

Spectrum™ Routing for Big Data

Version 2.6.1

User Guide



Table of Contents

1 - Welcome

Introducing Spectrum™ Routing for Big Data	4
Spectrum™ Routing for Big Data Architecture	5
System Requirements and Dependencies	6

2 - Routing UDFs

Overview	9
Working with Routing Hive UDFs	10

3 - Security

Authentication	19
Hive - Kerberos Authentication	19
Hive - Sentry Authorization	20

1 - Welcome

In this section

Introducing Spectrum™ Routing for Big Data	4
Spectrum™ Routing for Big Data Architecture	5
System Requirements and Dependencies	6

Introducing Spectrum™ Routing for Big Data

Spectrum™ Routing for Big Data is a toolkit that can be used for processing enterprise data for large-scale spatial analysis. Billions of records from a single file can be processed in parallel using the Hive cluster processing framework, yielding results faster than ever. Unlike the traditional processing techniques that took weeks to process data, using this product, data processing can now be done in a just a few hours.

Spectrum™ Routing for Big Data is one of the products from Pitney Bowes used for working with Big Data. Other such products are:

- Spectrum™ Location Intelligence for Big Data
- Spectrum™ Geocoding for Big Data

Spectrum™ Routing for Big Data Architecture

What is Spectrum™ Routing for Big Data?

The Spectrum™ Routing for Big Data packages Routing components into an SDK for Big Data platforms like Hadoop for Hive.

SDK provides:

- Integration APIs for (PointToPointRoute, IsoChrono and IsoDistance)
- Input datasets and metadata

API Types:

- Pre-built Hive UDF wrappers for Routing operations
- Core Routing APIs (Security enabled via Kerberos and Apache Sentry for Hive)

System Requirements and Dependencies

Spectrum™ Routing for Big Data is collection of jar files that can be deployed to your Hadoop system.

Note: Installation is not required.

These jar files are compatible with any Hadoop Distribution having:

- Hadoop version 2.6 and above
- Spark version 1.6.0 and above

Note:

This product is verified on the following Hadoop Distributions.

- Hortonworks Data Platform 2.1 on Red Hat Linux 6.3
- Hortonworks Data Platform 2.4 on CentOS 7.0
- Hortonworks Data Platform 2.6 on CentOS 7.0
- Cloudera CDH 5.10 on CentOS 7.3
- Cloudera CDH 5.11 on CentOS 7.3
- Amazon EMR 5.5
- Amazon EMR 5.6

To use these jar files, 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.

To use the product, the following must be installed on your system:

for MapReduce:

- Java JDK version 1.7 or above
- Hadoop version 2.6.0 or above

for Spark:

- Java JDK version 1.7 or above
- Hadoop version 2.6.0 or above
- Spark version 1.6.0 or above

for Hive:

- Hive version 1.2.1 or above

for Hive Client

- For example, Beeline

for Zeppelin Notebook (not supported in Cloudera)

- Java JDK version 1.7 or above
- Hadoop version 2.6.0 or above
- Spark version 2.0 or above

2 - Routing UDFs

In this section

Overview	9
Working with Routing Hive UDFs	10

Overview

This section describes the Hive user-defined functions (UDFs) available for performing routing operations.

- **PointToPointRoute**: Helps generate the routing information required to travel between two distinct points.
- **IsoChrono**: Helps find points that can be reached from the specified starting location in the specified duration.
- **IsoDistance**: Helps find points that lie at some specified distance from the given starting location, considering every possible route.

Working with Routing Hive UDFs

Setting Up Routing Hive UDFs

This section takes you through the steps you need to perform to setup the Routing Hive UDFs:

1. Log on to Name Node using an FTP tool, for example, WinSCP.
2. Copy the `spectrum-bigdata-routing-2.6.1.zip` file to the home directory.
3. Use the following command to create a new folder, `PB`, in the root folder:

```
sudo mkdir /PB
```

4. Use the following code to grant access rights to the new folder and all its sub-folders to all other logged-in users.

```
sudo chown -R centos:centos /PB
```

5. Unzip the file `spectrum-bigdata-routing-2.6.1.zip` and extract its contents in the new folder, `PB`, as shown below:

```
unzip spectrum-bigdata-routing-2.6.1.zip -d /PB/spectrum-bigdata-routing
```

/PB/spectrum-bigdata-routing				
Name	Size	Changed	Rights	Owner
📁		8/9/2017 2:41:35 PM	rwxr-xr-x	centos
📁 hive		7/26/2017 9:52:20 AM	rwxr-xr-x	centos
📁 legal		7/26/2017 9:52:24 AM	rwxr-xr-x	centos
📄 Routing_UserGuide.pdf	2,378 KB	7/26/2017 9:38:38 AM	rw-r--r--	centos
📄 SpectrumSpatialBigData_ReleaseNotes.pdf	855 KB	7/26/2017 9:38:42 AM	rw-r--r--	centos

Setting Up Routing Data

Routing Hive UDFs are required in order to access a Routing dataset during its operation.

Note the location of the `dbList.json` configuration file; it contains details about the Routing datasets that are installed.

To know how to setup a Routing dataset, refer to the *Data Installation* section in the *Global Routing API User Guide*.

Running the Routing Hive Job from Command-Line

This section takes you through the steps you need to perform to run the Routing Hive job from the command-line.

1. Use the following command to start the Hive shell:

```
$ hive
```

Alternatively, use the following command to start Hive in INFO mode:

```
$ hive --hiveconf hive.root.logger=INFO,console
```

2. Use the following command to add the Routing JAR to Hive:

```
add jar /dir/on/server/spectrum-bigdata-spatial-routing-hive-2.6.1.jar;
```

3. Set the mandatory and optional system properties as needed. The `dbConfigJSONFile` system property is a mandatory property which specifies the location of the configuration file. Routing Hive UDFs require the following properties to be set in order to start a GRA engine with the given configurations:

- **dbConfigJSONFile** (Mandatory): Specifies the local file system path of the `dblist.json` file. Use the following command in the Hive console to set this property:

```
set
system:dbConfigJSONFile=<path_to_folder_containing_dblist.json>/dblist.json;
```

- **routeTimeout** (Optional): Specifies the amount of time, in milliseconds, allowed for a point-to-point routing request to be processed and completed. If the request is not processed or completed within the specified duration, it is terminated. Use the following command in the Hive console to set this property:

```
set system:routeTimeout=10000;
```

- **routeDefaultCoordSys** (Optional): Specifies the coordinate system to be used by default. Use the following command in the Hive console to set this property:

```
set system:routeDefaultCoordSys="EPSG:4326";
```

- **isoTimeout** (Optional): Specifies the amount of time (in milliseconds) allowed for a boundary routing request to be processed and completed. If the request is not processed or completed

within the specified duration, it is terminated. Use the following command in the Hive console to set this property:

```
set system:isoTimeout=17000;
```

- **allowFallback** (Optional): This is a Boolean property that controls whether the engine can use major roads in case the other roads cannot be used. If set to `true`, the engine can use the major roads. Use the following command in the Hive console to set this property:

```
set system:allowFallback=true;
```

- **shortProcessThreads** (Optional): Specifies the number of threads to be used for a point-to-point route requests and routing data requests. Use the following command in the Hive console to set this property:

```
set system:shortProcessThreads=16;
```

- **longProcessThreads** (Optional): Specifies the number of threads to be used for isochrone requests. Use the following command in the Hive console to set this property:

```
set system:longProcessThreads=16;
```

- **graEngineTimeoutInSeconnds** (Optional): Specifies the duration (in seconds) in which a GRA engine is disposed if it does not receive any request. The default value of this property is -1, which means that the GRA engines will never time out. Use the following command in the Hive console to set this property:

```
set system:graEngineTimeoutInSeconnds=10;
```

If you do not want the GRA engines to be disposed, setting this property is not required. If you have already specified a value for this property, but later you do not want the GRA engines to be disposed, leave the property blank or set it to -1.

```
set system:graEngineTimeoutInSeconnds=;
set system:graEngineTimeoutInSeconnds=-1
```

4. Register the Routing UDFs using the commands given below:

```
hive> create temporary function PointToPointRoute as
'com.pb.bigdata.spatial.routing.hive.PointToPointRoute';
```

```
hive> create temporary function IsoChrone as
'com.pb.bigdata.spatial.routing.hive.IsoChrone';
```

```
hive> create temporary function IsoDistance as
'com.pb.bigdata.spatial.routing.hive.IsoDistance';
```

The Routing functions are now ready to use.

For more information, see the [PointToPointRoute](#) on page 13, [IsoChrone](#) on page 15, or [IsoDistance](#) on page 16.

PointToPointRoute

Syntax

```
Select PointToPointRoute(X_Column_Of_Start_Point,
Y_Column_Of_Start_Point,
Start_Point_CRS, X_Column_Of_End_Point, Y_Column_Of_End_Point,
End_Point_CRS) From Table_Name;
```

Description

The `PointToPointRoute` UDF helps extract the routing information required for traveling between two distinct points. It takes the starting location, the ending location, the distance (this is optional) and the time units, and it returns the total distance and duration of a route that is either the fastest or the shortest.

Parameters

<i>X_Column_Of_Start_Point</i>	The X coordinate of the starting location.
<i>Y_Column_Of_Start_Point</i>	The Y coordinate of the starting location
<i>Start_Point_CRS</i>	The coordinate reference system used for the starting location.
<i>End_Point_CRS</i>	The coordinate reference system used for the ending location.
<i>X_Column_Of_End_Point</i>	The X coordinate of the ending location.
<i>Y_Column_Of_End_Point</i>	The Y coordinate of the ending location.

Return Values

Depending on the query, this function returns values described in the table below:

Time	The shortest duration of traveling between two points in minutes.
------	---

Distance

The total distance of the shortest path between two points in metres.

Note: To know more about the supported distance and time units, refer to the *Pitney Bowes Global Routing API SDK Product Guide*.

Example

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
  42.677640, "epsg:4326" );
```

This function call returns the distance of the fastest route between two points in metres and the total travel duration in minutes.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
  42.677640, "epsg:4326" ).distance;
```

This function call returns the distance of the shortest route between two points in metres.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
  42.677640, "epsg:4326" ).time;
```

This function call returns the duration of the fastest route between two points in minutes.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
  42.677640, "epsg:4326", "ft", null );
```

This function call returns the distance of the fastest route between two points in feet and the total travel duration in minutes.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
  42.677640, "epsg:4326", null, "msec" );
```

This function call returns the distance of the fastest route between two points in feet and the total travel duration milliseconds.

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326", -73.693102,
  42.677640, "epsg:4326", null, null );
```

This function call returns the distance of the fastest route between two points in metres and the total travel duration in minutes.

The Routing Hive UDFs are interoperable with the Spatial and Geocoding Hive UDFs, as shown in the example given below:

```
select PointToPointRoute(Geocode("485A", "Watervliet Shaker
Rd", "", "Latham", "NY", "12110", "usa").geometry, Geocode("204-240", "Jordan
Rd", "", "Troy", "NY", "12180", "usa").geometry);
```

Note: See the [Hive User Defined Geocoding Functions](#) section in the Spectrum™ Geocoding for Big Data User Guide or [Hive User Defined Spatial Functions](#) section in the Spectrum™ Location Intelligence for Big Data User Guide for more information on how to use the Geocoding and Spatial UDFs.

IsoChrone

Syntax

```
Select IsoChrone(X_Column, Y_Column, Coordinate_Reference_System,
Cost_Time, Time_Unit) From Table_Name;
```

Description

An isochrone is a polygon connecting points that take the same amount of time to reach from a particular point. This function returns an isochrone connecting a set of points that can be reached from the specified starting location in the given duration.

The `IsoChrone` UDF uses the `GetTravelBoundary` operation of the Global Routing API to generate an IsoChrone. The `GetTravelBoundary` operation helps determine the drive or walk time or the distance boundary from a given location. This function generates polygons corresponding to an isochrone or isodistance calculation.

Parameters

<i>X_Column</i>	Specifies the X coordinate of the starting location.
<i>Y_Column</i>	Specifies the Y coordinate of the starting location.
<i>Coordinate_Reference_System</i>	Specifies the coordinate reference system to be used.
<i>Cost_Time</i>	Specifies the time available to reach the end point.
<i>Time_Unit</i>	Specifies the unit of time.

Return Value

Polygon Geometry	The <code>WritableGeometry</code> of an isochrone.
------------------	--

Note: To know more about the supported time units, refer to the *Pitney Bowes Global Routing API SDK* product guide.

Example

```
Select IsoChrone ("-77.088217", 38.937072, "epsg:4326", 3, "min");
```

```
Select IsoChrone(Points.X, Points.Y, 3, "min") From Points;
```

Both these queries return a polygon geometry comprising all the points that can be reached from the specified starting point in the given time, that is, three minutes.

Furthermore, you can use the `WritableGeometry` returned by the `IsoChrone` UDF with the Spatial UDFs for further processing, as shown in the examples given below:

```
Select ToWKT(IsoChrone(true, 38.937072, "epsg:4326", 3, "min"));
```

```
select Within(FromWKT("POINT(-77.088217 38.937072)",
"epsg:4326"), IsoChrone ("-77.088217", 38.937072, "epsg:4326", 3, "min"));
```

IsoDistance

Syntax

```
Select IsoDistance(X_Column, Y_Column, Coordinate_Reference_System,
Cost_Distance, Linear_Unit) From Table_Name;
```

Description

An `Isodistance` is a polygon representing specific distance intervals from a particular point extended out along all possible paths. This function returns an `isodistance` connecting a set of points that lie at a specified distance from the given starting point, considering every possible route.

The `IsoDistance` UDF uses the `GetTravelBoundary` operation of the Global Routing API to generate the `Isodistance`. The `GetTravelBoundary` operations helps determine the drive or walk time or the distance boundary from a given location. This function generates polygons corresponding to an `isochrone` or `isodistance` calculation.

Parameters

<i>X_Column</i>	Specifies the X coordinate of the starting location.
<i>Y_Column</i>	Specifies the Y coordinate of the starting location.
<i>Coordinate_Reference_System</i>	Specifies the coordinate reference system to be used.

<i>Cost_Distance</i>	Specifies the distance to be travelled from the available to reach the end point.
----------------------	---

<i>Linear_Unit</i>	Specifeis the unit of distance.
--------------------	---------------------------------

Return Values

Polygon Geometry	Returns the <code>WritableGeometry</code> of an isodistance.
------------------	--

Note: For values of distance linear units, refer to the *Pitney Bowes Global Routing API SDK* product guide.

Example

```
Select IsoDistance (-77.088217, 38.937072, "epsg:4326", 3, "km");
```

```
Select IsoDistance (Points.X, Points.Y, 3, "km") From Points;
```

Both these queries return a polygon geometry comprising all the points that lie at the specified distance from the starting point.

You can use the `WritableGeometry` returned by `IsoDistance` with the Spatial UDFs for further processing, as shown below:

```
Select ToWKT(IsoDistance (-77.088217, 38.937072, "epsg:4326", 3, "km"));
```

```
Select Within(FromWKT("POINT(-77.088217 38.937072)", "epsg:4326"),
IsoDistance (-77.088217, 38.937072, "epsg:4326", 3, "km"));
```

3 - Security

In this section

Authentication	19
Hive - Kerberos Authentication	19
Hive - Sentry Authorization	20

Authentication

This section describes how to configure authentication for a Kerberos secured cluster.

By default, Hive runs in a non-secure mode where authentication is not required. By configuring the secure mode, each user and service needs to be authenticated by Kerberos, in order to use Hive services.

Hive - Kerberos Authentication

Steps to Run Hive Jobs on Hadoop Cluster:

1. Login into Hive shell using `Kinit` command and provide principal and keytab using `Kinit` command as shown below:

```
--kinit centos -kt /PB/spectrum-bigdata-routing/myKeyTab.keytab
```

2. Start the Hive console:

```
--hive
```

3. Login with `kadmin` on KDC server using following command:

```
$sudo kadmin.local
```

4. Add principal using the following command:

```
kadmin.local: addprinc<name of user/service>/<instance>@REALM.COM
```

5. Add principal to an existing keytab using the following command:

```
kadmin.local: ktadd -k /PB/spectrum-bigdata-routing/default.keytab<name of user/service>/<instance>@REALM.COM
```

Example Usage:

```
sudo kadmin.local
```

```
kadmin.local: addprinc centos@EXAMPLE.COM
```

```
kadmin.local: ktadd -k /PB/spectrum-bigdata-routing/myKeyTab.keytab centos@EXAMPLE.COM
```

6. Login into existing principal using a password.

```
$kinit <name of user/service>/<instance>@REALM.COM
```

7. Login into existing principal using a keytab.

```
$kinit <name of user/service>/<instance>@REALM.COM -kt  
/PB/spectrum-bigdata-routing/myKeyTab.keytab
```

8. Logout or remove TGT:

```
$kdestroy
```

Hive - Sentry Authorization

It is assumed that an authorization service is already configured and started. Follow the steps for executing a user-defined function:

1. Initialize the user with keytab using `kinit`. If keytab is used, authentication using user Id and password is not required.

For example:

```
kinit hive/<HOST>@<REALM> -k -t /local/path/to/hive.keytab
```

2. Enter the beeline shell and connect to Hiveserver2

For example:

```
!connect jdbc:hive2://<HOST>:<PORT>/;principal= hive/<HOST>@<REALM>
```

3. Make sure that the user has minimum privileges required to select a database, table or required columns.

4. To add a user-defined function in beeline shell, follow the steps below:

- Copy the user-defined function jar file to a local directory on the node where Hiveserver2 is installed.
- Add the directory location of the Jar file in the environment variable `HIVE_AUX_JARS_PATH`
- Restart the Hive service.
- Enter the beeline shell and ensure that the user has the permission for `GRANT` and `Create function`.
- Grant permission on the jar file location. The directory path must be the same as set in the `HIVE_AUX_JARS_PATH` environment variable. For example:

```
GRANT ALL ON URI 'file:///aux/jars/path/spectrum-bigdata-routing-hive-2.6.1.jar' TO  
ROLE admin;
```

- Register a user-defined function. \

In the case of `PointToPointRoute`, for example, register using the following command:

```
create temporary function PointToPointRoute as  
'com.pb.bigdata.spatial.routing.hive.PointToPointRoute';
```

5. Make sure that you have minimum select privileges to execute a user-defined function.

In the case of `PointToPointRoute`, for example:

```
select PointToPointRoute(-73.750333, 42.736103, "epsg:4326",-73.693102,  
42.677640, "epsg:4326" );
```

Copyright

Information in this document is subject to change without notice and does not represent a commitment on the part of the vendor or its representatives. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, without the written permission of Pitney Bowes Software Inc., 350 Jordan Road, Troy, New York 12180.

© 2017 Pitney Bowes Software Inc. All rights reserved. Location Intelligence APIs are trademarks of Pitney Bowes Software Inc. All other marks and trademarks are property of their respective holders.



3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com