present in the original SMF record, but does not have a valid time stamp value, then the JSON property value is an empty string (`""`).
		`"Start":"2019-08-31T00:00:00.885283+08:00"`
		`"Start Interval":"2019-10-07T10:30:00+08:00"`
		`"PHStart":""`

Exponential format

Large numbers might be represented in exponential format. For example, 987600000 might be represented as 9876E5.

Floating-point field values that happen to be an integer

Values of floating-point numeric fields—elapsed times and percentages—*always* contain a decimal point followed by at least one digit.

If an elapsed time or percentage happens to be a whole (integer) number, with no decimal fraction, then the property value still contains a decimal point and trailing zero. For example:

```
0.0
10.0
100.0
```

Some analytics platforms distinguish between integer and non-integer numbers and use different internal data types for each. Some analytics platforms infer this internal data type based on the first property value in the incoming data. Data corruption can occur if an analytics platform encounters values with decimal fractions for a field that has already been mapped to an integer data type.

The consistent presence of a decimal point ensures that such analytics platforms do not incorrectly map elapsed-time or percentage properties to an integer data type.

No extraneous trailing zeros in decimal fractions

The number of digits in a decimal fraction does not necessarily indicate the field precision.

For example, the elapsed time field value `0.1` does not indicate a precision of a tenth of a second.

For conciseness, numeric values have no trailing zeros in the decimal fraction, except for a single zero following the decimal point to identify a field as floating-point instead of integer.

Including or excluding fields that are missing from the original SMF records

For all field types, if CICS Performance Analyzer cannot output a meaningful JSON property value because the corresponding field is missing from the original SMF record, then the JSON property value is the JavaScript value `null`. For example:

```
"FCVSWait Time":null
```

This situation can occur when a report form refers to a field that exists only in recent versions of CICS, but the input SMF records were generated by a backlevel CICS version that does not support that field.

In output from *summary* report forms, where a JSON property value is aggregated from *multiple* original SMF records—for example, using a Total or Average function—the property value is `null` if the field is missing in one or more of the records being aggregated.

Some analytics platforms interpret the JavaScript value `null` as the string value `"null"`.

To avoid that issue, or simply to save space, instead of including missing fields with the value `null`, you can exclude them from JSON Lines output by specifying the parameter `MISSING(EXCLUDE)` in the `CICSPA` batch command.

No character escaping

Some characters in JSON strings, such as the quote ("), *must* be escaped. Other characters *can* be escaped.

CICS Performance Analyzer does not escape *any* characters in JSON Lines.

Potentially, this could result in invalid JSON. For example, the list of characters allowed in a CICS transaction identifier includes the quote (").

Rather than overengineering CICS Performance Analyzer to solve a potential problem that does not occur in practice, and unnecessarily affecting performance, the developers have chosen to see if any customers experience this issue.

If you experience issues caused by lack of character escaping in JSON Lines output by CICS Performance Analyzer, please contact IBM Software Support.

Character encoding

For many of the character fields that CICS Performance Analyzer outputs to JSON Lines, the source data—for example, the SMF records—contain no information about the character encoding of those fields. Specifically, their EBCDIC code page.

Fortunately, in practice, the contents of character fields that CICS Performance Analyzer outputs to JSON Lines are typically limited to the EBCDIC *invariant subset*: the subset of characters that have the same code points regardless of code page.

If you experience character encoding issues in JSON Lines output by CICS Performance Analyzer, please contact IBM Software Support.

For more details on character encoding, see "EBCDIC|ASCII" on page 73.

Using JSON Lines from CICS Performance Analyzer in Splunk

The JSON Lines output by CICS Performance Analyzer is platform-agnostic; it is designed to be ingested by any analytics platform that supports JSON Lines.

Splunk was cited as the analytics platform of choice by customers who requested JSON Lines output from CICS Performance Analyzer.

Minimal Splunk configuration

A minimal Splunk configuration for ingesting JSON Lines from CICS Performance Analyzer consists of the following stanza in `props.conf`:

```
[cicspa_minimal]
SHOULD_LINEMERGE = false
KV_MODE = json
pulldown_type = 1
```

Figure 20. Minimal Splunk configuration in `props.conf`

The source type "cicspa_minimal" in this stanza is an example only.

With this configuration, you can use the **Add Data** > **Upload** option in Splunk Web to add a file of JSON Lines on your computer to Splunk. For details, see the Splunk documentation heading "The Add Data page".

Tip: In the Source type dropdown list on the Set Source Type page, this source type will appear under the "Uncategorized" heading.

Splunk configuration by the IBM-supplied app

The IBM-supplied Splunk app for CICS Performance Analyzer implements the configuration described here. If you have installed the app, then this information is for your interest only, because the app has already configured Splunk for you.

The app configuration details are useful if, for example:

- You want to understand the app in more detail before you install it

- You want to develop your own app

- You want to customize some of the app's configuration settings, such as the TCP port—or ports, plural—on which Splunk listens for JSON Lines from CICS Performance Analyzer.

The app configures Splunk to ingest data using either of the following two methods:

- In Splunk Web, selecting **Add Data** to upload a file of JSON Lines that is on your computer

- Forwarding JSON Lines over a TCP network to a port that Splunk is listening to

For information about installing and using the app, see "Splunk app that visualizes data from CICS Performance Analyzer" on page 12.

Location of Splunk configuration stanzas

This documentation refers to Splunk configuration (`.conf`) file names, but not directory paths. It is your decision where to store the Splunk configuration stanzas for CICS Performance Analyzer.

In the IBM-supplied Splunk app for CICS Performance Analyzer, the configuration stanzas are located in the `.conf` files in the `default` subdirectory of the app:

Path to your Splunk directory`/etc/apps/cicspa/default/*.conf`

props.conf

The following stanza in `props.conf` defines the properties of the `cicspa` source type:

```
[cicspa]
SHOULD_LINEMERGE = false
KV_MODE = json
# Timestamp:
# - ISO 8601 extended format
# - Seconds to a maximum precision of 6 decimal places
# - With zone designator
TIME_PREFIX = (?=\d{4}-\d{2}-\d{2}T)
TIME_FORMAT = %Y-%m-%dT%H:%M:%S.%6N%:z
# Per-event overrides:
# Override the default sourcetype
TRANSFORMS-changesourcetype = set_sourcetype_cicspa, remove_code_property
category = Application
description = JSON Lines produced by IBM CICS Performance Analyzer for z/OS
pulldown_type = 1
```

Figure 21. `props.conf` in the IBM-supplied Splunk app

Data format

The combination of `SHOULD_LINEMERGE = false` and `KV_MODE = json` defines the incoming data as JSON Lines: one event per line, data in JSON format.

These two settings apply to different stages in the Splunk data pipeline: `SHOULD_LINEMERGE` applies to parsing, before indexing; `KV_MODE` applies later, to search-time field extraction.

Time stamp extraction

Rather than relying on all JSON Lines from CICS Performance Analyzer using the same, fixed property name for the event time stamp, this configuration uses the *first* value that matches `TIME_FORMAT` as the event time stamp.

This configuration has significant implications for the design of CICS Performance Analyzer performance report forms and statistics summary report forms. When creating one of these forms for output to JSON Lines, insert the time field that you want to use as the event time stamp *before* any other time fields. For other types of JSON Lines data from CICS Performance Analyzer, such as statistics alerts or data generated using statistics list report forms, CICS Performance Analyzer inserts an appropriate time field.

This configuration relies on the following default setting that is not explicitly specified in the previous example stanza:

```
MAX_TIMESTAMP_LOOKAHEAD = 128
```

`MAX_TIMESTAMP_LOOKAHEAD` specifies the number of characters into an event that Splunk looks for a time stamp. If Splunk does not find a value that matches `TIME_FORMAT`, time stamp extraction fails.

The `TIME_FORMAT` setting in this configuration matches the ISO 8601 date and time of day representation extended format with up to microsecond precision (fractions of a second to a maximum of 6 decimal places) and a zone designator.

Example time stamp values:

```
2018-12-31T13:59:00.123456+08:00
2018-12-31T05:59:01.03Z
2018-12-31T05:59:15Z
```

Transforms

Each incoming line of JSON Lines from CICS Performance Analyzer gets processed by two transforms: the first uses the value of the `code` property to override the source type, the second removes the `code` property.

For more information, see `transforms.conf`.

inputs.conf

If you exclusively ingest data from CICS Performance Analyzer into Splunk by using Splunk Web to upload files of JSON Lines that are on your computer, then you do not need any stanzas in `inputs.conf`. This situation is possible if you are performing small-scale ad-hoc testing, but unlikely in a production environment.

The following stanza in `inputs.conf` defines an unsecure TCP input that listens on port 1516, assigns the source type `cicspa` to all incoming events, and stores the events in an index that is also named `cicspa`:

```
[tcp://:1516]
index = cicspa
sourcetype = cicspa
disabled = 0
```

Figure 22. `inputs.conf` in the IBM-supplied Splunk app

The port number, index name, and source type shown here are examples only. The actual values are your choice. Note, however, that the IBM-supplied Splunk app for CICS Performance Analyzer is configured to use these example values, and the dashboards in the app perform searches that are compatible with these index and source type values.

Configuring Splunk to ingest data over *secure* TCP (SSL/TLS) is outside the scope of this documentation. For more information, see the Splunk docs.

indexes.conf

The `index` setting in the `[cicspa]` source type stanza in `props.conf` instructs Splunk to store events in an index named `cicspa`. The following stanza in `indexes.conf` defines rudimentary settings for a `cicspa` index:

```
[cicspa]
coldPath = $SPLUNK_DB/$_index_name/colddb
homePath = $SPLUNK_DB/$_index_name/db
thawedPath = $SPLUNK_DB/$_index_name/thaweddb
```

Figure 23. `indexes.conf` in the IBM-supplied Splunk app

You might choose to store events from CICS Performance Analyzer in a different index, such as the default `main` index. Note, however, that the IBM-supplied Splunk app for CICS Performance Analyzer performs searches that refer to the `cicspa` index.

transforms.conf

The following stanza in `transforms.conf` defines two transforms that Splunk applies to every incoming line of JSON Lines from CICS Performance Analyzer:

```
[set_sourcetype_cicspa]
# Set sourcetype to cicspa_ followed by value of code property
REGEX = \"code\":\"([^\"]+)\"
FORMAT = sourcetype::cicspa_$1
DEST_KEY = MetaData:Sourcetype
[remove_code_property]
# Remove code property to conserve license usage
REGEX = ^({.*)"code":"[^"]+",(.*)$
FORMAT = $1$2
DEST_KEY = _raw
```

Figure 24. `transforms.conf` in the IBM-supplied Splunk app

First, Splunk applies the `set_sourcetype_cicspa` transform, which uses the value of the `code` property in the incoming line to set the `sourcetype` default field.

Then, Splunk applies the `remove_code_property` transform, which removes the `code` property from the incoming line before indexing (before Splunk stores the event in an index).

The result: given an incoming line of JSON Lines containing `"code":"TRANSUM"`, the corresponding indexed event in Splunk will have the `sourcetype` value `cicspa_TRANSUM`.

macros.conf

`macros.conf` defines macros that are used by the app dashboards.

Rather than directly citing an index name in searches, the dashboards refer to macros. If you want to use a different index than `cicspa`, then, instead of editing all of the searches in all of the dashboards, you only have to edit one macro, `cicspa_search_start_default`.

Custom JavaScript

The app contains a small amount of custom JavaScript, supplied in the `appserver/static` subdirectory.

This custom JavaScript depends only on software already available in the Splunk Web framework.

The app has no external dependencies; it requires no additional libraries or frameworks.

Configuring Splunk for JSON Lines with or without a `code` property

CICS Performance Analyzer can generate JSON Lines with or without a `code` property. If you specify a `CODE` parameter in the `CICSPA` batch command that generates the JSON Lines, then each line of JSON Lines begins with a `code` property. If you omit the `CODE` parameter, the JSON Lines does not contain a `code` property.

The `code` property identifies the type of data in each line of JSON Lines. For example, the value offers a way of identifying which report form was used to generate the data; which fields the data contains, from which record type or types. You can use the value as a filter when searching data in analytics platform.

Splunk assigns each event (each line of JSON Lines) a *source type*. The source type identifies the type of data in the event. Other analytics platforms typically have a similar concept.

The presence or absence of the `code` property affects Splunk configuration requirements:

- If you include the `code` property in the JSON Lines, then you can configure Splunk to use the value of the `code` property as the Splunk event source type. In Splunk, this technique is known as overriding source types on a per-event basis. Using this technique, you can forward all JSON Lines from CICS Performance Analyzer to a single Splunk input, such as a single TCP port, and Splunk will assign each event its correct source type.

- However, if you omit the `code` property from the JSON Lines, then you must define a separate Splunk input, such as a separate TCP port, for each source type. That is, you must define a separate input for what would have been each `code` property value. For each input, you specify the corresponding source type. This reduces the index-time processing required to ingest the JSON Lines, because there is no need to override source types on a per-event basis.

The IBM-supplied Splunk app for CICS Performance Analyzer is configured to ingest JSON Lines that contains the `code` property.

Configuring Splunk for JSON Lines without a `code` property

If the JSON Lines has no `code` property, then you don't need a `transforms.conf` or the corresponding references to transforms in `props.conf`.

Instead, in `inputs.conf`, you define an input per source type. For example:

```
[tcp://:1516]
index = cicspa
sourcetype = cicspa_DSA
disabled = 0
[tcp://:1517]
index = cicspa
sourcetype = cicspa_ENQMANAG
disabled = 0
[tcp://:1518]
index = cicspa
sourcetype = cicspa_STORAGE
disabled = 0
…
```

Figure 25. inputs.conf snippet for a multi-port, one-sourcetype-per-port approach

Could you streamline this configuration with a single wildcarded stanza that sets the index for all ports in the appropriate range? Like this:

```
[tcp://:151*]
index = cicspa
```

Perhaps, but this has not been tested by the developers of the IBM-supplied Splunk app.

Ideally, in props.conf, you would specify a single wildcarded stanza for common settings, with specific stanzas for unique settings (but this doesn't work!):

```
[cicspa_*]
SHOULD_LINEMERGE = false
KV_MODE = json
TIME_PREFIX = (?=\d{4}-\d{2}-\d{2}T)
TIME_FORMAT = %Y-%m-%dT%H:%M:%S.%6N%:z
[cicspa_DSA]
category = Application
description = CICS statistics ID 029B
pulldown_type = 1
```

Figure 26. props.conf snippet for a multi-port, one-sourcetype-per-port approach

There is a workaround, but its official support is questionable. See:

- The following question in the Splunk Answers user community website: "Is it possible to use wildcards in sourcetype props.conf stanzas"

- The July 2014 Splunk blog post "Quick Tip: Wildcard Sourcetypes in Props.conf"

Getting data in

After you have configured Splunk to ingest JSON Lines from CICS Performance Analyzer—either by installing the IBM-supplied app, or by configuring Splunk yourself—you can get data into Splunk using either of the following methods:

- In Splunk Web, selecting **Add Data** to upload a file of JSON Lines that is on your computer. For example, a file output by CICS Performance Analyzer that you have transferred from z/OS to your personal computer. For an example step-by-step procedure, see "Uploading the sample data" on page 15.

- Submitting a CICS Performance Analyzer report set that forwards JSON Lines over a TCP network to a port that Splunk is listening to. For details, see "Using CICS Performance Analyzer to output JSON Lines" on page 28.

New CICSPA batch command syntax

The enhancements for JSON Lines output introduce new syntax to the CICSPA batch command.

Summary of changes to batch syntax

There are no *breaking* changes. All existing syntax is supported as-is. There is no need to change existing JCL.

Summary of changes:

- New CICSPA control operand CONNECTION to define a destination for streaming data over TCP.

- New parameters on the existing CICSPA report operands LIST, SUMMARY, STATISTICSLIST, STATISTICSSUMMARY, and STATSALERT.

- New CICSPA report operand ZCON to run the z/OS Connect EE report, Extract and Forwarding. New parameters for the new CICSPA report operand ZCON.

New CICSPA control operand: CONNECTION

The enhancements for JSON Lines output introduce the new control operand CONNECTION to the CICSPA batch command.

CONNECTION defines a destination for streaming JSON Lines or CSV over a network to a TCP socket. Typically, the TCP socket is on a remote host that is running an analytics platform such as Splunk. The analytics platform ingests the data that you send. The connection settings include details such as the destination host name and port number and, optionally, security (SSL/TLS) details.

To stream JSON Lines or CSV to a TCP socket, instead of writing to an MVS data set or a z/OS UNIX file, you specify a CONNECTION control operand, and then, on the report operand—LIST, SUMMARY, STATISTICSLIST, STATISTICSSUMMARY, STATSALERT or ZCON —you specify a STREAM parameter instead of a DDNAME parameter. The STREAM parameter instructs CICS Performance Analyzer to stream output using the CONNECTION settings instead of writing output to a ddname.

Syntax

```
CONNECTION(HOST('host_name'),
          PORT(port_number),
          [TIMEOUT(seconds)|TIMEOUT(0)],
          [Security parameters])
```

HOST

    The destination hostname or IP address, up to 255 characters.

    Enclose the value in single quotes.

PORT

    The destination port number.

    Each report operand can specify its own PORT parameter, overriding this port number.

TIMEOUT

    How long to wait before timing out. Either:

    −   An integer number of seconds

    −   0 (the default) to wait forever (no timeout)

Security parameters

Security parameters are required only for secure (SSL/TLS) connections.

```
SECURITY(TLS*|TLSV1.2|TLSV1.1|TLSV1.0,…),
[FIPS,]
KEYRING('[user_id/]saf_key_ring_name' |
        '*TOKEN*/pkcs#11_token_name' |
        'pkcs#12_unix_file_path' |
        'key_database_unix_file_path'),
[PASSWORD('password')|STASH('stash_unix_file_path')]
[CIPHERS(cipher_suites),]
[CERTLABEL('label')]
```

Note: Enclose the values of the following parameters in single quotes: KEYRING, PASSWORD, STASH, CERTLABEL.

SECURITY

    Specifies one or more security protocols to try, in order. The special value TLS*
    represents the list of all supported versions of TLS, starting with the latest version:

    TLSV1.2,TLSV1.1,TLSV1.0

    If you omit the SECURITY parameter, CICS Performance Analyzer attempts to open a
    connection without SSL/TLS.

FIPS

Sets z/OS System SSL Federal Information Processing Standards (FIPS) mode on. For information about FIPS mode, see the z/OS System SSL documentation.

KEYRING

Specifies a collection of certificates that includes the certificates required for this connection. Can be one of the following:

SAF key ring

Specified in the format `owner_user_id/key_ring_name` or, if the current user owns the key ring, just `key_ring_name`. For example:

`my/cicspa_keyring`

If the current user owns the key ring, the current user must have READ access to the IRR.DIGTCERT.LISTRING resource in the FACILITY class. If another user owns the key ring, the current user must have UPDATE access to that resource.

Certificate private keys are not available when using a SAF key ring owned by another user, except for SITE certificates where CONTROL authority is given to IRR.DIGTCERT.GENCERT in the FACILITY class or for user certificates where READ or UPDATE authority is given to *ring_owner.ring_name*.LST resource in the RDATALIB class.

Key database

A key database created by the z/OS gskkyman utility. The key database is specified as a z/OS UNIX file path. For example:

`/u/my/sslcerts/cicspa.kdb`

PKCS #12 file

Specified as a z/OS UNIX file path. For example:

`/u/my/sslcerts/cicspa.p12`

PKCS #11 token

Specified in the format `*TOKEN*/token_name`. For example:

`*TOKEN*/cicspa.pkcs11.token`

The `*TOKEN*` qualifier indicates that the value refers to a PKCS #11 token rather than a SAF key ring.

If you specify a key database or PKCS #12 file, but you do not specify either a STASH parameter or a PASSWORD parameter, then CICS Performance Analyzer looks for a stash file in the same directory as the key database or PKCS #12 file, and with the same base file name, but with .sth extension. For example, if the KEYRING parameter specifies the following z/OS UNIX file path:

```
/u/my/sslcerts/fuw.kdb
```

or:

```
/u/my/sslcerts/fuw
```

(with no extension)

then CICS Performance Analyzer looks for a stash file at the following path:

```
/u/my/sslcerts/fuw.sth
```

STASH

Specifies the z/OS UNIX path of the stash file that contains the password for the key database or PKCS #12 file.

If `KEYRING` specifies a SAF key ring or PKCS #11 token, `STASH` is ignored.

The stash file name must have a `.sth` extension. If the specified file name has a different extension, that extension is ignored and replaced with the `.sth` extension.

If the `PASSWORD` parameter is specified, `STASH` is ignored.

PASSWORD

Specifies the password for the key database or PKCS #12 file.

If `KEYRING` specifies a SAF key ring or PKCS #11 token, `PASSWORD` is ignored.

CIPHERS

Specifies a list of candidate cipher suites to try, in order. The list is a concatenation of 4-digit hexadecimal cipher suite numbers supported by z/OS System SSL. For example:

```
CIPHERS(000A000D001000130016)
```

If you omit `CIPHERS`, CICS Performance Analyzer uses the system default list of cipher suites. That list changes depending on whether or not FIPS mode is on.

Tip: To match a z/OS System SSL cipher suite number to the corresponding OpenSSL cipher suite name, go to the z/OS System SSL documentation and look up the "short name" for that cipher suite in the table of cipher suite definitions. The short name is the name that is defined in the associated RFC. Then go to the OpenSSL documentation for the ciphers command, and use the RFC name to find the corresponding OpenSSL name.

For more information on cipher suite definitions, see the z/OS System SSL documentation.

CERTLABEL

Specifies the label of the client certificate that is used to authenticate CICS Performance Analyzer (the client) to the destination host (server). The client certificate, and its private key, must be in the collection that is specified by the `KEYRING` parameter.

`CERTLABEL` is only used if the destination host requires client authentication.

If the destination host requires client authentication, but you omit `CERTLABEL`, then CICS Performance Analyzer uses the default certificate from the collection that is specified by the `KEYRING` parameter.

New parameters of `CICSPA` report operands

The enhancements for JSON Lines output introduce new parameters to the following report operands of the `CICSPA` batch command:

`CICSPA` report operand	Description
`LIST`	Form-based performance list extract
`SUMMARY`	Form-based performance summary extract
`STATISTICSLIST`	Form-based statistics list extract
`STATISTICSSUMMARY`	Form-based statistics summary extract
`STATSALERT`	Statistics alerts extract
`ZCON`	z/OS Connect request list summary extract

Syntax

New parameters are shown highlighted:

```
CICSPA
[LIST|SUMMARY|STATISTICSLIST|STATISTICSSUMMARY|STATSALERT|ZCON](

  …
  [JSON,]
  [LABELS(label-options)|NOLABELS,]
  [CODE('string'),]
  [CODECMF('string'),]
  [CODEZCON('string'),]
  [EBCDIC|ASCII,]
  [MISSING(INCLUDE|EXCLUDE),]
  [TIMEFORMAT(ISO8601),]
  [OUTZONE(Z|+hh:mm|-hh:mm),]
  [STREAM[,PORT(port-number)]]
  …
)
```

Notes:

- The ellipses (…) represent existing parameters not shown here. For details, see the CICS Performance Analyzer documentation in IBM Knowledge Center.

- `LABELS` and `NOLABELS` are existing parameters for the operands that already supported output to CSV: `LIST`, `SUMMARY`, `STATISTICSLIST`, `STATISTICSSUMMARY`, `ZCON`.

  Previously, however, `LABELS` had no subordinate values in parentheses. The parentheses with subordinate values following `LABELS` are optional: you can still use `LABELS` as a keyword by itself without parentheses.

- Before the enhancements for JSON Lines, the `STATSALERT` operand did not support output to delimiter-separated values. In addition to introducing JSON Lines output to `STATSALERT`, these enhancements also introduce delimiter-separated values output. For details, see "`STATSALERT` now also outputs CSV" on page 79.

- Except for `MISSING`, all of these new parameters for JSON Lines also apply to CSV.

Parameter descriptions

JSON

Specifies JSON Lines as the output data format.

If you omit the `JSON` parameter, then the output data format reverts to CSV.

LABELS(*label-options*)

The `LABELS` parameter controls the length and case of the property names in the JSON Lines output.

The syntax and allowed values for *label-options* depends on the data type:

Performance data and statistics alerts

With the `CICSPA` report operands `LIST,` `SUMMARY,` `STATSALERT and ZCON`, you can control the *case* of property names:

```
LABELS(ASIS|UPPER|LOWER)
```

Statistics data

With the `CICSPA` report operands `STATISTICSLIST` and `STATISTICSSUMMARY`, you can control the *length and case* of property names:

```
LABELS(STANDARD|SHORT|CICS,ASIS|UPPER|LOWER)
```

Length (statistics data only):

STANDARD

Base the names on the corresponding CICS Performance Analyzer report column headings. This is the default value.

Maximum length: 48 characters for unsummarized fields, 52 for summarized fields. Property names of summarized fields contain an additional trailing 4 characters: a space followed by a 3-character abbreviation for the summarization function.

CICS

Base the names on the corresponding CICS statistics field names. These names are typically shorter, more cryptic than the `STANDARD` names.

Maximum length: 63 for unsummarized fields, 67 for summarized fields.

SHORT

Use abbreviated forms of the `STANDARD` names.

Maximum length: 18 for unsummarized fields, 22 for summarized fields.

Case:

ASIS

Follow the same convention of mixed upper and lower case as the report column headings. This is the default value.

UPPER

All uppercase.

LOWER

All lowercase.

Specifying `LABELS` by itself, with no following parentheses, is equivalent to specifying `LABELS(STANDARD,ASIS)` for statistics data or `LABELS(ASIS)` for other data types.

These settings only affect the case of property *names*, not property *values*.

NOLABELS

By definition, JSON Lines involves "labels", in the form of property names (keys). For that reason, the combination of `JSON` and `NOLABELS` is nonsensical.

If you specify both `NOLABELS` and `JSON`, then CICS Performance Analyzer ignores `NOLABELS` and proceeds as if you had specified `LABELS` or the equivalent, more specific, `LABELS(STANDARD,ASIS)`.

CODE('*string*')

Inserts an optional property named `code` at the start of each line of JSON Lines output.

Enclose the string value in single quotes.

For information about why you might choose to insert a `code` property, see "Optional code property" on page 49.

CODECMF('*string*')

Inserts an optional property named `code` which characterizes the type of CICS Performance data for z/OS Connect EE extract at the start of each line of JSON Lines output.

Enclose the string value in single quotes.

For information about why you might choose to insert a `codecmf` property, see "Optional code property" on page 49.

CODEZCON('*string*')

Inserts an optional property named `code` which characterizes the type of z/OS Connect EE data for z/OS Connect EE extract at the start of each line of JSON Lines output.

Enclose the string value in single quotes.

For information about why you might choose to insert a `codezcon` property, see "Optional code property" on page 49.

EBCDIC|ASCII

Specifies the character encoding of the JSON Lines output:

EBCDIC
    Which EBCDIC code page? The answer is not straightforward.

The JSON property names, values, and syntax tokens that CICS Performance Analyzer generates consist of characters exclusively from the *invariant subset* of EBCDIC. These characters have the same code point regardless of code page, so the question "Which code page?" is irrelevant.

However, when inserting character field values from SMF records into the JSON output, CICS Performance Analyzer copies the original field byte values from the original SMF records. The SMF records contain no information about the encoding, such as the EBCDIC code page, of such fields. Typically, in practice, such character fields also only contain characters from the EBCDIC invariant subset. However, that is not guaranteed: indeed, it is possible to store binary data in character fields.

A similar caveat applies to statistics alert descriptions: CICS Performance Analyzer uses the original byte values of these descriptions as property names in the JSON output.

ASCII

To output JSON Lines in ASCII, CICS Performance Analyzer uses the z/OS default character conversion table to convert from EBCDIC to ASCII.

For details, see the following z/OS topic in IBM Knowledge Center: z/OS DFSMS > z/OS DFSMS Using Data Sets > Converting character sets > Converting from EBCDIC to ASCII

Note that, using this conversion table, many characters outside of the invariant subset of EBCDIC, and characters that exist in EBCDIC but not in ASCII, get converted to the ASCII substitute character (X'1A').

The JSON standard (ECMA-404, December 2017) states that "JSON syntax describes a sequence of Unicode code points".

By definition, then, text encoded in EBCDIC *cannot be JSON*. In practice, this is typically not an issue; for example, because you typically convert EBCDIC-encoded data to another encoding when transferring it off z/OS.

By contrast, ASCII is a (7-bit) subset of the UTF-8 Unicode character encoding.

`MISSING(INCLUDE|EXCLUDE)`

Specifies whether to include or exclude fields that are selected for output but are missing from the input SMF record.

- `MISSING(INCLUDE)` includes such missing fields in the output, with the JavaScript value `null`. This is the default behavior.

- `MISSING(EXCLUDE)` excludes such missing fields from the output.

The decision to include or exclude a property is made per output line.

`TIMEFORMAT(ISO8601)`

Sets the output format of *all* time stamp fields to `ISO8601`. `TIMEFORMAT(ISO8601)` overrides any format specified for individual fields.

`TIMEFORMAT(ISO8601)` is the only supported value for this parameter.

`ISO8601` outputs time stamps in ISO 8601 date and time of day representation extended format with up to microsecond precision (fractions of a second to a maximum of 6 decimal places), with an optional zone designator. To output time stamps with zone designators, you must also specify the `OUTZONE` parameter. For example:

- With `OUTZONE(Z)`:

  `2018-12-31T05:59:01.03Z`

- With `OUTZONE(+08:00)`:

  `2018-12-31T13:59:00.123456+08:00`

- With no `OUTZONE` parameter (local time):

  `2018-12-31T13:59:00.123456`

If you omit `TIMEFORMAT(ISO8601)`, then time stamps revert to their normal output formats. For details, see the CICS Performance Analyzer product documentation.

In form-based performance list (`CICSPA LIST`) and summary (`CICSPA SUMMARY`) extracts, you can specify the output format of each time stamp field individually. For example:

`START(DATETIM),STOP(TIMET)`

`TIMEFORMAT(ISO8601)` overrides these per-field formats, as if you had specified:

`START(ISO8601),STOP(ISO8601)`

`TIMEFORMAT(ISO8601)` also specifies the format of time stamps in statistics-based data:

- Time stamp fields in statistics list report forms
- `Collection Time` in statistics list, statistic summary, and statistics alert data

`TIMEFORMAT(ISO8601)` enables you to output JSON Lines with all time stamps in this ISO 8601 format *without* updating existing report forms that might present time stamps in other, perhaps more human-readable, formats.

`EOL(CR|CRLF|LF|NEWLINE)|NOEOL`

The end-of-line (EOL) delimiter for each line of JSON Lines. `EOL` appends one or more bytes to the end of each line.

The default behavior `NOEOL` (no delimiter) for the default output character encoding EBCDIC is suitable for writing to record-oriented MVS data sets, where no EOL delimiter is required. However, when writing to z/OS UNIX files or forwarding over TCP, you need to specify an appropriate EOL delimiter.

If you specify the `STREAM` parameter, then the default `EOL` parameter value is `LF`.

The actual bytes appended depend on which `EOL` parameter value you specify, and whether you specify the `ASCII` or `EBCDIC` parameter:

Table 5. Hexadecimal byte values appended by the `EOL` parameter to ASCII or EBCDIC output

EOL parameter value	Description	ASCII	EBCDIC
`CR`	Carriage return (`\r`)	X'0D'	X'0D'
`CRLF`	Carriage return and line feed (`\r\n`): the Windows EOL character pair	X'0D0A'	X'0D25'
`LF`	Line feed (`\n`): the default UNIX EOL character	X'0A'	X'25'
`NEWLINE`	Newline	*Not applicable* (see Note below)	X '15'

Note: ASCII does not contain a direct equivalent to the EBCDIC newline character. If you specify `EOL(NEWLINE)` and `ASCII`, then each line will end with the ASCII substitute character (X '1A'). The substitute character indicates that a character in the EBCDIC source encoding could not be translated to the ASCII target encoding.

When writing to a z/OS UNIX file, the appropriate EOL delimiter depends on whether you want to write output encoded in ASCII or EBCDIC:

- To write ASCII-encoded output to a z/OS UNIX file:

  1. On the DD statement for the file, use the default `FILEDATA` parameter value, `FILEDATA=BINARY`. Do not specify `FILEDATA=TEXT`. If you specify `FILEDATA=TEXT`, the output file will contain an EBCDIC newline character (X'15') at the end of each line, which is not appropriate for an ASCII-encoded file.

  2. On the `CICSPA` report operand, specify the `ASCII` parameter and your choice of `EOL` parameter value: `CR`, `CRLF`, or `LF`.

- To write EBCDIC-encoded output to a z/OS UNIX file, *either*:

- On the DD statement for the file, use the default `FILEDATA` parameter value, `FILEDATA=BINARY` and, on the `CICSPA` report operand, specify `EOL(NEWLINE)`.

- On the DD statement for the file, specify `FILEDATA=TEXT` and, on the `CICSPA` report operand, specify `EOL(NONE)` (the default value).

Do not specify the combination of `FILEDATA=TEXT` on the DD statement and an end-of-line delimiter on the `CICSPA` report operand; if you do, each output record will have *two* EOL delimiters: the delimiter specified by the EOL parameter followed by the EBCDIC newline (X'15') appended by `FILEDATA=TEXT`.

OUTZONE(Z|+*hh*:*mm*|-*hh*:*mm*)

Only applies to time stamp fields with the format `ISO8601`. That is, `OUTZONE` only applies if you specify `TIMEFORMAT(ISO8601)`, which sets the format of all time stamp fields, or to the individual fields for which you have explicitly specified the corresponding format `ISO8601`.

`OUTZONE` appends an ISO 8601 extended format zone designator to time stamps.

`OUTZONE` simply appends the string that you specify to each time stamp. `OUTZONE` does not modify the date and time components of the time stamp.

You must ensure that the zone designator that you specify is correct for your data. In practice, this typically means specifying the time zone of the system that generated the SMF records. In detail:

- For performance data (that is, data from *CMF* SMF records), the `OUTZONE` value that you specify must match either:

  - The time zone of the CICS system that generated the records. More specifically, the date/time offset (SMF field SMFMNDTO) in the original CMF records.

  - The CICS Performance Analyzer `ZONE` (Time Zone) global option, which overrides the date/time offset in the original CMF records

- For statistics data, time stamp fields are stored in the CICS SMF records in local time, with no zone information. You must specify that zone as the `OUTZONE` value.

Examples:

`OUTZONE(+08:00)`

`OUTZONE(Z)`

STREAM

Forwards the JSON Lines or CSV output over TCP using the destination details specified by the `CONNECTION` control operand.

Specifying `STREAM` sets the following default parameters: `ASCII EOL(LF)`. You can override these defaults by explicitly specifying different parameters.

Optionally, each report can specify a `PORT` parameter to override the port number specified by `CONNECTION`.

All `STREAM` parameters in a CICS Performance Analyzer SYSIN data set refer to the same `CONNECTION`; the same destination host name.

The new `STREAM` parameter and the existing `DDNAME` parameter are mutually exclusive. Specify `STREAM` to forward data over TCP or `DDNAME` to extract data to an MVS data set or a z/OS UNIX file. If you specify both `STREAM` and `DDNAME`, then `DDNAME` is ignored.

PORT(*number*)

Only applies if `STREAM` is also specified.

Overrides the port number specified by the `CONNECTION` control operand.

New `ISO8601` format for time stamps in performance forms

The enhancements for JSON Lines output introduce the new value `ISO8601` to the existing list of allowed formats of time stamp fields in performance list and performance summary report forms.

`ISO8601` formats time stamps in ISO 8601 date and time of day representation extended format. If the corresponding `CICSPA` report operand also specifies an `OUTZONE` parameter, then the time stamp includes a trailing zone designator.

Syntax

You can specify `ISO8601` as the value of *format* in the following syntax:

- In **performance list** report forms:

  ```
  LIST( … FIELDS( … STOP|START(format) ) … )
  ```

  For example, `START(ISO8601)`.

- In **performance summary** report forms:

  ```
  SUMMARY( … FIELDS( …
  STOP|START|OSTART(format),[,ASCEND|DESCEND]) … )
  ```

  For example, `STOP(ISO8601,ASCEND)`.

Relationship with the `TIMEFORMAT` parameter

The `ISO8601` format value and the `TIMEFORMAT(ISO8601)` parameter of the CICSPA batch command are closely related.

Specifying `TIMEFORMAT(ISO8601)` overrides the format value that you specify for each time stamp field, as if you had specified `ISO8601` for all time stamp fields.

For details, including example time stamps output values, see "`TIMEFORMAT`" on page 75.

Background and requirement for this time format

Before the JSON Lines enhancements to CICS Performance Analyzer, the existing syntax of the `FIELDS` parameter of the `CICSPA LIST` and `SUMMARY` operands allowed you to specify one of several output formats for time stamp fields. However, those existing formats did not include an ISO 8601 date and time of day representation.

The JSON Lines enhancements to CICS Performance Analyzer were driven by customer requests to export data to analytics platforms. Analytics platforms typically offer flexible options for ingesting time stamp values in many different formats. However, increasingly, analytics platforms and other consumers of JSON Lines favor time stamps in ISO 8601 format.

The new `TIMEFORMAT(ISO8601)` parameter, and corresponding `ISO8601` format that you can specify for each field in performance-based forms, outputs time stamps in an "analytics platform friendly" format.

`STATSALERT` now also outputs CSV

Before the enhancements to output JSON Lines, the `CICSPA STATSALERTS` operand could only output a formatted "human-readable" report of statistics alerts. `STATSALERT` now also offers output to CSV or JSON Lines.

Example JCL

Performance summary

The following JCL (shown with new syntax highlighted) outputs performance summary data to JSON Lines:

```
//CPAUID    JOB NOTIFY=&SYSUID
//CICSPA    EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//EXPT0001 DD  SYSOUT=*
//CPAJEX01 DD  SYSOUT=*
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       SUMMARY(OUTPUT(EXPT0001),
               DDNAME(CPAJEX01),
               JSON,
               CODE('perf_summ'),
               TIMEFORMAT(ISO8601),
               OUTZONE(+08:00),
               MISSING(EXCLUDE),
               INTERVAL(00:15:00),
               FIELDS(START(ISO8601,ASCEND),
                      APPLID(ASCEND),
                      TRAN(ASCEND),
                      TASKCNT,
                      DISPATCH(TIME(TOT)),
                      CPU(TIME(TOT)),
                      SUSPEND(TIME(TOT)),
                      DISPWAIT(TIME(TOT)),
                      DSPDELAY(TIME(TOT)),
                      RESPONSE(TIME(TOT)))))
/*
```

In this example, `TIMEFORMAT(ISO8601)` is unnecessary, because the only time stamp field, `START`, already specifies the format `ISO8601`.

Here is a snippet of a single line of JSON Lines output (in ddname CPAJEX01, shown with line breaks and indenting for readability):

```
{
  "code":"perf_summ",
  "Start Interval":"2019-10-07T10:30:00+08:00",
  "APPLID":"FUWTCIC",
  "Tran":"CEMT",
  "#Tasks":1,
  "Dispatch Time Total":0.2610,
  "User CPU Time Total":0.0998,
  …
}
```

Writing to a z/OS UNIX file as EBCDIC

The following JCL writes JSON Lines to a z/OS UNIX file in EBCDIC, with a newline character (X'15') at the end of each line.

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//EXPT0001 DD  SYSOUT=*
//CPAJEX01 DD  PATH='/u/uid/cicspa_perf_summ_ebcdic.txt',
//            FILEDATA=TEXT,
//            PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//            PATHDISP=(KEEP,DELETE),
//            PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       SUMMARY(OUTPUT(EXPT0001),
               DDNAME(CPAJEX01),
               JSON,
               CODE('perf_summ'),
               TIMEFORMAT(ISO8601),
               OUTZONE(+08:00),
               MISSING(EXCLUDE),
               INTERVAL(00:15:00),
               FIELDS( … ))
/*
```

The newline EOL delimiter is a result of specifying the `FILEDATA=TEXT` parameter on the output DD statement. The `CICSPA` report operand does not specify an `EOL` parameter.

Writing to a z/OS UNIX file as ASCII

The following JCL writes JSON Lines to a z/OS UNIX file in ASCII, with a line feed character (X'0A') at the end of each line.

```
//CPAUID    JOB NOTIFY=&SYSUID
//CICSPA    EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//EXPT0001 DD  SYSOUT=*
//CPAJEX01 DD  PATH='/u/uid/cicspa_perf_summ_ascii.txt',
//             FILEDATA=BINARY,
//             PATHOPTS=(OWRONLY,OCREAT,OEXCL),
//             PATHDISP=(KEEP,DELETE),
//             PATHMODE=(SIRUSR,SIWUSR,SIRGRP)
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       SUMMARY(OUTPUT(EXPT0001),
               DDNAME(CPAJEX01),
               JSON,
               ASCII,
               EOL(LF),
               CODE('perf_summ'),
               TIMEFORMAT(ISO8601),
               OUTZONE(+08:00),
               MISSING(EXCLUDE),
               INTERVAL(00:15:00),
               FIELDS( … ))
/*
```

Note that the DD statement specifies `FILEDATA=BINARY` while the `CICSPA` report operand specifies `ASCII` and `EOL(LF)`.

In practice, in this case, you can omit `FILEDATA=BINARY` from the DD statement, because `FILEDATA=BINARY` is the default behavior.

Forwarding over TCP

The following JCL forwards JSON Lines in ASCII over unsecure TCP to a remote analytics platform.

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//EXPT0001 DD  SYSOUT=*
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       CONNECTION(HOST(analytics),
                  PORT(6789)),
       SUMMARY(OUTPUT(EXPT0001),
               STREAM,
               JSON,
               CODE('perf_summ'),
               TIMEFORMAT(ISO8601),
               OUTZONE(+08:00),
               MISSING(EXCLUDE),
               INTERVAL(00:15:00),
               FIELDS( … ))
/*
```

Note that this JCL contains no DD statement for the output JSON Lines. Instead, the `CICSPA SUMMARY` report operand specifies the `STREAM` parameter, which implicitly refers to the `CONNECTION` control operand.

For a secure TCP port, adding the following parameters to the CONNECTION control operand in the previous example:

```
SECURITY(TLS*),
FIPS,
KEYRING('my/cpa.analytics')
```

where `my/cpa.analytics` is the name of a SAF key ring owned by the user who submits the JCL.

Statistics alerts

The following JCL (shown with new syntax highlighted) outputs statistics alerts to JSON Lines:

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//CPAHDBRG DD  DISP=SHR,DSN=<my.cpa.hlq>.REPOSTRY
//SYSPRINT DD  SYSOUT=*
//STEX0001 DD  SYSOUT=*
//CPAOSX01 DD  SYSOUT=*
//SYSIN DD *
CICSPA IN(SMFIN001),
       STATSALERT(OUTPUT(STEX0001),
                  DDNAME(CPAOSX01),
                  JSON,
                  CODE('stat_alert'),
                  TIMEFORMAT(ISO8601),
                  OUTZONE(+08:00),
                  MISSING(EXCLUDE),
                  STALTDEF(ALERTS1))
/*
```

This example JCL uses a set of statistics alert definitions named ALERTS1 stored in the CICS Performance Analyzer repository `<my.cpa.hlq>.REPOSTRY`.

Here is an example snippet of a single line of JSON Lines output (in ddname CPAOSX01, shown with line breaks and indenting for readability):

```
{
  "code":"stat_alert",
  "Collection Time":"2019-08-31T00:00:00+08:00",
  "Sev":"W",
  "Alert":"EDSA peak usage as % of EDSALIM",
  "APPLID":"FUWFWFR",
  "Image":"FTS1",
  "System Type":"TS",
  "Interval Type":"EOD",
  "Threshold":">=90",
  "Actual":92,
  "Resource":"",
  "Resource value":"",
  "stid":"029A"
}
```

Statistics list

The following JCL (shown with new syntax highlighted) outputs statistics list data to JSON Lines:

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//STEX0001 DD  SYSOUT=*
//CPAOSX01 DD  SYSOUT=*
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       STATISTICSLIST(OUTPUT(STEX0001),
                      DDNAME(CPAOSX01),
                      JSON,
                      CODE('stat_list_010a'),
                      TIMEFORMAT(ISO8601),
                      OUTZONE(+08:00),
                      MISSING(EXCLUDE),
                      TYPE(EOD,INT,USS,REQ,RRT),
                      FIELDS(XMGNUM,
                      XMGMXT,
                      XMGTAMXT,
                      XMGPAT,
                      XMGPQT,
                      XMGTAT,
                      XMGTDT))
/*
```

Here is a single line of JSON Lines output (in ddname CPAOSX01, shown with line breaks and indenting for readability):

```
{
  "code":"stat_list_010a",
  "APPLID":"FUWTCIC",
  "Image":"FTS1",
  "Type":"EOD",
  "Collection Time":"2019-08-31T00:00:00+08:00",
  "Transactions":24471,
  "Current MAXTASK":1500,
  "Times at MAXTASK":0,
  "Peak Active User Transactions":30,
  "Peak Queued User Transactions":0,
  "Total Active User Transactions":24469,
  "Total Delayed User Transactions":0,
  "stid":"010A"
}
```

Statistics summary

The following JCL (shown with new syntax highlighted) outputs statistics summary data to JSON Lines:

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//STEX000A DD  SYSOUT=*
//CPAOSX01 DD  SYSOUT=*
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       STATISTICSSUMMARY(OUTPUT(STEX000A),
                         DDNAME(CPAOSX01),
                         JSON,
                         CODE('stat_summ_composite'),
                         TIMEFORMAT(ISO8601),
                         OUTZONE(+08:00),
                         MISSING(EXCLUDE),
                         TYPE(EOD,INT,USS,RRT),
                         FIELDS(APPLID(ASCEND),
                         COLLECTTIME(ASCEND),
                         XMGNUM(TOT),
                         XMGMXT(FIN),
                         XMGTAMXT(TOT),
                         XMGPAT(MAX),
                         XMGPQT(MAX),
                         XMGTAT(TOT),
                         XMGTDT(TOT),
                         NQGTNQSI(TOT),
                         NQGTNQSW(TOT),
                         … )
/*
```

Here is a snippet of a single line of JSON Lines output (in ddname CPAOSX01, shown with line breaks and indenting for readability):

```
{
  "code":"stat_summ_composite",
  "APPLID":"FUWFWAR",
  "Collection Time":"2019-08-31T00:00:00+08:00",
  "Tot Transactions":36380,
  "Fin Current MAXTASK":1500,
  "Tot Times at MAXTASK":0,
  "Max Peak Active User Transactions":467,
  "Max Peak Queued User Transactions":0,
  "Tot Total Active User Transactions":36343,
  "Tot Total Delayed User Transactions":0,
  "Tot ENQs Issued":72685,
  "Tot ENQs Waited":456,
   …
  "stid":""
}
```

Performance list

The following JCL (shown with new syntax <mark>highlighted</mark>) outputs performance list data to JSON Lines:

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//EXPT0001 DD  SYSOUT=*
//CPAOEX01 DD  SYSOUT=*
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       LIST(OUTPUT(EXPT0001),
            DDNAME(CPAOEX01),
            JSON,
            CODE('perf_list'),
            TIMEFORMAT(ISO8601),
            OUTZONE(+08:00),
            MISSING(EXCLUDE),
            FIELDS(APPLID,
                   TRAN,
                   START(ISO8601),
                   DISPATCH(TIME),
                   CPU(TIME),
                   SUSPEND(TIME),
                   TASKNO,
                   DISPWAIT(TIME),
                   … )
/*
```

Here is a snippet of a single line of JSON Lines output (in ddname CPAOSX01, shown with line breaks and indenting for readability):

```
{
  "code":"perf_list",
  "APPLID":"FUWTCIC",
  "Tran":"CWXN",
  "Start":"2019-08-31T00:00:00.885283+08:00",
  "Dispatch Time":0.0687,
  "User CPU Time":0.0011,
  "Suspend Time":0.7523,
  "TaskNo":"54272",
  "DispWait Time":0.180,
  …
}
```

z/OS Connect list and summary

The following JCL (shown with new syntax highlighted) outputs z/OS Connect list or summary data to JSON Lines:

```
//CPAUID   JOB NOTIFY=&SYSUID
//CICSPA   EXEC PGM=CPAMAIN
//STEPLIB  DD  DISP=SHR,DSN=<cpa.hlq>.SCPALINK
//SMFIN001 DD  DISP=SHR,DSN=<smf.input>
//SYSPRINT DD  SYSOUT=*
//EXPT0001 DD  SYSOUT=*
//CPAOEX01 DD  SYSOUT=*
//SYSIN    DD  *
CICSPA IN(SMFIN001),
       ZCON(OUTPUT(ZCONN0001),
            ZCNDDNM(CPAOZX01),
            CMFDDNM(CPAOZX02),
            EXTERNAL(CPAXW001),
            LIST,
            CICSSP,
            CMF,
            ZCHEADERS,
            JSON,
            CODECMF('zcon_perf_summ'),
            CODEZCON('zcon_req_summ'),
            LABELS(ASIS),
            TIMEFORMAT(ISO8601)
            OUTZONE(+08:00),
            MISSING(INCLUDE),
            EBCDIC,
            NOEOL)
/*
```

Here is a snippet of a single line of JSON Lines output (in ddname ZCNDDNM(CPAOZX01), shown with line breaks and indenting for readability):

```
{
  "code":"ZCON_REQUEST_LIST",
  "Jobname": "ZOSCONN",
  "Jobid": "STC15062",
  "SID": "MV22",
  "SSI": "ZCON",
  "System": "MV22",
  "Sysplex": "MV22",
  "Date": "2021-01-21T17:31:17.498741+08:00",
  ….
  "ZC Tracking Token": "MV22    MV22    D927CB92BDF75920"
}
```

Configuring the Elastic Stack to use JSON Lines from CICS Performance Analyzer

Configuring the Elastic Stack to use JSON Lines from CICS Performance Analyzer involves three steps:

1. **Configuring Elasticsearch with an index template**
   to map all string values to the keyword data type

2. **Configuring Logstash**
   to ingest JSON Lines over TCP

3. **Creating an index pattern, or patterns, in Kibana**
   for data from CICS Performance Analyzer

The configuration described here has been tested with Elastic 7.4.0.

These instructions assume that you understand general Elastic configuration concepts.

As of September 2021, there is no set of IBM-supplied Kibana dashboards for visualizing data from CICS Performance Analyzer; no equivalent of the IBM-supplied Splunk app.

Running the Elastic Stack in a Docker container

If you have Docker, then you can use the following `docker run` command to start a container running the Elastic Stack (Elasticsearch, Logstash, and Kibana):

```
docker run -d -p 15601:5601 -p 19200:9200 -p 15046:5046 --name
elastic-740 sebp/elk:740
```

where `-p 15046:5046` defines the port mapping for the TCP port on which Logstash will be listening for JSON Lines from CICS Performance Analyzer.

Configuring Elasticsearch with an index template

By default, from Elastic 5.0, Elasticsearch maps string fields to the `text` data type. Elasticsearch parses the contents of text fields into tokens for full-text search. You might not want that default behavior. Many string fields in log data are names or identifiers. It makes more sense to search these fields as whole values, so the `keyword` data type is a better choice. You can configure Elasticsearch by creating an index template that maps string fields to the `keyword` data type.

The following Elasticsearch REST API request body contains an index template that maps all string fields in the index pattern "cicspa-*" to the `keyword` data type:

```
{
  "index_patterns": "cicspa-*",
  "mappings": {
    "dynamic_templates": [ {
      "string_fields": {
        "match": "*",
        "match_mapping_type": "string",
        "mapping": {
        "type": "keyword"
        }
      }
    } ]
  }
}
```

Figure 27. Elasticsearch index template (REST API request body)

Save the JSON in the previous example to a file. For example:

`cicspa-index-template.json`

Get the curl tool.

Use the following curl command to define the index template to Elasticsearch:

```
curl -XPUT localhost:9200/_template/cicspa --header "Content-Type:
application/json" --data-binary @cicspa-index-template.json
```

Figure 28. curl command to create an Elasticsearch index template

This example curl command assumes that Elasticsearch is running on the local computer and that the index template JSON file is in the current directory.

Configuring Logstash

The following Logstash config ingests JSON Lines over TCP from CICS Performance Analyzer:

```
input {
  tcp {
    port => 5046
    codec => line
  }
}
filter {
  grok {
    match => { "message" => "%{TIMESTAMP_ISO8601:_time}" }
  }
  date {
    match => [ _time, ISO8601 ]
  }
  json {
    source => "message"
    remove_field => [ _time, message ]
  }
  # Convert code value to lowercase, to ensure lowercase index names
  mutate {
    lowercase => [ "code" ]
  }
  # Convert large numeric values to integers
  ruby {
    code => '
      event.to_hash.each { |k, v|
        if v.is_a? Numeric
          if v > 99999999
            event.set(k, v.to_i)
          end
        end
      }
    '
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "cicspa-%{code}-%{+YYYY.MM.dd}"
    # The following manage_template setting assumes that
    # you have created an index template for the cicspa-* index pattern.
    manage_template => false
  }
}
```

Figure 89. Logstash config to ingest JSON Lines over TCP

This config initially uses the `line` codec to read each line of the incoming JSON Lines as plain text. The config passes each line into a `message` field. The config searches the

message for the first value that matches an ISO 8601 time stamp, and then sets the value of the Logstash `@timestamp` field to that value.

After extracting the time stamp, the config parses the message as JSON.

The config uses Ruby code to ensure that large numeric values are output as integers; that is, without a trailing `.0`. This code is a workaround for Logstash behavior described in the Logstash GitHub project issue #12398, "Logstash converts integer value in scientific notation to float, causing mapping error in Elasticsearch".

Finally, the config forwards the event to Elasticsearch.

This example config assumes that:

- You are sending JSON Lines that contains ISO 8601-format time stamps, where the first time stamp is the one you want to use as the event time stamp.

- Each incoming line of JSON Lines contains a `code` property.

If the JSON Lines from CICS Performance Analyzer that you want to ingest in Elastic is limited to data types that always use the same fixed field name for the event time stamp, such as `Start Interval`, then you can use the following more concise Logstash config that uses the `json_lines` codec:

```
input {
  tcp {
    port => 5046
    codec => json_lines
  }
}
filter {
  date {
    match => ["Start Interval", "ISO8601"]
  }
  mutate {
    lowercase => [ "code" ]
  }
  ruby {
    code => '
      event.to_hash.each { |k, v|
        if v.is_a? Numeric
          if v > 99999999
            event.set(k, v.to_i)
          end
        end
      }
    '
  }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "cicspa-%{code}-%{+YYYY.MM.dd}"
    # The following manage_template setting assumes that
    # you have created an index template for the cicspa-* index pattern.
    manage_template => false
  }
}
```

Figure 30. Logstash config to ingest JSON Lines over TCP: fixed event time stamp field name

For more information about event time stamp field names, see "Event time stamps" on page 51.

Creating an index pattern in Kibana

After configuring Elasticsearch and Logstash, forward some JSON Lines to Elasticsearch.

Then, in Kibana, create an index pattern:

1. Select **Management** (the "cog" icon) > **Index Patterns**

2. Click **Create index pattern**

3. In the Search box, enter:

    `cicspa-*`

    This is just an initial example index pattern. Typically, you would create more-specific index patterns: one for each type of JSON Lines from CICS Performance Analyzer (each `code` property value).

4. Specify **@timestamp** as the Time Filter field name

Now you can create visualizations in Kibana using this data.

Managing JSON Lines output from CICS Performance Analyzer

You—the CICS Performance Analyzer user—decide when and how often to run the CICS Performance Analyzer batch jobs that output JSON Lines. In the JCL for those jobs, you control what data is output, and to where.

You can use `CICSPA` operands to control the time range of the data and to filter the data by various other criteria; for example, to only output data for transactions that are of interest to you.

Managing data volume

When deciding what types of data to forward from CICS Performance Analyzer to an analytics platform, it's useful to understand the relative data volumes involved.

Performance list data—one event per transaction—is potentially very high volume. Other data types—performance summary, statistics—are relatively low volume.

You can use CICS Performance Analyzer to reduce data volume by:

- Using report forms with very few fields

- Using selection criteria to only forward events for selected applids and/or transaction identifiers

- Forwarding performance list data only for short time periods of special interest rather than continuously, 24/7

- Using *summary* report forms to specify longer reporting intervals

Summarization is a key strength of CICS Performance Analyzer: you can forward useful data to analytics platforms at relatively low volumes.

Developer notes

Issues with `multiselect_all.js`

The custom `appserver/static/multiselect_all.js` JavaScript file supplied with the sample CICS PA app is supposed to make multiselect inputs (dropdowns that support multiple selections) with an "All" option behave intuitively: to disallow "All" with other values. In testing, this worked as expected in the Splunk 7.3.0 environment in which the app was originally developed. However, when developing this new dashboard in Splunk 8.1.0, it was observed that this behavior is no longer consistent. This issue needs further investigation.

`cicspa_*` macro dependency

All of the dashboards in the sample CICS PA app use macros that are defined in the app.

References

JSON.org website
http://www.json.org

The JSON Data Interchange Syntax (ECMA 404 standard)
http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

JSON Lines (standard)
http://jsonlines.org/

Notices

This information was developed for products and services offered in the U.S.A.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new

editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

Trademarks

Terms and conditions for product documentation

derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights: Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at http://www.ibm.com/privacy and the section titled "Cookies, Web Beacons, and Other Technologies" in IBM's Online Privacy Statement at http://www.ibm.com/privacy/details. Also, see the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.