Hortonworks SmartSense

User Guide

(April 3, 2017)

Hortonworks SmartSense: User Guide

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source. Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

Table of Contents

List of Tables

1. Document Navigation

Hortonworks SmartSense gives all support subscription customers access to a unique service that analyzes HDP cluster diagnostic data, identifies potential issues, and recommends specific solutions and actions. These analytics proactively identify unseen issues and notify customers of potential problems before they occur.

The Hortonworks SmartSense Tool (HST) provides cluster diagnostic data collection capabilities, enabling customers to quickly gather configuration, metrics, and logs that they can use to analyze and troubleshoot SmartSense support cases.

Hortonworks SmartSense User Guide provides you with the latest information about using SmartSense. For SmartSense installation and upgrade instructions, see the Hortonworks SmartSense Installation. After installing SmartSense, refer to the *Hortonworks SmartSense User Guide* for information about using SmartSense in an Ambari or non-Ambari environment and performing additional configuration.

If you have already installed SmartSense and you are ready to use it, choose your scenario:

- Using SmartSense with Ambari [2]
- Using SmartSense in a Non-Ambari Environment (Deprecated) [15]

2. Using SmartSense with Ambari

SmartSense is automatically included in Ambari 2.2.x and later. The integration between Ambari and SmartSense is facilitated by the Ambari stack and views extension mechanisms. These extensions enable you to add SmartSense as a native Ambari service, and they automatically deploy an Ambari view, enabling you to quickly capture data using the Ambari web UI.

2.1. Roles Required for Using SmartSense

The following table describes bundle capture-related actions and roles required to perform them:

Action	Ambari Administrator	Other Users*
Access Ambari View	✓	✓
Initiate SmartSense capture	✓	
Initiate support capture	✓	
View "Bundles" page	✓	✓
View bundle	✓	✓
Upload a bundle	✓	✓
Download encrypted bundles	✓	✓
Download unencrypted bundles	✓	✓
Delete bundles	✓	✓
View capture schedule	✓	✓
Update capture schedule	✓	
Pause capture schedule	✓	
Activate capture schedule	✓	
Delete capture schedule	✓	
View recommendations	✓	✓

Action	Ambari Administrator	Other Users*
Apply recommendations	✓	
Revert recommendations	✓	

"Other Users" include Cluster Administrator, Cluster User, Service Operator, Service Administrator, and Cluster Operator as defined in Understanding Cluster Roles.

2.2. Capturing Bundles

After you install the SmartSense service and view, data collection can begin.

To trigger an ad hoc capture, access the **SmartSense View** by clicking the ▦ icon and selecting **SmartSense View**, and then follow steps depending on your use case:

- If you would like to prevent issues, improve security, and/or increase availability and performance of your cluster: Capturing for Proactive Analysis
- If you are working with support to troubleshoot a support case: Capturing for Troubleshooting

2.2.1. Capturing for Proactive Analysis

To capture bundles for analysis, follow these steps:

1. Under **Select the intent for data capture**, select **Proactive Analysis**.
2. Click the **Capture** button.

   SmartSense will analyze cluster configuration and metrics for all cluster nodes, and will produce recommendations to prevent issues, improve security, availability and performance of your cluster.

Also see Viewing, Downloading, and Uploading Bundles.

2.2.2. Capturing for Troubleshooting

1. Under **Select the intent for data capture**, select **Support Case Troubleshooting**.
2. Enter your **Case Number**.
3. Select the type of diagnosis:
   - **Cluster Service**:

     a) Select services for diagnosis.

     b) Next, select hosts for diagnosis: **All Hosts** or choose **Only Specific Hosts** and select specific hosts.
   - **YARN Application**:

Enter **Application ID**. The YARN application details, application master logs and a subset of container logs will be captured.

- **Hive Query**:

  Enter one of the following: **Tez DAG ID, YARN App ID**, or **MR Job ID**. The SQL query, execution plan, application logs will be captured.

4. Click the **Capture** button.

   This triggers Ambari agents on each node to invoke the HST agent to capture specific data.

   After HST agents complete their captures and report data to the HST server, the completed bundle is available in the bundles list for download, or it is automatically uploaded to the SmartSense Gateway, if configured.

Also see Viewing, Downloading, and Uploading Bundles.

2.3. Automatically Capturing and Uploading Bundles via SmartSense Gateway

When enabled, the gateway automatically uploads completed bundles to Hortonworks when a capture is completed. This includes SmartSense analysis as well as support case troubleshooting bundles. You can also schedule SmartSense Analysis bundles for capture and automatic upload in the SmartSense Ambari view.

2.3.1. Creating a New Capture Schedule

If you have deleted the default capture schedule, you can create a new one:

1. Access **SmartSense View** by clicking and selecting **SmartSense View**.
2. Click the **Schedule** link in the top right corner to access the scheduler settings.
3. Select the scheduling period (weekly or monthly) and the day of the week and time of day that you want the capture to take place.
4. Click **Set Capture Schedule**.

Note

Scheduler changes take up to one hour to take effect.

2.3.2. Updating the Capture Schedule

The SmartSense view provides a way to easily create, update, pause, resume, and remove the schedules used for automated bundle capture and upload. When you deploy it, SmartSense creates a default capture schedule. To view this default capture schedule and update it, follow these steps:

1. Access **SmartSense View** by clicking and selecting **SmartSense View**.
2. Click the **Schedule** link in the top right corner to access the scheduler settings.
3. Remove, pause, or resume existing schedules.

   You can also update the capture schedule by selecting a new scheduling period (weekly or monthly) or changing the day of the week and time of day that you want the capture to take place.

Note

Scheduler changes take up to one hour to take effect.

2.4. Viewing, Downloading, and Uploading Bundles

Completed bundles can be manually downloaded and uploaded.

You can also automate and schedule this process by using the SmartSense Gateway. When using the SmartSense Gateway, all bundles are uploaded to Hortonworks. When support case troubleshooting bundles are received, they trigger a case notification. This case notification uses the case number provided during the capture initiation process.

To view and download bundles, follow these steps:

1. Access the **SmartSense View** by clicking and selecting **SmartSense View**.
2. Click the **Bundles** link in the top right corner.

   This page shows all bundles that have been captured and their status. If data is still being captured, the UI automatically updates itself with the capture progress until completed.
3. After the bundle is in a completed state, you can:
   - Download it manually by clicking **Download** and selecting either **Download Encrypted** or **Download Unencrypted**.
   - Upload it manually by clicking **Upload**.

   Alternatively, if a SmartSense Gateway is configured, the bundle is automatically uploaded to Hortonworks.

2.5. Viewing SmartSense Recommendations

From SmartSense View in Ambari Web UI, you can access your SmartSense recommendations.

Prerequisites

In order for recommendations to be generated for your cluster, you first need to capture a bundle and then upload it through HTTPS gateway for analysis.

Alternatively, you can schedule automatic capture and upload using the SmartSense Gateway.

Steps

1. Access the **SmartSense View** by clicking ▦ and selecting **SmartSense View**.
2. Click the **Recommendations** link in the top right corner.
3. From the recommendations page, you can view open recommendations, and view previously deferred and ignored recommendations.
4. To make sure that the recommendations are up-to-date, from the ••• menu select **Get Latest**.

Additional options to Show History, and Export as Excel are available from the ••• menu in the top right corner.

2.5.1. Reviewing Open Recommendations

The following information is available for each recommendation:

Table 2.1. Recommendation Summary

Column	Description
Priority	One of: Critical High Medium Low
Service	The HDP service to which the recommendation applies.
Recommendation	The summary of the recommendation.
Category	Broad category (such as "Operations", "Performance", or "Security") to which the recommendation belongs.
Avg. Rating	Average customer rating for the recommendation.
Classifiers	These markers indicate actions available for any recommendation: ⚙ means that the recommendation can be applied automatically through Ambari. 🔧 means that the configuration is not managed by Ambari and you must apply the recommendation manually.

Column	Description
	☰ means that in order to apply the recommendation you must also apply dependent recommendations. ↺ means that the recommendation has previously been applied and then reverted.

Click on a column name to sort the recommendations accordingly.

To review and apply, ignore, or defer a recommendation, click on its corresponding row.

2.5.2. Reviewing a Recommendation

To review a recommendation, click on its corresponding row.

The following information is available for each recommendation:

Table 2.2. Recommendation Details

Column	Description
Priority	One of: Critical High Medium Low
Status	One of: • Open • Applied • Ignored • Deferred • Reverted • Reopened
Recommendation Category	Broad category (such as "Operations", "Performance", or "Security") to which the recommendation belongs.
Affects	Describes to which specific software component the recommendation is related.
Rule Id	Unique ID that identifies the SmartSense rule related to the SmartSense recommendation.
Description	Background and context related to the recommendation.
Findings	Description of how your cluster deviates from the recommended configuration.
Recommendation	An outline of specific changes that need to be made to your cluster to apply the recommendation.
Configurations	Lists affected configuration properties, including: • **Config File** - The specific file that needs to be changed

Column	Description
	• **Property Name** - The specific property that needs to be changed • **Captured Value** - Configured value at the time of bundle capture • **Current Value** - Current configured value in Ambari • **Recommended Value** - Recommended value for this cluster
Affected hosts	Hosts on which the configuration change is required.

In addition, the following actions are available for each recommendation:

- **Ignore** - Let us know that you do not want to apply this recommendation and help us understand why.
- **Defer** - Let us know that you will be applying this at a later date, but not right now.
- **Mark As Applied** (for recommendations that have to be applied manually), or **Proceed to Apply** (for recommendations that can be applied automatically)

If you ignore or defer a recommendation, you can still apply it later.

2.5.3. Applying a Recommendation

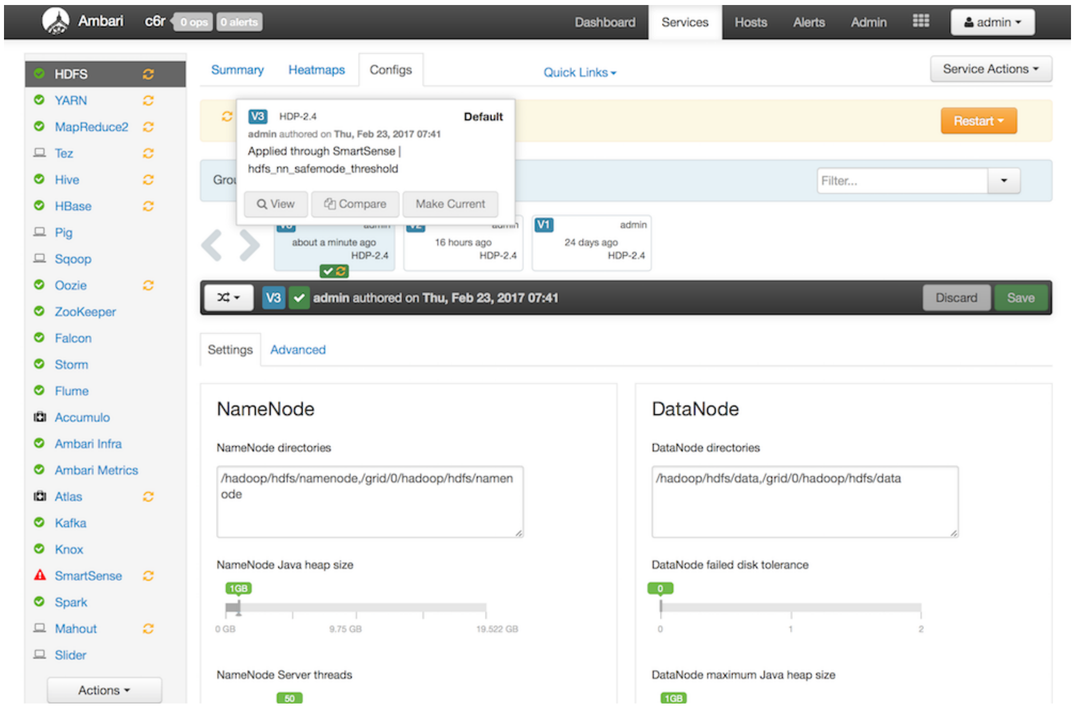
While some recommendations can be applied automatically, others have to be applied manually.

There are two ways to tell how a recommendation can be applied:

- When reviewing open recommendations, you can see in the **Classifiers** column, what options are available for which recommendation.
- When reviewing a specific recommendation, you can see one of the two options: **Mark As Applied** (for recommendations that must be applied manually) or **Proceed to Apply** (for recommendations that must be applied automatically).

Applying a Recommendation Automatically

1. From the **Recommendations** page, click on the table row corresponding to the recommendation that you want to review.
2. Click on **Proceed to Apply**.
3. Review recommended changes.
4. Enter a comment in the **Change Notes** field. This comment will later allow you to track the Ambari configuration version created after applying a configuration.
5. Click on **Apply**.
6. You can optionally provide feedback for this recommendation and then click on **Submit Feedback**. Or you can opt out and click **I will provide later**. You can still provide feedback later, from the **History** page.
7. You can view the configuration change in Ambari configuration history.



V3 HDP-2.4 Default
admin authored on Thu, Feb 23, 2017 07:41
Applied through SmartSense | hdfs_nn_safemode_threshold

V3 about a minute ago HDP-2.4 · V2 16 hours ago HDP-2.4 · V1 admin 24 days ago HDP-2.4

V3 admin authored on Thu, Feb 23, 2017 07:41

NameNode

NameNode directories: /hadoop/hdfs/namenode,/grid/0/hadoop/hdfs/namenode

NameNode Java heap size: 1GB (0 GB – 9.75 GB – 19.522 GB)

NameNode Server threads: 50

DataNode

DataNode directories: /hadoop/hdfs/data,/grid/0/hadoop/hdfs/data

DataNode failed disk tolerance: 0 (0 – 1 – 2)

DataNode maximum Java heap size: 1GB

Applying a Recommendation Manually

1. From the **Recommendations** page, click on the table row corresponding to the recommendation that you want to review.
2. Apply the recommendation manually.
3. Click on **Mark As Applied**.
4. Click on **I have** to confirm that you've applied the changes.

You can revert previously applied recommendations. This option is available on the **History** page.

2.5.4. Reviewing and Reverting Previously Applied Recommendations

To view the history of previously applied recommendations:

1. Click on the menu and select **Show History**.

   The **History** tabs allows you to review previously applied, ignored, deferred, and reverted recommendations, and, if needed, revert applied recommendations and review and reopen deferred and ignored recommendations.

2. To get more details about a specific recommendation, click on the .
3. You have an option to **Review and Revert**. If a recommendation has previously been reverted, this option is grayed out.

2.5.5. Exporting Recommendations as Excel Spreadsheet

You can export SmartSense recommendations to an Excel spreadsheet (XLSX file format).

To do that, click on the ••• menu and select **Export as Excel**. The spreadsheet will be downloaded to your default download location.

2.6. Configuring Anonymization Rules

As data is captured, specific types of data are automatically anonymized. By default, IP addresses and the domain component of host names are anonymized. To customize these anonymization rules, follow these steps:

1. Navigate to the Ambari **Dashboard** and click the **SmartSense** service.
2. Click the **Config** tab.
3. Navigate to the **Data Capture** section.
4. Add the new anonymization rule (or change the existing rule) by following the details provided in Configure Data Anonymization Rules.

2.7. Accessing the Activity Explorer

The Activity Explorer includes an embedded instance of Apache Zeppelin, which hosts prebuilt notebooks that visualize cluster utilization data related to user, queue, job duration, and job resource consumption. To access the Activity Explorer:

Note

The quick link to the Activity Explorer is available only in Ambari 2.4 and later. If you are using **Ambari version earlier than 2.4**, you must access the Activity Explorer using the following URL: *http://<activity_explorer_host>:9060/*.

1. Navigate to the Ambari **Dashboard** and click the **SmartSense** service.
2. In the **Summary** tab, click **Quick Links** > **Activity Explorer**.

   This launches the Activity Explorer in a new browser tab.
3. Log in with your Activity Explorer admin credentials.
4. From the **Notebook** dropdown in the top toolbar, select the name of the notebook that you want to view.

   The following preconfigured notebooks are available:

   - Chargeback Dashboard [11]
   - HDFS Dashboard [12]

- MapReduce & Tez Dashboard [13]
- YARN Dashboard [14]

Zeppelin organizes data in notebooks, where each notebook contains rows of paragraphs. Each paragraph visualizes the results of a single SQL statement using either a table, bar chart, pie chart, area chart, line chart, or scatter plot.

Once you opened a notebook, be aware of these three operations:

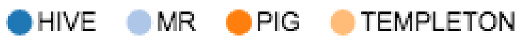
1. Since the notebooks represent a view of SmartSense utilization data at a specific point in time, they need to be refreshed. In order to refresh all of the data shown in all paragraph of a notebook, you need to:

   a. Hover over the row containing the notebook title, and a set of controls will appear.

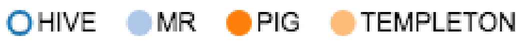
   b. Click on the ▷ button to “Run all paragraphs”. The data for each paragraph in the notebook will be refreshed.

2. Top N paragraphs show the top 10 entries by default, but you can change this number by entering a new number in the **Top** input field and then typing enter.

3. Charts have interactive filters that let you select and deselect specific resources by clicking on the circle in the chart legend. For example, if there are four resources being displayed in a chart, and you only want to see four, you can click on a colored circle in the legend to filter it out:

   
   ● HIVE ● MR ● PIG ● TEMPLETON

   Once clicked, the inside of the circle will change to white, and the entry will not be displayed in the chart. For example, if you deselect "Hive", the legend will look like this:

   
   ○ HIVE ● MR ● PIG ● TEMPLETON

2.7.1. Chargeback Dashboard

The Chargeback Dashboard helps operators understand which resources are being consumed and what costs are associated with these resources. This dashboard exposes five types of resources:

- **CPU Hours** (in hours) - The amount of CPU used by MapReduce and Tez jobs
- **Memory Hours** (in gigabytes) - The amount of memory consumed by MapReduce and Tez jobs, and length of consumption
- **Storage** (in gigabytes) - The amount of HDFS space being consumed
- **Data IO** (in gigabytes) - The amount of data read and written to HDFS
- **Network IO** (in gigabytes) - The amount of data sent and received over the cluster’s network

Paragraph	Description
Chargeback Report	This paragraph lets you associate a financial cost with each of the five resources presented in the previous paragraph. Based on these per unit financial costs, you can see how much should be charged for each resource type. The report also sums up the charge per resource to a per user total, so it's easy to see how much should be charged back to that specific user for their total resource consumption. The goal is to show how much money each user should be charged for the cluster resources that they have consumed.

2.7.2. HDFS Dashboard

The HDFS Dashboard helps operators better understand how HDFS is being used and which users and jobs are consuming the most resources within the file system.

This dashboard includes the following paragraphs:

- File Size Distribution
- Top N Users with Small Files
- Top N Largest HDFS Users
- Average File Size
- HDFS File Size Distribution Trend
- HDFS Utilization Trend
- HDFS File Size Distribution Trend by User
- HDFS File Size Distribution Trend by User
- Jobs With High Number of HDFS Operations
- HDP 2.5: Jobs Creating Many HDFS Files
- Jobs With Large Amount of Data Written

Most of these paragraphs have titles that are self-explanatory. A few of them are described below to provide more context:

Paragraph	Description
File Size Distribution	For any large multi-tenant cluster, it's important to identify and keep the proliferation of small files in check. The paragraph displays a pie chart showing the relative distribution of files by file size categorized by Tiny (0-10K), Mini (10K-1M), Medium (30M-128M), and Large (128M+) files. The goal is to show how dominant specific file size categories are within HDFS. If there are many small files, you can easily identify (in the next paragraph) who is contributing to those small files.
Top N Users with Small Files	Understanding how prevalent files of specific sizes are is helpful, but the next step is understanding who is responsible for creating those files. The goal of this paragraph is to show who is responsible for creating the majority of small files within HDFS.

Paragraph	Description
Top N Largest HDFS Users	This paragraph helps you understand where all of the HDFS capacity is being consumed, and who is consuming it. The goal is to help you quickly understand which user or users are storing the most data in HDFS.
HDFS File Size Distribution Trend by User	Each "by User" paragraph allows you to see how an individual user's file sizes are trending. This paragraph helps answer questions related to points in time where large or small files start becoming more or less prevalent for specific users, and can help measure the success of coaching users on Hadoop best practices.
HDP 2.5: Jobs Creating Many HDFS Files	When troubleshooting issues related to HDFS NameNode performance, it's helpful to understand which jobs are creating the most files, and potentially putting the largest amount of load on the NameNode. In HDP 2.5, new counters have been added to track how many files are created by each YARN application. This is helpful in troubleshooting erroneous jobs that are unintentionally creating hundreds of thousands, or even millions of files within HDFS.

2.7.3. MapReduce & Tez Dashboard

The MapReduce & Tez Dashboard was created to provide key information for workloads that use MapReduce or Tez for execution.

This dashboard includes the following paragraphs:

- Top N Longest Running Jobs
- Top N Resource Intensive Jobs
- Top N Resource Wasting Jobs
- Job Distribution By Type
- Top N Data IO Users
- CPU Usage By Queue
- Job Submission Trend By Day.Hour

Most of these paragraphs have titles that are self-explanatory. A few of them are described below to provide more context:

Paragraph	Description
Top N Resource Wasting Jobs	Resource wasting is calculated by calculating the difference between the memory asked for and the memory that was actually used. For example, if a job asks for 100 8GB containers but only uses 5GB per container, 3GB per container is considered wasted. This is calculated per job, and the top 10 are listed.
Job Submission Trend By Day.Hour	This paragraph shows the number of jobs submitted by day and hour with the notation being <day>.<hour>. For example: • Monday.1 - 1am on Monday • Monday.20 - 8pm on Monday The goal of this dashboard is to identify specific job submission hotspots during the week and day. You can use this information to identify the best time to schedule resource intensive jobs to execute.

2.7.4. YARN Dashboard

The YARN Dashboard provides key information for queue, application, container, and NodeManager host metrics.

This dashboard includes the following paragraphs:

- Application Runtime Duration by Queue
- Top N Applications by Number of Containers Requested
- Top N Applications by Number of Containers Failed
- Top N Hosts by Number of Containers Executed
- Top N Hosts by Number of Application Failures
- Top N Hosts by Localization Time
- Top N Hosts by Container Launch Delay

Most of these paragraphs have titles that are self-explanatory. One of them is described below to provide more context:

Paragraph	Description
Top N Applications by Number of Containers Failed	This paragraph shows the top jobs with the highest number of failed containers and the reason for each failure, so that you can quickly identify which containers failed and why.

3. Using SmartSense in a Non-Ambari Environment (Deprecated)

Deploying Hortonworks SmartSense Tool (HST) on a cluster that is not managed by Apache Ambari requires manual installation and configuration.

3.1. Capturing Bundles in a Non-Ambari Environment (Deprecated)

You have two options for data capture when HST is deployed outside of Ambari: using an HST agent CLI and using the Web UI.

3.1.1. HST Server Web UI Capture (Deprecated)

To use this option, you must enable the HST web UI for capture: Enable Capture Through UI. The web UI capture method enables users to capture data by simply clicking the desired services to capture, entering their case number, and clicking **Capture**.

To access the HST server web UI, navigate to http(s)://`HST Server FQDN`:9000/. The default user name and password is:

- **Default Username**: admin
- **Default Password**: admin

3.1.2. HST Agent CLI Capture (Deprecated)

HST agents collect data for the specific node on which they are installed. To capture data for all nodes in the cluster, which is the most common use case, you must run the `hst capture` command on all nodes. Typically this is done using pdsh or other parallel distributed shell utilities. **Running the `hst capture` command on all nodes in parallel is highly recommended**, because it allows bundles to be captured in the least amount of time. For the all agents to consolidate data in the same bundle, it is important that all agents initiate capture within three minutes after the first agent initiates.

To initiate capture of service data for a specific case number, use the following syntax:

```
# hst capture {service} {case number} {optional: level}
```

The HST agent can collect data for multiple services simultaneously. To obtain the list of supported services, run the following command:

```
# hst list-services

Supported services:
  AMS           : Collect data for Ambari metrics issue
  Ambari        : Collect data for Ambari issue
  Falcon        : Collect data for Falcon issue
  Ganglia       : Collect data for Ganglia issue
  HBase         : Collect data for HBase issue
  HCatalog      : Collect data for HCatalog issue
  HDFS          : Collect data for HDFS issue
  Hive          : Collect data for Hive issue
  Kafka         : Collect data for Kafka issue
  Knox          : Collect data for Knox issue
  MR            : Collect data for MapReduce issue
  Nagios        : Collect data for Nagios issue
  Oozie         : Collect data for Oozie issue
  Pig           : Collect data for Pig issue
  Ranger        : Collect data for Ranger issue
  Spark         : Collect data for Spark issue
  Sqoop         : Collect data for Sqoop issue
  Storm         : Collect data for Storm issue
  Tez           : Collect data for Tez issue
  YARN          : Collect data for YARN issue
  ZK            : Collect data for ZooKeeper issue
```

You can specify services individually, combine services using commas as delimiters, or specify all services by using the all keyword.

Support Case Troubleshooting Capture Example

For example, to capture data just for Hadoop Distributed File System (HDFS) and for case number 0001, run **hst** as follows:

```
# hst capture HDFS 0001
```

To capture data for HDFS, Apaceh Hive, and Apache Oozie for case number 0002, run hst as follows:

```
# hst capture HDFS,HIVE,OOZIE 0002
```

To capture L3 capture-level data for every service listed for case number 0003, run hst as follows:

```
# hst capture all 0003 L3
```

SmartSense Analysis Capture Example

To capture data for SmartSense Analysis, only configuration and metrics are required and 0 is used as the case number:

```
# hst capture all 0
```

3.2. Viewing and Downloading Bundles in a Non-Ambari Environment (Deprecated)

After a bundle has been initiated, you can use the HST Server web UI to check bundle status and then download the bundle when it is complete.

3.2.1. HST Server Login (Deprecated)

To access the HST Server web UI, navigate to

```
http(s)://HST Server FQDN:9000/
```

The default user name and password are both “admin”. :

3.2.2. Viewing and Downloading Bundles (Deprecated)

After a bundle is captured, you can either download it or, if using the SmartSense Gateway, have it automatically uploaded. For more information about the gateway, see the Installing SmartSense Gateway.

If a gateway is not configured, you must manually upload the bundle to Hortonworks by using SFTP. The connectivity details for the SmartSense SFTP environment are available in this article: https://hortonworks.secure.force.com/articles/en_US/How_To/Uploading-SmartSense-Bundles (To view this article, you need a valid Hortonworks support account).

3.3. Configuring Anonymization Rules in a Non-Ambari Environment (Deprecated)

1. Use SSH to access the HST server host.
2. Edit the `/etc/hst/conf/anonymization_rules.json` file to add or change existing anonymization rules by following details provided in Configure Data Anonymization Rules.

4. Uploading Support Bundles

You can use SmartSense Gateway to automatically upload bundles, or you can upload bundles manually.

For more information about uploading support bundles, see Bundle Transport.

5. Configuring SmartSense

This chapter guides you through common configuration tasks such as changing capture levels, configuring data anonymization rules, and changing server and agent configurations.

5.1. Configuring Data Anonymization Rules

Anonymization rules define regular expressions to anonymize sensitive data (like IP addresses, and so on). Each rule uses JSON format to define what to match and the value to replace.

Rule Types

You can define the following types of anonymization rules:

- Pattern-based - Anonymize data by pattern, using the ***extract*** field to match and extract content to anonymize.
- Property-based - Anonymize structured content. The supported formats are: XML, property, ini, and YAML files.
- XPath-based - Anonymize XML data using XPATH.
- JSONPath-based - Anonymize JSON data using JSONPATH.

In addition, there are **domain-based rules** that can be used to anonymize domain names. They are a special case of pattern rules where the anonymization pattern is build from local host FQDN. The domain-based rules cannot be customized.

For a detailed description of all the fields required to define annonymization rules, refer to Fields for Defining Anonymization Rules.

Note

Anonymization rule formats vary between different SmartSense versions. Make sure that you consult the documentation that matches your SmartSense version.

5.1.1. Fields for Defining Anonymization Rules

To define anonymization rules, use the following fields:

Table 5.1. Fields for Defining Anonymization Rules

Field	Description
name	Provides a descriptive name for data anonymized by the rule. It has to be unique across all rules.
rule_id	Defines the class of rules the current rule belongs to. The supported rule IDs are: *PATTERN*, *PROPERTY*, *XPATH*, JSONPATH. This parameter is case-insensitive.
patterns	Defines a list of data patterns to be anonymized. It is applicable only to *Pattern* rules, where rule_id=PATTERN.

Field	Description
	These patterns are matched in a case-insensitive manner, which means that the following pattern `keystore.pass=([^\\s]*)` matches with any of the following values: • keystore.pass=123 • KeyStore.Pass=123 • KEYSTORE.PASS=123

extract	Specifies a pattern to extract data matched through the list of patterns. The extract pattern is matched in a case-insensitive manner.
	
	For example, in order to anonymize the **oozie.https.keystore.pass** password, the following pattern and extract values are used:
	
	`"patterns": ["oozie.https.keystore.pass=([^\\s]*)"]`
	
	`"extract": "=([^\\s]*)",`
	
	This pattern is matched with values such as `oozie.https.keystore.pass=1234`.
	
	The extract pattern is used to extract and anonymize only the values after the = (which in this example is `1234`). The `[^\\s]*` denotes all non-whitespace characters, and the capturing group `()` is used the exclude = from the anonymized value.
	
	If the extract pattern is not configured, the entire value matched with the pattern is anonymized (which in this example is `oozie.https.keystore.pass=1234`), regardless of capturing groups used in the patterns.

properties	Specifies a list of property name patterns to anonymize; these are case-insensitively matched. It is applicable only to *Property* rules.

parentNode	This field is applicable to property anonymization in XML files. It allows you to define the parent node of the property that you want to anonymize. By default, *parentNode* is set to `"parentNode": "property"`, because typically the XML block to anonymize has the parent node *property*, like in the following example:
	
	`<property>`
	`  <name>fs.s3a.proxy.password</name>`
	`  <value>Abc7j*4$aTh</value>`
	`  <description>Password for authenticating with proxy server.</description>`
	`</property>`
	
	For example, you can anonymize `main.ldapRealm.contextFactory.systemPassword` in the following XML block that has a parent node called *param* by setting `"parentNode": "param"` in the anonymization rule:
	
	`<param>`
	`  <name>main.ldapRealm.contextFactory.systemPassword</name>`
	`  <value>pass</value>`
	`</param>`
	
	The rule to anonymize the above content configures *param* as the root tag `"parentNode": "param"`:

Field	Description

	`{`
	`"name": "KNOX LDAP Password",`
	`"rule_id": "Property",`
	`"properties": ["main.ldapRealm.contextFactory.`
	`systemPassword"],`
	`"include_files": ["topologies/*.xml"],`
	`"action" : "REPLACE",`
	`"parentNode": "param",`
	`"replace_value": "Hidden"`
	`}`

action	The supported actions are: *ANONYMIZE*, *DELETE*, *REPLACE*. The *action* value is **not** case sensitive, so *Anonymize* or *delete* are also accepted values. ANONYMIZE action encrypts the data using the key indicated by *shared* flag, DELETE deletes the data, and REPLACE replaces the data with a predefined value, which can be customized using *replace_value*.
replace_value	This field is used by the *REPLACE* action to specify a replacement for the data to anonymize. The default value is *Hidden*.
shared	Indicates which key to use for anonymization (shared or private). This value is used when the anonymization action is set to *ANONYMIZE*. It is a boolean type property (true/false). If set to true - the Hortonworks support team can unmask data if needed for diagnostic purposes; for example, host names and IP addresses for resolving issues on specific hosts or communication between hosts. Note that unmasked data is not stored in Hortonworks repositories; it is discarded as soon as the analysis finishes. The default value is true. Rules configured with `shared = false` cannot be unmasked by Hortonworks (and in some cases might become a roadblock for support case analysis.)
include_files	Specifies a list of *glob* file patterns for which the rule applies. If not configured, the rule is applicable to all files.
exclude_files	Specifies a list of *glob* file patterns which are excluded from anonymization. If not configured, no file is excluded from the rule application.
enabled	A flag (true/false) which specifies if the rule is enabled to be executed. By default, it is set to *true*.

5.1.2. Pattern-Based Anonymization Rules

Write pattern-based rules to anonymize data by pattern, using the *extract* pattern to extract content to anonymize.

Required and Optional Fields

- name
- rule_id (should be set to PATTERN)
- patterns
- extract (optional)

- include_files (optional)
- exclude_files (optional)
- action (optional, default value is ANONYMIZE)
- replace_value (optional, applicable only when action=REPLACE)
- shared (optional, default value is *true*)
- enabled (optional, default value is *true*)

For more information on each field, refer to Fields for Defining Anonymization Rules.

Rule Definition Example (without *extract*)

```
    {
      "name": "EMAIL",
      "rule_id": "Pattern",
      "patterns": ["(?<![a-z0-9._%+-])[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,6}
(?![a-z0-9._%+-])$?"],
      "shared": false
    }
```

The content of the input file *version.txt* is:

```
Hadoop 2.7.3.2.5.0.0-1245
Subversion git@github.com:hortonworks/hadoop.git -r
 cb6e514b14fb60e9995e5ad9543315cd404b4e59
Compiled by jenkins on 2016-08-26T00:55Z
```

The content of the output file *version.txt*, with anonymized email address, is:

```
Hadoop 2.7.3.2.5.0.0-1245
Subversion ‡qpe@unqfay.mjp‡:hortonworks/hadoop.git -r
 cb6e514b14fb60e9995e5ad9543315cd404b4e59
Compiled by jenkins on 2016-08-26T00:55Z
```

Rule Definition Example (with *extract*)

```
    {
      "name": "KEYSTORE",
      "rule_id": "Pattern",
      "patterns": ["oozie.https.keystore.pass=([^\\s]*)",
 "OOZIE_HTTPS_KEYSTORE_PASS=([^\\s]*)"],
      "extract": "=([^\\s]*)",
      "include_files": ["java_process.txt", "pid.txt", "ambari-agent.log",
 "java_process.txt", "oozie-env.cmd"],
      "shared": false
    }
```

The content of the input file *oozie-env.cmd* is:

```
oozie.https.keystore.pass=abcde
set OOZIE_HTTPS_KEYSTORE_PASS=12345
```

To anonymize the content of the input file, the following anonymization patterns configured in the rule will be used:

```
"oozie.https.keystore.pass=([^\\s]*)", "OOZIE_HTTPS_KEYSTORE_PASS=([^\\s]*)"
```

`oozie.https.keystore.pass=([^\\s]*)` and `OOZIE_HTTPS_KEYSTORE_PASS=([^\\s]*)` match with `oozie.https.keystore.pass=abcde` and `OOZIE_HTTPS_KEYSTORE_PASS=12345` respectively.

Next, the extract pattern `"=([^\\s]*)` is used to identify *12345* and *abcde*, which are the values to be anonymized.

The content of the output file *oozie-env.cmd* is:

```
oozie.https.keystore.pass=‡vvdwa‡
set OOZIE_HTTPS_KEYSTORE_PASS=‡zdowg‡
```

The values of `oozie.https.keystore.pass` and `OOZIE_HTTPS_KEYSTORE_PASS` have been anonymized.

For more examples, refer to Examples of Pattern-Based Anonymization Rules.

5.1.2.1. Examples of Pattern-Based Anonymization Rules

This section includes examples of commonly used pattern-based anonymization rules.

Example 1: Mask by pattern across all log files, without *extract* pattern

To mask all email addresses in all log files, use the following rule definition:

```
{
  "name": "EMAIL",
  "rule_id": "Pattern",
  "patterns": ["(?<![a-z0-9._%+-])[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,6}(?!
[a-z0-9._%+-])"],
  "include_files": ["*.log*"],
  "shared": false
}
```

Example 2: Mask by pattern across all log files, with extract pattern

To mask encryption keys, logged in the following format *Key=12..* with a value consisting of 64 hexadecimal characters, use the following rule definition:

```
{
  "name": "ENC_KEYS",
  "rule_id": "Pattern",
  "patterns": ["Key=[a-f\\d]{64}\\s"],
  "extract": "=([a-f\\d]{64})",
  "include_files": ["*.log*"],
  "shared": false
}
```

Input data, *test.log* is:

```
encryption key=
1234567890adc1234567aaabc1234567890adc1234567aaabc12345678901234 for keystore
derby.system.home=null
```

Output data, *test.log*, with the encryption keys anonymized, is:

```
encryption key=
‡8697685738fnx1736987qigyx7611731027yds0096404hlsph91727138403654‡ for
 keystore
derby.system.home=null
```

Example 3: Mask by pattern across all files, except a few files

To mask email addresses in all files, except *hdfs-site.xml* and *.property* files, use the following rule definition:

```
{
  "name": "EMAIL",
  "rule_id": "Pattern",
  "patterns": ["(?<![a-z0-9._%+-])[a-z0-9._%+-]+@[a-z0-9.-]+\\.[a-z]{2,6}(?!
[a-z0-9._%+-])"],
  "exclude_files" : ["*.properties", "hdfs-site.xml"],
  "shared": false
}
```

Input data, *version.txt*, is:

```
Hadoop 2.7.3.2.5.0.0-1245
Subversion git@github.com :hortonworks/hadoop.git -r
 cb6e514b14fb60e9995e5ad9543315cd404b4e59
Compiled by jenkins on 2016-08-26T00:55Z
```

Output file *version.txt*, with an anonymized email address, is:

```
Hadoop 2.7.3.2.5.0.0-1245
Subversion ‡qpe@unqfay.mjp‡ :hortonworks/hadoop.git -r
 cb6e514b14fb60e9995e5ad9543315cd404b4e59
Compiled by jenkins on 2016-08-26T00:55Z
```

5.1.3. Property-Based Anonymization Rules

Property-based rules anonymize structured content. The supported formats are: XML, property, ini, and YAML files.

Required and Optional Fields

- name
- rule_id (should be set to PROPERTY)
- properties
- parentNode (optional, applicable only for XML, default value is "property")
- include_files
- exclude_files (optional)
- action (optional, default value is ANONYMIZE)
- replace_value (optional, applicable only when action=REPLACE)
- shared (optional, default value is *true*)

- enabled (optional, default value is *true*)

For more information on each field, refer to Fields for Defining Anonymization Rules.

Rule Definition Example

```
    {
      "name": "PASSWORDS",
      "rule_id": "Property",
      "properties": [".*password.*", ".*awsAccessKeyId.*"],
      "include_files": ["*.xml", "*.properties", "*.yaml", "*.ini"],
      "exclude_files" : ["capacity-scheduler.xml"],
      "action" : "REPLACE",
      "replace_value": "Hidden"
    }
```

The following examples show how the rule defined above anonymizes specific password-related properties in XML, property, ini, and YAML files.

- **XML file content:**

  ```
  <property>
    <name>fs.s3a.proxy.password</name>
    <value>Abc7j*4$aTh</value>
    <description>Password for authenticating with proxy server.</description>
  </property>
  ```

  The XML file content, with password value anonymized:

  ```
  <property>
    <name>fs.s3a.proxy.password</name>
    <value>Hidden</value>
    <description>Password for authenticating with proxy server.</description>
  </property>
  ```

- **Property file content:**

  ```
  javax.jdo.option.ConnectionPassword=pswd
  ```

  The property file content, with password value anonymized:

  ```
  javax.jdo.option.ConnectionPassword=Hidden
  ```

- **Ini file content:**

  ```
  connection_password=pswd
  ```

  The ini file content, with password value anonymized:

  ```
  connection_password=Hidden
  ```

- **YAML file content:**

  ```
  "metrics_collector:\n" +
                  "  truststore.path : \"/etc/security/clientKeys/all.jks\"\n"
   +
                  "  truststore.type : \"jks\"\n" +
                  "  truststore.password : \"bigdata\"\n"
  ```

  The YAML file content, with password value anonymized:

```
"metrics_collector:\n" +
                "  truststore.path : \"/etc/security/clientKeys/all.jks\"\n"
 +
                "  truststore.type : \"jks\"\n" +
                "  truststore.password : Hidden\n"
```

For more examples, refer to Examples of Property-Based Anonymization Rules.

5.1.3.1. Examples of Property-Based Anonymization Rules

This section includes examples of commonly used property-based anonymization rules.

Example 1: Mask one configuration parameter in multiple files

Rule definition example:

```
{
  "name": "JPA_PASSWORD",
  "rule_id": "Property",
  "properties": ["oozie.service.JPAService.jdbc.password"],
  "include_files": ["oozie-site.xml", "sqoop-site.xml"],
  "action" : "REPLACE",
  "replace_value": "Hidden"
}
```

This rule anonymizes the value of `oozie.service.JPAService.jdbc.password` in oozie-site.xml and sqoop-site.xml.

Input data, *sqoop-site.xml*:

```
  <configuration>

    <property>
      <name>oozie.service.JPAService.jdbc.px</name>
      <value>at@!_*rue</value>
    </property>
```

Output data, *sqoop-site.xml*, with anonymized `oozie.service.JPAService.jdbc.px` parameter value:

```
  <configuration>

    <property>
      <name>oozie.service.JPAService.jdbc.px</name>
      <value>Hidden</value>
    </property>
```

Example 2: Mask multiple configuration parameters in multiple files

Rule definition example:

```
{
  "name": "JDBC_JPA_PASSWORDS",
  "rule_id": "Property",
  "properties": ["oozie.service.JPAService.jdbc.password", "javax.jdo.option.
ConnectionPassword"],
  "include_files": ["oozie-site.xml", "sqoop-site.xml", "hive-site.xml"],
  "action" : "REPLACE",
  "replace_value": "Hidden"
}
```

Example 3: Mask a configuration that matches a pattern

Rule definition example:

```
{
  "name": "GLOBAL_JDBC_PASSWORDS",
  "rule_id": "Property",
  "properties": [".*password"],
  "include_files": ["*.xml"],
  "action" : "REPLACE",
  "replace_value": "Hidden"
}
```

Input data:

ssl-server.xml

```
  <configuration>
    <property>
      <name>ssl.server.keystore.keypassword</name>
      <value>big123!*</value>
    </property>
```

ssl-client.xml

```
  <configuration>
    <property>
      <name>ssl.client.keystore.password</name>
      <value>NBg7j*4$aTh</value>
    </property>
```

Output data:

Anonymized *ssl-server.xml*

```
  <configuration>
    <property>
      <name>ssl.server.keystore.keypassword</name>
      <value>Hidden</value>
    </property>
```

Anonymized *ssl-client.xml*

```
  <configuration>
    <property>
      <name>ssl.client.keystore.password</name>
      <value>Hidden</value>
    </property>
```

5.1.4. XPath-Based Anonymization Rules

XPath-based rules anonymize XML data using XPath.

Required and Optional Fields

- name
- rule_id (should be set to XPATH)

- paths
- include_files
- exclude_files (optional)
- action (optional, default value is ANONYMIZE)
- replace_value (optional, applicable only when action=REPLACE)
- shared (optional, default value is *true*)
- enabled (optional, default value is *true*)

For more information on each field, refer to Fields for Defining Anonymization Rules.

Rule Definition Example

```
  {
    "name": "XPATH_RULE",
    "rule_id": "XPATH",
    "paths": ["/data/record[1]/value"],
    "include_files": ["*test_config.xml"],
    "shared": true
  }
```

Sample Input XML Data

```
<data>
    <record>
        <name>password</name>
        <value>valueToAnonymize</value>
    </record>
    <record>
        <name>name</name>
        <value>value</value>
    </record>
</data>
```

Sample Output XML Data (After Anonymization)

```
<data>
    <record>
        <name>password</name>
        <value>¶smfz923swc¶</value>
    </record>
    <record>
        <name>name</name>
        <value>value</value>
    </record>
</data>
```

For more examples, refer to Examples of XPath-Based Anonymization Rules.

You can use this reference documentation for XPath.

5.1.4.1. Examples of XPath-Based Anonymization Rules

This section includes examples of commonly used XPath-based anonymization rules.

Example 1: Rule with nested XML structure

Rule definition example:

```
{
  "name": "NESTED_XPATH_RULE",
  "rule_id": "XPATH",
  "paths": ["/configs/properties/passwd"],
  "include_files": ["*config.xml"],
  "shared": true
}
```

Input data:

```
<?xml version="1.0" encoding="UTF-8" ?>
<configs>
    <properties>
        <user>abc</user>
        <passwd>1234</passwd>
    </properties>
</configs>
```

Output data (after anonymization):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configs>
    <properties>
     <user>abc</user>
        <passwd>¶9165¶</passwd>
    </properties>
</configs>
```

Example 2: Rule with XML array structure

Rule definition example:

```
{
  "name": "ARRAY_XPATH_RULE",
  "rule_id": "XPATH",
  "paths": ["/configs/properties[2]/passwd"],
  "include_files": ["*config.xml"],
  "shared": true
}
```

Input data:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configs>
    <properties>
        <database>mysql</database>
        <url>user@host:port</url>
    </properties>
    <properties>
        <user>abc</user>
        <passwd>1234</passwd>
    </properties>
</configs>
```

Output data (after anonymization):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configs>
    <properties>
        <database>mysql</database>
        <url>user@host:port</url>
    </properties>
    <properties>
        <user>abc</user>
        <passwd>¶9165¶</passwd>
    </properties>
</configs>
```

Example 3: Rule with XML map structure

Rule definition example:

```
{
  "name": "MAP_XPATH_RULE",
  "rule_id": "XPATH",
  "paths": ["/configs/properties/passwd"],
  "include_files": ["*config.xml"],
  "shared": true
}
```

Input data:

```
<?xml version="1.0" encoding="UTF-8" ?>
<configs>
    <db>mysql</db>
    <properties>
        <user_name>sa</user_name>
        <passwd>sa_pass</passwd>
    </properties>
    <pooli_size>32</pooli_size>
    <timeout>10</timeout>
</configs>
```

Output data (after anonymization):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><configs>
    <db>mysql</db>
    <properties>
        <user_name>sa</user_name>
        <passwd>¶vm_wtto¶</passwd>
    </properties>
    <pooli_size>32</pooli_size>
    <timeout>10</timeout>
</configs>
```

Example 4: Rule to mask all array elements

Rule definition example:

```
{
  "name": "ALL_FROM_ARRAY_XPATH_RULE",
  "rule_id": "XPATH",
  "paths": ["/configs/properties[*]/passwd"],
  "include_files": ["*config.xml"],
  "shared": true
}
```

Input data:

```
<?xml version="1.0" encoding="UTF-8" ?>
<configs>
    <properties>
        <user>abc1</user>
        <passwd>pass1</passwd>
    </properties>
    <properties>
        <user>abc2</user>
        <passwd>pass2</passwd>
    </properties>
</configs>
```

Output data (after anonymization):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configs>
    <properties>
        <user>abc1</user>
        <passwd>¶smfz7¶</passwd>
    </properties>
    <properties>
        <user>abc2</user>
        <passwd>¶smfz8¶</passwd>
    </properties>
</configs>
```

Example 5: Rule to mask some array elements which have *passwd*

Rule definition example:

```
{
  "name": "SOME_FROM_ARRAY_XPATH_RULE",
  "rule_id": "XPATH",
  "paths": ["/configs/properties[passwd]/passwd"],
  "include_files": ["*config.xml"],
  "shared": true
}
```

Input data:

```
<?xml version="1.0" encoding="UTF-8" ?>
<configs>
    <properties>
        <user>abc1</user>
        <passwd1>pass1</passwd1>
    </properties>
    <properties>
        <user>abc2</user>
        <passwd2>pass2</passwd2>
    </properties>
    <properties>
        <user>abc3</user>
        <passwd>pass3</passwd>
    </properties>
</configs>
```

Output data (after anonymization):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<configs>
    <properties>
        <user>abc1</user>
        <passwd1>pass1</passwd1>
    </properties>
    <properties>
        <user>abc2</user>
        <passwd2>pass2</passwd2>
    </properties>
    <properties>
        <user>abc3</user>
        <passwd>¶smfz9¶</passwd>
    </properties>
</configs>
```

5.1.5. JSONPath-Based Anonymization Rules

JSONPath-based rules anonymize JSON data using JSONPath.

Required and Optional Fields

- name
- rule_id (should be set to JSONPATH)
- paths
- include_files
- exclude_files (optional)
- action (optional, default value is ANONYMIZE)
- replace_value (optional, applicable only when action=REPLACE)
- shared (optional, default value is *true)*
- enabled (optional, default value is *true*)

For more information on each field, refer to Fields for Defining Anonymization Rules.

Rule Definition Example

```
  {
    "name": "JSONPATH_RULE",
    "rule_id": "JSONPATH",
    "paths": ["$.users[0].password"],
    "include_files": ["*test_config.json"],
    "shared": true
  }
```

Sample Input JSON Data

```
{
  "users": [
    {
      "name": "Logsearch Admin",
      "username": "admin",
      "password": "testdata"
    },
    {
      "name": "Admin",
      "username": "admin",
      "password": "test data"
    }
  ]
}
```

Sample Output JSON Data (After Anonymization)

```
{
  "users": [
    {
      "name": "Logsearch Admin",
      "username": "admin",
      "password": "¶smfvvcz9¶"
    },
    {
      "name": "Admin",
      "username": "admin",
      "password": "test data"
    }
  ]
}
```

For more examples, refer to Examples of JSONPath-Based Anonymization Rules.

You can use this reference documentation for JSONPath.

5.1.5.1. Examples of JSONPath-Based Anonymization Rules

This section includes examples of commonly used JSONPath-based anonymization rules.

Example 1: Rule with nested JSON elements

Rule definition example:

```
{
   "name": "NESTED_JSONPATH_RULE_1",
   "rule_id": "JSONPATH",
   "paths": ["$.configs.properties.passwd"],
   "include_files": ["*config.json"],
   "shared": true
 }
```

Input data:

```
{
    "configs": {
        "properties":
            {
                "user": "abc",
                "passwd": "12!@"
            }
    }
}
```

Output data (after anonymizarion):

```
{
  "configs": {
    "properties": {
      "user": "abc",
      "passwd": "¶91!@¶"
    }
  }
}
```

Example 2: Rule with indexed JSON array objects

Rule definition example:

```
  {
    "name": "ARRAY_JSONPATH_RULE",
    "rule_id": "JSONPATH",
    "paths": ["$.configs.properties[1].passwd"],
    "include_files": ["config.json"],
    "shared": true
  }
```

Input data:

```
{
    "configs": {
        "properties": [
            {
                "database": "mysql",
                "url": "user@host:port"
            },
            {
                "user": "abc",
                "passwd": "12!@"
            }
 ]
    }
}
```

Output data (after anonymization):

```
{
  "configs": {
    "properties": [
      {
        "database": "mysql",
        "url": "user@host:port"
      },
      {
        "user": "abc",
        "passwd": "¶91!@¶"
      }
    ]
  }
}
```

Example 3: Rule with JSON map

Rule definition example:

```
    {
      "name": "MAP_JSONPATH_RULE",
      "rule_id": "JSONPATH",
      "paths": ["$.properties.passwd"],
      "include_files": ["*config.json"],
      "shared": true
    }
```

Input data:

```
 {
   "db":"mysql",
   "properties":
   {
     "user_name":"sa",
     "passwd":"sa_pass"
   },
   "pooli_size":32,
   "timeout":10
 }
```

Output data (after anonymization):

```
{
  "db": "mysql",
  "properties": {
    "user_name": "sa",
    "passwd": "¶vm_wtto¶"
  },
  "pooli_size": 32,
  "timeout": 10
}
```

Example 4: Rule to mask all JSON objects from list

Rule definition example:

```
{
  "name": "ALL_FROM_ARRAY_JSONPATH_RULE",
  "rule_id": "JSONPATH",
  "paths": ["$.configs.properties[*].passwd"],
  "include_files": ["*config.json"],
  "shared": true
}
```

Input data:

```
{
   "configs": {
       "properties": [
           {
               "user": "abc1",
               "passwd": "pass1"
           },
           {
               "user": "abc2",
               "passwd": "pass2"
           }
   ]
   }
}
```

Output data (after anonymization):

```
{
 "configs": {
   "properties": [
     {
       "user": "abc1",
       "passwd": "¶smfz7¶"
     },
     {
       "user": "abc2",
       "passwd": "¶smfz8¶"
     }
   ]
 }
}
```

5.2. Change Server and Agent Configurations in a Non-Ambari Environment (Deprecated)

All HST configurations are stored in `/etc/hst/conf`. Both `hst-server.ini` and `hst-agent.ini` have server and agent configurations. Changes performed on the HST server host are automatically propagated to all of the agents. Note that any change to the `hst-server.ini` file requires that you restart the HST server.

5.3. Configuring Bundle Upload

SmartSense Gateway is automatically configured with HTTPS so you don't normally need to perform this configuration. However, if a specific custom configuration is required by your corporate network firewall policies, you can use these instructions to configure SmartSense Gateway to upload bundles by using either SFTP or HTTPS:

5.3.1. Configuring the Gateway to Use SFTP (Deprecated)

You can configure the gateway to use SFTP to upload bundles to Hortonworks support using the connectivity and configuration details available in this article: https://support.hortonworks.com/s/article/SmartSense-Gateway-setup (To view this article, you need a valid Hortonworks support account).

Note

Using an SFTP-based gateway is deprecated, effective end of April 2018. If you are using SFTP-based gateway you should Upgrade to HTTPS-based gateway.

5.3.2. Configuring the Gateway to Use HTTPS

You can configure the gateway to use HTTPS to upload bundles to Hortonworks by using the connectivity and configuration details available in this article: https://support.hortonworks.com/s/article/SmartSense-Gateway-setup (To view this article, you need a valid Hortonworks support account).

To use an authenticated proxy to upload bundles to Hortonworks, follow these steps:

1. On the SmartSense Gateway host, edit the `/etc/hst/conf/gateway/hst-gateway.ini` file and supply the appropriate values for your environment:

```
; All proxy configurations are applicable only for HTTPS provider type
;#set to true#to#set up#a#proxy#between#gateway#and#SmartSense#environment
;default:false
provider.https.proxy.enabled=true
;#fully#qualified#proxy#hostname
provider.https.proxy.hostname=your.proxy.host
;#proxy#port#that#will#be#used#by#gateway#for#outbound#access
provider.https.proxy.port=3128
;#supported proxy#types#:#HTTP#/#HTTPS#[default:HTTP]
provider.https.proxy.type=HTTP
; supported proxy authentication #types#:#NONE#/#BASIC#/#DIGEST#[default:NONE]
provider.https.proxy.auth.type=BASIC
;#proxy#username#for#identified#auth.type
provider.https.proxy.auth.username=proxyuser
;#proxy#password#for#identified#auth.type
provider.https.proxy.auth.password=proxypassword
;#[optional]#any#additional#proxy#setup#parameters
; use#"|" to#separate#multiple#parameters
;#for example:#digest#requires#setting#parameters#such as
;#realm=default|nonce=12GHtqeZA!7Ke43
provider.https.proxy.auth.parameters=
```

2. After you update the configuration file, restart the SmartSense Gateway:

```
hst gateway restart
```

5.4. SmartSense Performance Tuning

To achieve optimal performance for your cluster size, you may need to increase the JVM memory settings.

The default setting, 2048 MB, is appropriate for a cluster with up to 100 nodes. For each additional 100 nodes, increase this setting by 0.5 GB to improve performance.

To adjust the setting, in the Ambari Web UI, navigate to the SmartSense service's **Config** section > **Advanced** > **Advanced hst-server-conf** where you will find the **Server max heap size** configuration property.