precisely

# Spectrum Spatial for Big Data

Version 5.1

## Release Notes

Contents:

# What's New?

The Spectrum Spatial for Big Data version 5.1 contains the following features. For more information on SDK, see **Spectrum Spatial for Big Data User Guide**.

## Spectrum Geocoding for Big Data

### *Spark Version Support*

With this release we have added support for **Spark 3.5.0+**.

### *Library Upgrades*

- Spectrum Geo-Addressing SDK (GA-SDK) 5.1.682
- Global Routing SDK (GRA) 3.0.1
- Location Intelligence SDK (LI SDK) 1.14.2

### *SERP Validation Code support in Addressing Geocode API*

The CA9 SERP support for Geo Addressing SDK is now available. SERP return codes indicate the quality of the input address as determined by the Canada Post's Software Evaluation and Recognition Program regulations. From Geo Addressing SDK different SERP validation codes will be returned as a part of Custom Fields with the key `SERP_VALIDATION_CODE`. The customPreference parameter SERP_MODE is used to achieve this capability, default value for `SERP_MODE` is false.

SERP Validation Codes can be:

- **V**: Validated the address or the input provided was valid.
- **C**: The input was correctable. “Correctable" address is one that can be corrected to match on.
- **N**: No Match or the input was non-correctable.

### *Bug Fixes*

Following bugs have been fixed for this release:

- When using an output field expression, a few fields on the result object were not accessible. These fields can now be accessed directly using output field expression. For example: `location.explanation.type`.

- Added a few custom fields which were missing in the response object. These custom fields are are now available in the addressing response. For example: `PRECISION_LEVEL`
- Fixed DPV (Delivery Point Validation) results coming from Operational Addressing SDK (OAS), for addresses to return accurate and consistent results . These DPV values are now consistent and available with the DPV datasets.
- While using PySpark users were not able to provide custom preferences using `PreferenceBuilder`. Custom Preferences support with `PreferenceBuilder` is now available with PySpark SDK.

## *PySpark Support for Operational Addressing SDK*

PySpark Support for Operational Addressing SDK is available in the 5.1.0.10 release, which includes Geo Addressing APIs in Python for Spark Integration (PySpark) for the following operations:

1. Geocode
2. Verify
3. Reverse Geocode
4. Lookup

For more information, see **Executing the PySpark Job** in the *Spectrum Geocoding for Big Data User Guide*. The Pydocs for PySpark Addressing APIs are available at: **Geo Addressing SDK APIs for PySpark documentation**.

> **Note:** This feature is only supported for Spark having Scala Version 2.12.

## *Snowflake Connector for Spark Geocoding Sample*

The Snowflake Connector for Spark Geocoding sample is now available on GitHub. This sample provides information about using Snowflake as an Apache Spark Data Source and the process of using Geo Addressing SDK APIs. This sample also demonstrates how to connect to Snowflake, create the data frame and execute the Geo Addressing SDK APIs in PySpark. Sample notebooks are provided for EMR and Databricks clusters on the GitHub Page:

**https://github.com/PreciselyData/big-data**.

## *New Operational Addressing SDK*

This release includes APIs for Spark integration for the following operations:

- **Reverse Geocode:** Accepts latitude/longitude coordinates for a point as input and returns address information that is the best match for that point.
- **Lookup:** Accepts unique key for an address and returns a geocoded matched candidate. Supported keys come from USA or AUS GNAF data (for example, P0000GL638OL for USA data and

GAACT715000223 for AUS) and are of types PB_KEY or GNAF_PID.

For more information, see the *Spectrum Geocoding for Big Data User Guide* on the **Spectrum Spatial for Big Data** documentation landing page, as well as the Spark Addressing API Scaladocs available in your distribution:

`/pb/addressing/software/spark2/sdk/scaladocs/bigdata`

## *Databricks Geocoding Sample*

Databricks Geocoding Sample demonstrates how to install, configure, and execute a geocoding process in Databricks. The samples have been updated with the version 3.0 of PDX (Precisely Data Experience). Samples have also been updated to use OAS (Operational Addressing SDK). The Geocoding Installation notebook files and instructions are provided on GitHub:

**https://github.com/PreciselyData/big-data**.

## Hadoop Distribution and Framework Support

**New**

- Added support for Cloudera 7.5

# Known Issues

- Geocoding SDK: When using the Geocoding CLI tool to manually configure your reference data, the bash command now requires an additional flag: `--t json`. This command generates the `JsonDataConfig.json` file, which contains configuration information for your geocoding datasets and needs to be regenerated for every software and data release.

```
cli configure --t json --s /precisely/geo_addr/data --d /precisely/geocoding/software/resources/config
```

  It is strongly recommended to use automatic data configuration, by providing the location of the data, to configure the geocoding engine. For more information, see the *Spectrum Geocoding for Big Data User Guide* on the **Spectrum Spatial for Big Data** documentation landing page.
- (HAD-4323) Some types of queries will cause Hive to evaluate UDFs in the HiveServer2 process space instead of on a data node. The Routing UDFs in particular use a significant amount of memory and can shut down the Hive server due to memory constraints. To process these queries, we recommend increasing the amount of memory available to the HiveServer2 process (for example, by setting HADOOP_HEAPSIZE in hive-env.sh).
- (HAD-2967) A null pointer exception is thrown while running the following constant geocoding queries with Hive on Spark or TEZ:

```
SELECT ReverseGeocode(-76.97921145819674,38.89325106109181,"epsg:4326");
```

```
SELECT Geocode("","One Second Street","","Jersey City","","07302","USA");
```

  This is due to an issue in Hive: **https://issues.apache.org/jira/browse/HIVE-16587**

  **Workaround**

```
CREATE table dual (country string);
      INSERT into dual values("usa");
      SELECT Geocode('','2920 LANGSTON PLACE SOUTHEAST APARTMENT 101',
      '','WASHINGTON','DC','20020',a.country) from dual a;
```

# System Requirements and Dependencies

This product is verified on the following Hadoop distributions:

- Cloudera 6.2.1 , 6.3, 7.1 and 7.5
- Hortonworks 3.1
- EMR 5.30, 6.0, 6.30

To use the product, you must be familiar with configuring Hadoop in Hortonworks, Cloudera, or EMR, and developing applications for distributed processing. For more information, refer to **Hortonworks**, **Cloudera**, or **EMR** documentation.

The following additional tools must be available to use certain product features:

*for Hive:*

- Hive version 1.2.1 or above
- Hive in Spectrum Geocoding for Big Data is not supported on Cloudera 5.16

*for Spark and Zeppelin Notebook:*

- Java JDK version 1.8 or above
- Hadoop version 2.6.0 or above
- Spark version 2.0 or above
- Zeppelin Notebook is not supported in Cloudera

1700 District Ave Ste 300
Burlington MA 01803-5231
USA

www.precisely.com