IBM Common Data Provider for z Systems:

Splunk® Integration Example

This document can be found on the web at www.ibm.com/support/techdocs

Version Date: March 2018

Mike Bonett

IBM - Washington Systems Center (bonett@us.ibm.com)

Special Notices

This document reflects the IBM Washington Systems Center's understanding on integrating the IBM Common Data Provider (CDPz) for z Systems with Splunk. It was produced and reviewed by the members of the IBM Washington Systems Center organization. This document is presented "As-Is" and IBM does not assume responsibility for the statements expressed herein. It reflects the opinions of the IBM Washington Systems Center. These opinions are based on hands on experiences with the IBM Common Data Provider for z Systems. If you have questions about the contents of this document, please direct them to Mike Bonett (bonett@us.ibm.com).

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the 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.

Trademarks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States and/or other countries: CICS, IBM, NetView, System z, WebSphere, z/OS, z/VM, z Systems. A full list of U.S. trademarks owned by IBM may be found at http://www.ibm.com/legal/copytrade.shtml.

Elasticsearch is a trademark of Elatsticsearch BV, registered in the U.S. and in other countries
Kibana is a trademark of Elatsticsearch BV, registered in the U.S. and in other countries
Logstash is a trademark of Elatsticsearch BV, registered in the U.S. and in other countries
Splunk is a trademark of Splunk, Inc., registered in the U.S. and in other countries.
Microsoft, Windows, and the Windows logo are registered trademarks of Microsoft Corporation in the United States and/or other countries.
UNIX is a registered trademark in the United States and other countries licensed exclusively through The Open Group.
LINUX and Linux are registered trademarks of Linus Torvalds.
Java and all Java-based trademarks and logos are registered trademarks of Oracle and/or its affiliates.
Other company, product and service names may be trademarks or service marks of others.

Acknowledgements

Many thanks to the following people for reviewing this paper and/or providing input to the intformation:

- Barry Lamkin, IBM Global Markets - Systems SW Sales
- Mike Sine, IBM Washington Systems Center

Contents

Introduction

The IBM Common Data Provider for z Systems (which may be referred to as "CDPz" in the document) provides a streamlined method of capturing z/OS performance and operational data for use by analytics products. The data sources CDPz can capture and forward to the target include:

- SYSLOG/OPERLOG
- JOBLOGs, with custom mapping of WebSphere and CICS JOBLOGs
- SMF Records
- NetView for z/OS logs
- Unix System Services syslogd and files

CDPz can send the data to any target that can receive data over an IP port. It provides custom integration for the following software products:

- IBM Operations Analytics for z Systems
- The Elastic Stack (ElasticSearch, Logstash, Kibana)
- Splunk

Details on CDPz can be found at https://www.ibm.com/support/knowledgecenter/SSGE3R_1.1.0/com.ibm.cdpz.doc/welcome.html

This paper will walk through an example of integrating the Common Data Provider for z Systems with Splunk, using the z/OS SYSLOG as the data source to be send to Splunk.

Splunk integration components

Two components, provided by CDPz, are used on the Splunk platform to integrate CDPz output with Splunk:

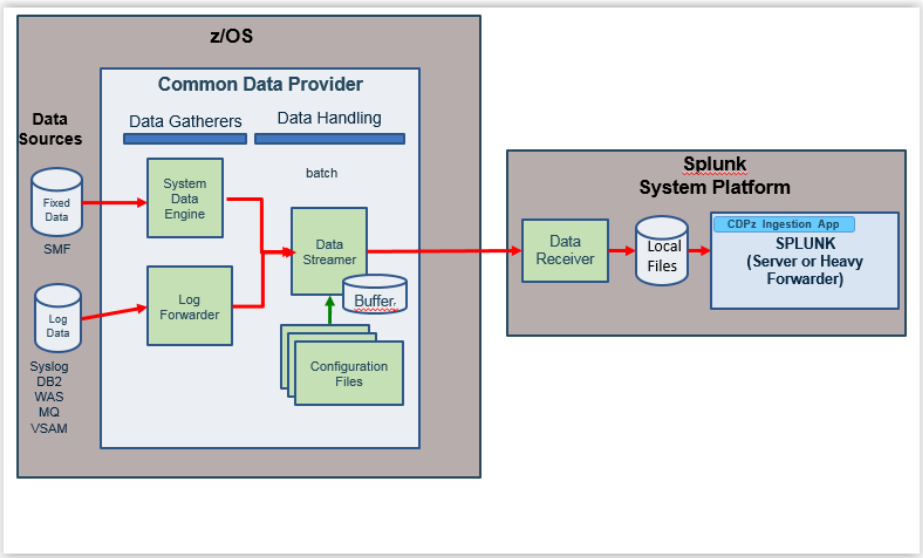
- A Data Receiver. This is a Java program that receives information from CDPz and writes it to a file on the platform it is running on.
- A Splunk Ingestion Kit. This is a Splunk data ingestion app that is installed on the Splunk component that initially ingests the data – either the Splunk server itself, in a scaled-out environment, a Splunk server configured as a Heavy Forwarder that receives the data, parses it, and sends it to a Splunk indexer for access/searching by the central Splunk Server.

The Data Receiver must be installed on the same platform as the Splunk Server or a Splunk Heavy Forwarder.

On the z/OS side, the System Data Engine, Log Forwarder and Data Streamer components of CDPz must be configured appropriately and stated (they run as started tasks or batch jobs) to send data to the Data Receiver:

- If SMF or customer data is to be forwarded, the System Data Engine must be used to access those data sources.
- If SYSLOG, OPERLOG, or data from NetView, CICS, or JOBLOGs is to be forwarded, the Log Forwarder must be used to access those data sources.
- The Data Streamer will receive the data from the System Data Engine.

An example of the architecture is shown in this picture:



The steps to enable this architecture are:

- Install and configure the Data Receiver on the target Splunk platform.
- Install the CDPz Splunk Ingestion Kit app into the Splunk server/Heavy Forwarder running on the Splunk Platform.
- Configure the Common Data Provider for z Systems components to extract, format, and forward data to the Data Receiver.

The rest of this paper will illustrate these steps by applying them to an example implementation.

An example integration environment

To better illustrate the integration, the rest of this paper walks through an example implementation. For this example, SYSLOG data was captured and sent to a Splunk server running on a Linux systems platform.

Installing and configuring the Data Receiver

The Data Receiver is a .jar file (DataReceiver.jar) that is downloaded to the Splunk platform. A Java runtime environment must be installed; Java 1.8 was used in this example.

The DataReceiver.jar file can be placed in any location; in this example **/opt/cdpdatareceiver** is used.

Two environment variables must be defined before the Data Receiver is started:

- CDPDR_HOME – the location of the DataReceiver.jar file (in this example **/opt/cdpdatareceiver**).
- CDPDR_PATH – the location where the Data Receiver will write the data received from CDPz into files that will be read by Splunk (in this example **/opt/cdpdatareceiver/data**).

A **cdpdr.properties** file should be created in the CDPRDR_HOME directory to define the Data Receiver parameters. Here is an example of the properties file:

```
port = 19080
cycle = 3
trace = y
ssl = n
```

- The **port** parameter is the IP port the Data Receiver will listen on to receive data from CDPz. In this example we use port 19080. Whatever port is chosen should be opened through any local firewall.
- The **cycle** parameter is the number if cycles used for the files storing the data records received from CDPz. This number determines how often the Data Receiver will cycle through files as it stores the records. For example, with a value of 3, it will initially create 3 files suffixed with -0, -1, and -2. At the top of each hour it will switch to the next file in the sequence to store incoming data. When it reaches the end it will begin again with the first file, thus limiting the amount of disk space used. It is best to start with this default, observe the pattern of received data and subsequent file sizes, and then adjust as needed for optimal disk space usage.
- The **trace** parameter indicates if tracing is to be turned on or off. This is normally used for debugging purposes and should not be active for normal operations.

- The **ssl** parameter indicates if Secure Sockets Layer (SSL) will be used between the Data Receiver and the CDPz Data Streamer. While this example does not use SSL, it is a good practice to use it. To use SSL some additional steps must be performed (details are in the Common Data Provider Users Guide, which is in the Knowledge Center link above).
    - Download the setupDataReceiverSSL.sh script from the CDPz installation directory on z/OS to the Data Receiver platform (NOTE: if using windows download setupDataReceiverSSL.bat)
    - Make the file executable.
    - Ensure that the CDPDR_HOME and CDPDR_PATH environment variables are defined.
    - Ensure that the JAVA_HOME variable points to the Java home directory.
    - Execute the setupDataReceiverSSL.sh with the input parameters documented in the script. The script will generate a local keystore and a public certificate.
    - Upload the public certificate.
    - Execute the importCertificate.sh script (provided within the CDPz installation directory tree) to import the certificate into the CDPz Data Streamer's keystore.

NOTE: the parameters can be overridden when the Data Receiver is started using command line parameters; consult the Common Data Provider Users Guide for details.

Using the cdpdr.properties file, the Data Receiver can be started with the command

```
java -ja  -Dfile.encoding=UTF-8 DataReceiver.jar
```

The Java parameter –Dfile.encoding=UTF-8 is required, as that is the encoding CDPz uses to send data to the Data Receiver.

The command can be placed in a script to start the Data Receiver as a background process (and also to perform any other setup actions such as setting the location of the java executable). Such a script can be used as a basis to define the data streamer as a system service.

The Data Receiver will emit a startup message similar to the following:

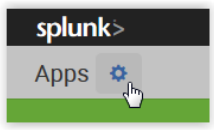
```
<timestamp> HBO6064I CDP Data Receiver - Version: V1R1M0, Build ID:
<ID>,  APAR: <APAR ID>
```

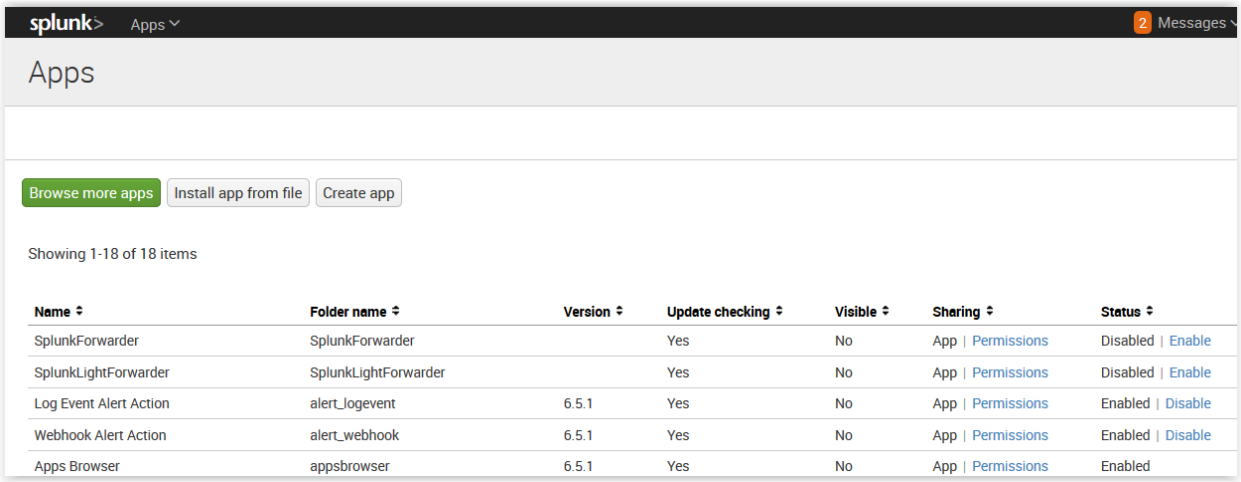
Its process listening on the input port – in this example port 19080 – will verify that it is active and waiting to process data.

Installing the CDPz Splunk Ingestion Kit

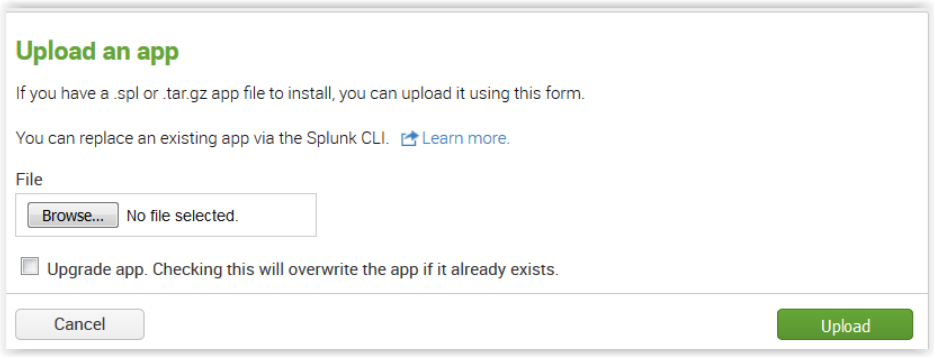
CDPz provides a Splunk Ingestion Kit which consists of a Splunk app file (a compressed tar.gz file with the .spl file extension) which is imported into Splunk. It contains the various definitions required to Splunk to process the incoming data. Versions are provided for both UNIX/Linux and Windows platforms.

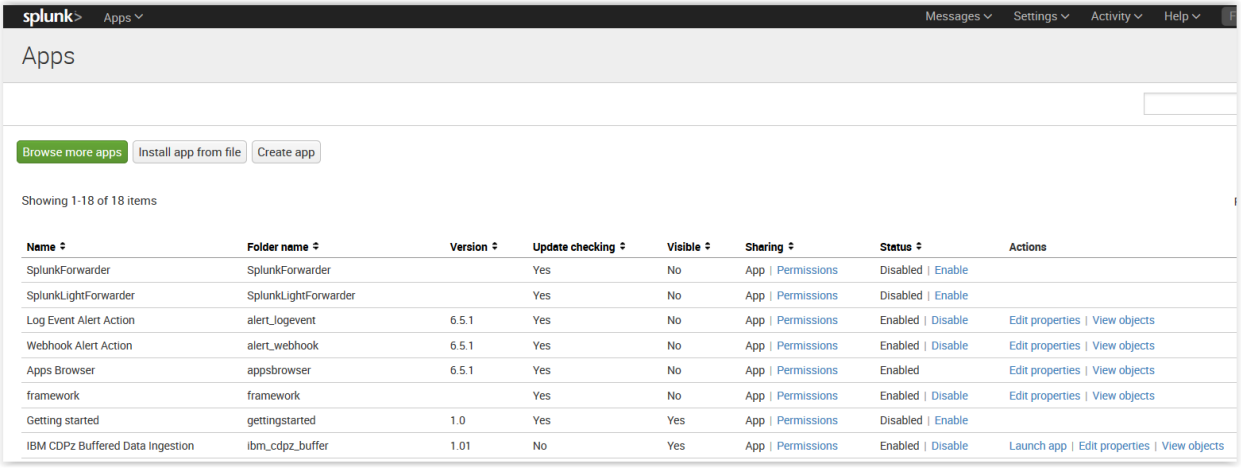
For this example the UNIX/Linux .spl file (ibm_cdpz_buffer_nix.spl) was installed in Splunk with the following steps:

- Download the .spl file to a local workstation (one which can provide a browser session to log into Splunk).
- Log in to the Splunk user interface on the same platform where the Data Receiver is running.
- On the left side of the menu, click the blue gear next to the word Apps:
- 

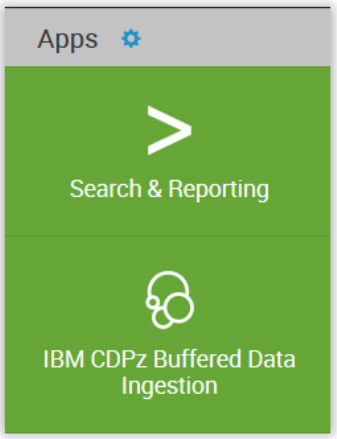
- This will bring up the list of installed apps:
- 

- Click on the Install app from file button. This will display an upload window:

- 

- Browse to where the .spl file was downloaded and select it. When prompted, select Enable Now.
- When the upload and processing is complete, the integration app will be shown as IBM CDPz Buffered Ingestion:
- 

-

- On the Splunk main menu, the App will also be shown in the Apps list:
- 

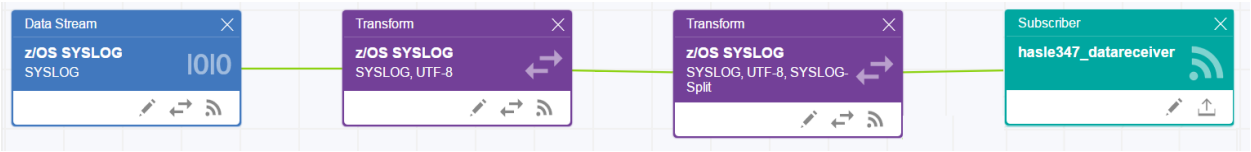
Configuring the IBM Common Data Provider for z Systems

A flow must be defined in IBM Common Data Provider for z Systems to identify

- The data source that will be used
- Any required transformations
- The output to where the data is sent.

These actions are performed in the CDPz configuration tool, which is installed as a plugin to the z/OS Systems Management Facility (z/OSMF). The tool allows a graphical representation of the flow to be built, which then generates the files that the CDPz data streamer will read at its startup and use to collect, transform, and send data.

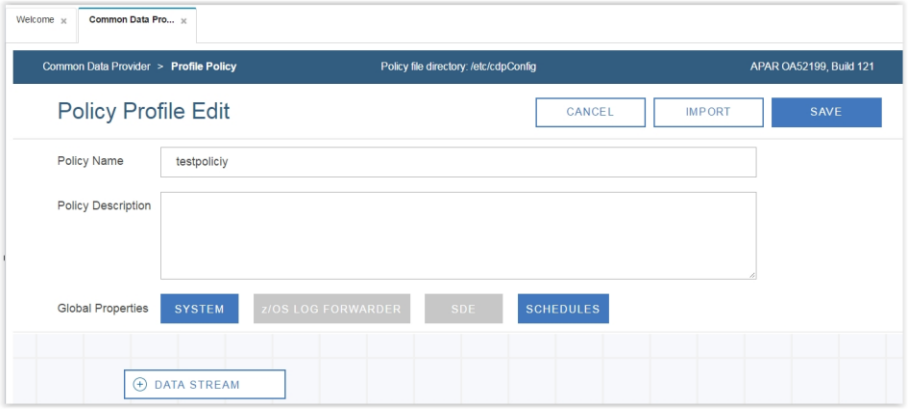
A graphical representation of the flow would be as follows:



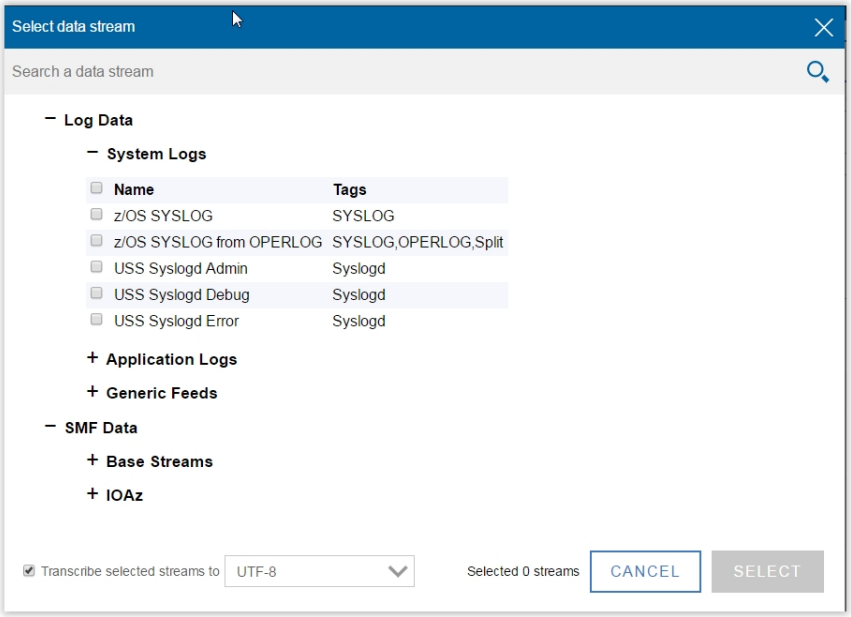
This shows a Data Stream Source (SYSLOG) two transforms (UTF8 and Split) and a subscriber (hasle347_datareceiver). Each component is defined, in order, as follows:

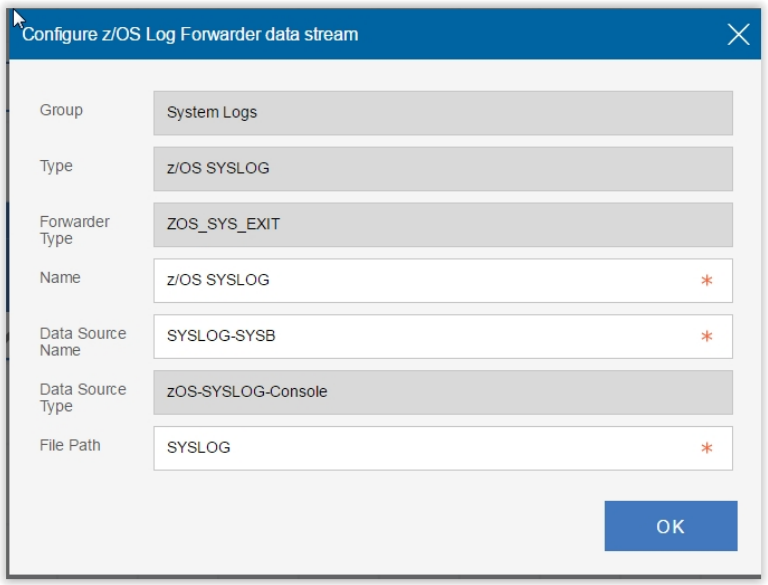
- Data Stream and UTF-8 transform

  When editing a policy, a data stream is added by clicking on the + Data Stream Icon in the Policy editor:



  On the subsequent windows the z/OS SYSLOG data stream was selected, which captures data from the Common Data Provider Log Forwarder, as well as the option to transform to the UTF-8 code page:

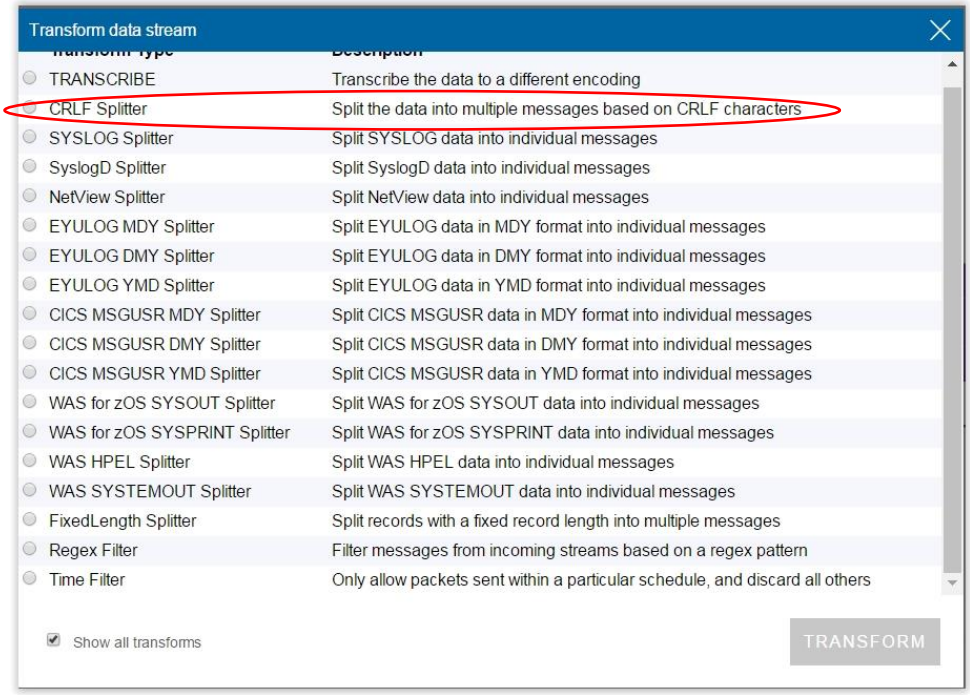




These selections generate both the Data Stream and UTF-8 Transform icons in the workflow.

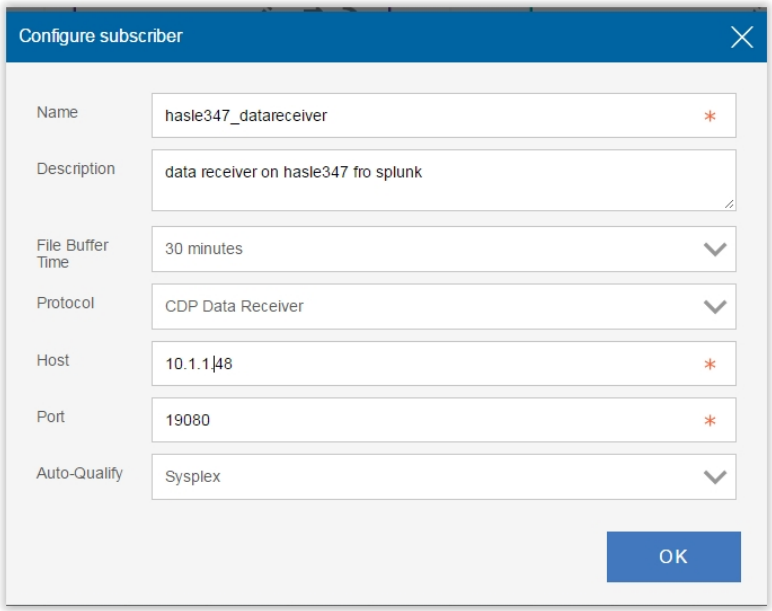
- Transform – Split

  The Split transform is required to split the data so that each message is sent as a separate record. In the policy editor it is created by clicking on the  icon on the UTF-8 transform icon, which creates a new transform placed after the UTF-8 transform in the flow. For this transform the SYSLOG Splitter option is used:



- Subscriber – hasle347_datareceiver

  The subscriber icon is creating by clicking on the  icon in the Splitter transform and selecting Add Subscriber. The subscriber can then be configured:



  - The Protocol attribute must be CDP Data Receiver (or CDP Data Receiver SSL, if using SSL).
  - The Host attribute must be the hostname or IP address where the Data Receiver is running.

- The Port attribute must match the port used by the Data Receiver.

After the flow is defined in the policy, the Common Data Provider Data Streaming must be restarted to pick up the additions.

Results

In this example, when all of these components are running, z/OS SYSLOG messages will appear in Splunk:

- The CDPz Log Forwarder on (z/OS) which captures the SYSLOG messages.
- The CDPz Data Streamer (on z/OS) which uses the policy flow defined above to send the messages to the Data Receiver.
- The CDPz Data Receiver (on the Splunk platform, Linux in this example), which receives the forwarded data and writes it to files.
- The Splunk Server (if the Data Receiver is running on the Splunk Server platform, as in this example) or the Splunk Heavy Forwarder (if the Data Receiver is running on a Heavy Forwarder platform).
- The CDPz Splunk Ingestion Kit, installed and enabled on the Splunk Server or Splunk Heavy Forwarder, which contains the definitions to map the incoming data to fields for use within Splunk.

When messages arrive to the Data Receiver, they are written to files in the directory defined in the CDPDR_PATH variable. Here is an example of a message record within that file:

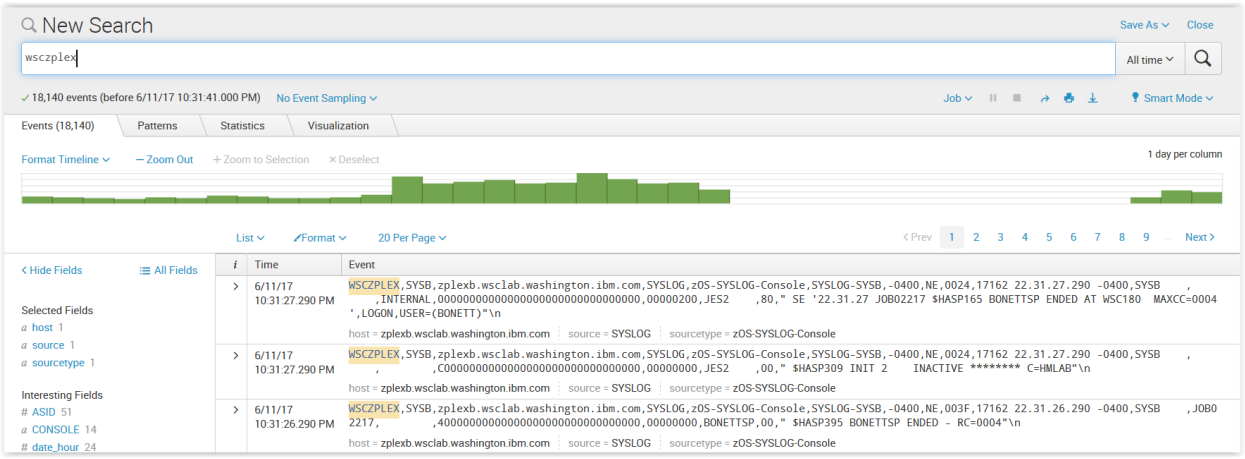
```
WSCZPLEX,SYSB,zplexb.wsclab.washington.ibm.com,SYSLOG,zOS-SYSLOG-
Console,SYSLOG-SYSB,-0400,NE,004F,17162 21.10.31.110 -0400,SYSB
,STC02212,        ,00000000000000000000000000000000,00000200,BPXAS
,00," BPXP024I BPXAS INITIATOR STARTED ON BEHALF OF JOB PFA RUNNING IN
ASID 0059"\n
```

Each record is in Comma Separated Value (CSV) format. The first seven fields of each record identify:

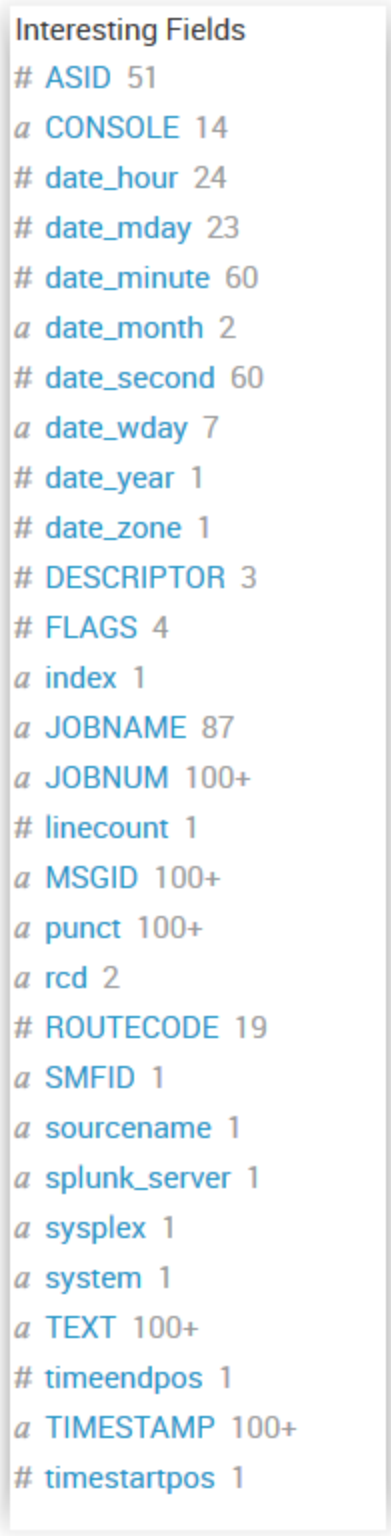
- Sysplex Name
- System Name
- Hostname
- Path (SYSLOG for a SYSLOG data stream source)
- Source Type (zOS-SYSLOG-CONSOLE for a SYSLOG data stream source)
- Source Name (the name of the source as defined in the Policy data stream definition; in this example, SYSLOG-SYSB)
- Time zone offset from GMT time

The fields after that are parsed from the message contents. We can see the field names in the Splunk user interface. There are many different ways to access the messages. In this example the following steps were performed:

- Log on to the Splunk user interface.
- In the Apps list on the left, click ‘IBM CDPz Buffered Ingestion” to bring up the search window.
- Enter a search string – such as the name of the sysplex – and the records that contain that string will be displayed, the most recent first:



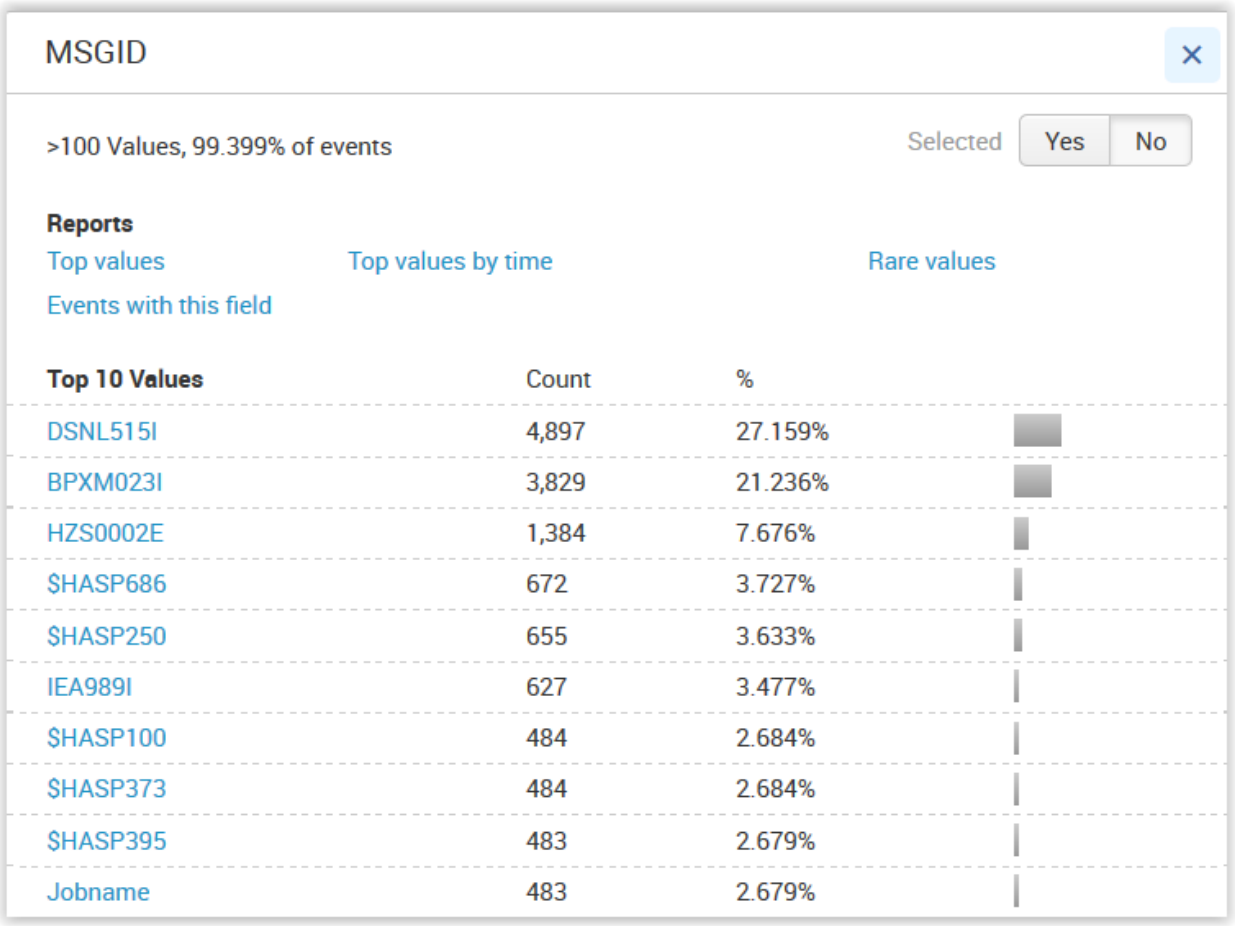
The left side of the search screen will show all of the fields associated with the records, along with a count of the unique values found within each field from the records returned from the search



The MSGID field is actually not defined by the Spunk ingestion kit. Splunk allows you to add additional fields based on the record contents; in this example MSGID was added by

highlighting the section of the record with the message ID; Splunk then uses that location to derive the MSGID for each record.

Clicking on a field is one quick way to find the most frequent values that occur for that field. For example, clicking on the MSGID field shows the most frequent message IDs in the data:



Much more can be done with searching (such additional search conditions using field values and times intervals), build dashboards, etc. that are beyond the scope of this paper but documented in the Splunk documentation. The key is that once the z/OS SYSLOG data is available to Splunk, it can be treated just like any other data that comes into Splunk.

Summary

While this example implementation is simple, it can be easily built upon to send additional data sources to Splunk. The data rates should be monitored, as it can be desirable to set up several Splunk Heavy Forwarders to receive different sets of data from CDPz, for performance purposes.

It should also be clear that once the Data Receiver writes the data onto the local file system, any process that can read the files can be used for any type of processing that is wanted. This provides a lot of flexibility, but also means these files should be protected so that only the desired processes and functions can read them.

For further information on the Common Data Provider for z Systems, the following information sources are available:

- IBM Common Data Provider 1.1 Knowledge Center

  https://www.ibm.com/support/knowledgecenter/SSGE3R_1.1.0/com.ibm.cdpz.doc/welcome.html

  Contains links to program directories (SMP/E installation) and Users Guide (product setup, customization, and integrations).

- IBM Common Data Provider on IBM DeveloperWorks

  http://ibm.biz/CDPzWIki

  Contains FAQs, documentations, and product service information.