pitney bowes

Spectrum™ Routing for Big Data

Version 3.2

Spectrum™ Routing for Big Data User Guide

Table of Contents

1 - Welcome

In this section

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 Hive's 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 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, IsoChrone, and IsoDistance)
- Input datasets and metadata

API Types:

- Pre-built Hive UDF wrappers for Routing operations
- Core Routing APIs

System Requirements and Dependencies

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

This product is verified on the following Hadoop distributions.

- Cloudera 5.12 and 6.0
- Hortonworks 2.6 and 3.0
- EMR 5.10
- MapR 6.0 and above, with MapR Expansion Pack (MEP) 5.0.0

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

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

for Hive:

- Hive version 1.2.1 or above

for Hive Client

- Beeline, for example

for Spark and Zeppelin Notebook:

- Java JDK version 1.8 or above

- Hadoop version 2.6.0 or above
- Spark version 1.6.0 or above (2.0 or above required for MapR and Cloudera 6.0)
- Zeppelin Notebook is not supported in Cloudera

2 - Hive User-Defined Routing Functions

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

- **PointToPointRoute:** returns the time and distance to travel from one location to another
- **IsoChrone:** returns a polygon that represents the maximum geographic area you can travel in a specified amount of time from a specified location
- **IsoDistance:** returns a polygon that represents the maximum geographic area you can travel in a specified distance from a specified location

In this section

Hive Variables

The Routing UDFs use Hive variables to set various properties, which take precedence over the system properties. They can also be set as part of an option in a UDF query, where they will take precedence over the Hive variables as well as the system properties.

Variable	Required	Example
`pb.routing.config.location` Location of dbList.json containing the routing configuration.	Yes	hdfs:///pb/routing/config/dbList.json /mapr/<*cluster-name*>/pb/routing/config/dbList.json
`pb.routing.engine.timeout` Specifies the duration (in milliseconds) after which a GRA engine shuts down 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.	No	300
`pb.routing.routeTimeout` Specifies the amount of time (in milliseconds) allowed for a point-to-point routing job to be processed and completed. If the job is not processed or completed within the specified duration, the SQL returns null to the console and exception to logs for each record processed. This property can be used for the PointToPointRoute UDF.	No	10000
`pb.routing.isoTimeout` Specifies the amount of time (in milliseconds) allowed for a boundary routing job to be processed and completed. If the job is not processed or completed within the specified duration, the SQL returns null to the console and exception to logs for each record processed. This property can be used for the IsoDistance and IsoChrone UDFs	No	17000

Variable	Required	Example
`pb.routing.allowFallback` This Boolean property controls whether the engine can use major roads in case other roads cannot be used. If set to true, the engine can use major roads.	No	false
`pb.download.location` **Note:** Use only if reference data was distributed remotely via HDFS or S3. Location of the directory where reference data will be downloaded to. This path must exist on every data node and the HiveServer2 node.	No	/pb/downloads
`pb.download.group` **Note:** Use only if reference data was distributed remotely via HDFS or S3. This is an optional property and only specific to POSIX-compliant platforms like Linux. It specifies the operating system group which should be applied to the downloaded data on a local file system, so that each Hadoop service can update the data when required. This group should be present on all nodes in the cluster and the operating system user executing the Hadoop service should be a part of this group. For more information, see **Download Permissions** on page 37.	No	pbdownloads
`pb.routing.error.limit` The number of routing errors to allow before failing a task for "too many errors". This prevents a task (and thus the job) from continuing in the case of a likely configuration error. The default value is 10.	No	1
`pb.routing.error.limit.disabled` Disables the error limit. All errors will be logged but will not cause the task or job to fail.	No	true

Setting Variables

Since these are Hive configuration variables, you can set them permanently by editing the hiveserver2-site.xml file in your cluster.

Example (using HDFS)

```
pb.routing.config.location=hdfs:///pb/routing/config/dbList.json;
pb.download.location = /pb/downloads
pb.download.group= pbdownloads
```

Example (using MapR)

```
pb.routing.config.location=/mapr/<cluster-name>/pb/routing/config/dbList.json;
```

Alternatively you can set them temporarily in each Hive session that you open.

For example, if you wish to perform routing in Beeline you can set the variables in the following way:

Example (using HDFS)

```
set pb.routing.config.location=hdfs:///pb/routing/config/dbList.json;
set pb.routing.allowFallback=true;
set pb.download.location = /pb/downloads;
set pb.download.group= pbdownloads;
```

Example (using MapR)

```
set
pb.routing.config.location=/mapr/<cluster-name>/pb/routing/config/dbList.json;
set pb.routing.allowFallback=true;
```

Variables set in the local session will override any hive-site variables. This means you can have default values set in the hiveserver2-site file and override them in Beeline when necessary.

System Variables (Deprecated)

Hive variables used to exist in the system namespace. This is deprecated as of version 3.1 and no longer the recommended method.

Engine Options

These options govern the operation of the routing engine. When they are set as part of an option in a UDF query, they will take precedence over the Hive variables as well as the system properties.

Hive Variable/System Property	Type	Description
`pb.routing.config.location`	String	Location of dbList.json containing the routing configuration.
`pb.routing.engine.timeout`	Number	The amount of time in milliseconds that will trigger a timeout in the routing engine.
`pb.routing.routeTimeout`	Number	The amount of time in milliseconds that will trigger a timeout when calculating a route (also used in the Global Routing SDK).
`pb.routing.isoTimeout`	Number	The amount of time in milliseconds that will trigger a timeout when calculating an isochrone or isodistance (also used in the Global Routing SDK).
`pb.routing.allowFallback`	Boolean	Fall back to use only the major roads in the network if a minor road cannot be found (also used in the Global Routing SDK).
`pb.download.location` **Note:** Use only if reference data was distributed remotely via HDFS or S3.	String	Location of the directory where reference data will be downloaded to. This path must exist on every data node and the HiveServer2 node.
`pb.download.group` **Note:** Use only if reference data was distributed remotely via HDFS or S3.	String	This is an optional property and only specific to POSIX-compliant platforms like Linux. It specifies the operating system group which should be applied to the downloaded data on a local file system, so that each Hadoop service can update the data when required. This group should be present on all nodes in the cluster and the operating system user executing the Hadoop service should be a part of this group. For more information, see **Download Permissions** on page 37

PointToPointRoute

Description

The `PointToPointRoute` UDF helps extract the routing information required for traveling between two distinct points. It takes the starting and ending locations, then returns the total distance and duration of a route that is either the fastest or the shortest (as best calculated by the algorithm used).

> **Note:** The function may time out when using a large number of datasets that are stored in remote locations (such as HDFS and S3). If you are using Hive with the MapReduce engine, you can adjust the value of the `mapreduce.task.timeout` property.

Syntax

```
PointToPointRoute(WriteableGeometry startPoint, WriteableGeometry
endPoint, Map<String, String> options)
```

Parameters

Parameter	Type	Description
startPoint	WriteableGeometry	starting location. This must be a point for those UDFs that require it.
endPoint	WriteableGeometry	ending location. This must be a point for those UDFs that require it.
options	**Map**	**options** that affect the calculation of the route and **engine options** that affect the operation of the routing engine

Syntax

```
PointToPointRoute(Number|String startX, Number|String startY,
Number|String endX, Number|String endY, Map<String, String> options)
```

Parameters

Parameter	Type	Description
startX	Number or String	x ordinate of the starting location in WGS 84
startY	Number or String	y ordinate of the ending location in WGS 84
endX	Number or String	x ordinate of the starting location in WGS 84
endY	Number or String	y ordinate of the ending location in WGS 84
options	**Map**	**options** that affect the calculation of the route and **engine options** that affect the operation of the routing engine

Options

The *options* keys cannot be derived from a column in the table that the UDF is executed on. The keys of the *options* map must also be of type String or convertible to String, and the values should be convertible to a type appropriate for the key's value (typically this means that all values must be of type String). Keys and values of any type can be enclosed in either double or single quotes; a number or Boolean value will be processed correctly with or without quotes. For information on options that affect the calculation of the route, see **Engine Options** on page 10.

Key	Type	Description
pb.routing.historicSpeedBucket	String	Specifies whether the routing calculation uses the historic traffic speeds. These speeds are based on different time-of-day buckets. The data must have historic traffic speeds included in order to use this feature. The data for each country/region has the same bucket definitions, where the speeds for these bucket values may vary. The available values are: **None** The default value. Historic traffic data is not used in the calculation. Instead an averaged speed value is used. **AMPeak** Calculate routes with the peak AM speeds. The AMPeak time bucket is from 07:00 to 10:00. **PMPeak** Calculate routes with the peak PM speeds. The PMPeak time bucket is from 16:00 to 19:00. **OffPeak** Calculate routes with the off peak (daytime) speeds. The OffPeak time bucket is from 10:00 to 16:00. **Night** Calculate routes with the nighttime speeds. The Night time bucket is from 22:00 to 04:00.
pb.routing.majorRoads	Boolean	Whether to include all roads in the calculation or just major roads. If you choose to include only major roads (that is, set the value to true), performance will improve but accuracy may decrease. The default value is false.
pb.routing.optimizeBy	String	The type of optimizing to use for the route. Valid values are *time* or *distance*. The default value is time.

Key	Type	Description
pb.routing.returnDistanceUnit		The unit to return distance. If not specified then this value is meter. Available linear units are: • in • inch • ft • foot • yd • yard • mi • mile • mm • millimeter • cm • centimeter • m • meter • km • kilometer
pb.routing.returnTimeUnit		The unit to return time. If not specified, the default value is minute. Available time unit values are: • msec • millisecond • sec • s • second • m • min • minute • h • hr • hour • day • week
pb.routing.error.limit	Number	The number of routing errors to allow before failing a task for "too many errors". This prevents a task (and thus the job) from continuing in the case of a likely configuration error. The default value is 10. Optionally you can use the equivalent Hive variable, `pb.routing.error.limit`.

Key	Type	Description
pb.routing.error.limit.disabled	Boolean	Disables the error limit. All errors will be logged but will not cause the task or job to fail. Optionally you can use the equivalent Hive variable, `pb.routing.error.limit.disabled`.

Return Values

This function returns values described in the table below:

Distance	The total distance of the shortest path between two points in meters.
Time	The shortest duration of time it takes to travel between two points in minutes.
Error	Any error that occurs; if none, then null.

Note: For more information about the supported distance and time units, see the ***Global Routing SDK User Guide***, which is also available on the **Spectrum Spatial for Big Data** documentation landing page.

Examples

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

```
SELECT PointToPointRoute(-77.750333, 38.736103, -77.693102, 38.677640);
```

This function call returns the distance of the shortest route between two points in meters with all roads included:

```
SELECT PointToPointRoute(ST_Point(-77.750333, 39.736103),
ST_Point(-77.693102, 38.677640), map('pb.routing.majorRoads',
'false')).distance;
```

The Routing Hive UDFs are also interoperable with the Spatial and Geocoding Hive UDFs:

```
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 *Spectrum™ Geocoding for Big Data User Guide* and *Spectrum™ Location Intelligence for Big Data User Guide* on the **Spectrum Spatial for Big Data** documentation landing page for more information on how to use the Geocoding and Spatial UDFs.

Sometimes calculating a route takes longer than expected and times out. You can adjust this using the engine options:

```
SELECT PointToPointRoute(-77.750333, 38.736103, -77.693102, 38.677640,
 map('pb.routing.engine.timeout', 300)) FROM <table_name>;
```

The previous example had the time-out specified as an number. It can also be specified as a string:

```
SELECT PointToPointRoute(-77.750333, 38.736103, -77.693102, 38.677640,
 map('pb.routing.engine.timeout', '300')) FROM <table_name>;
```

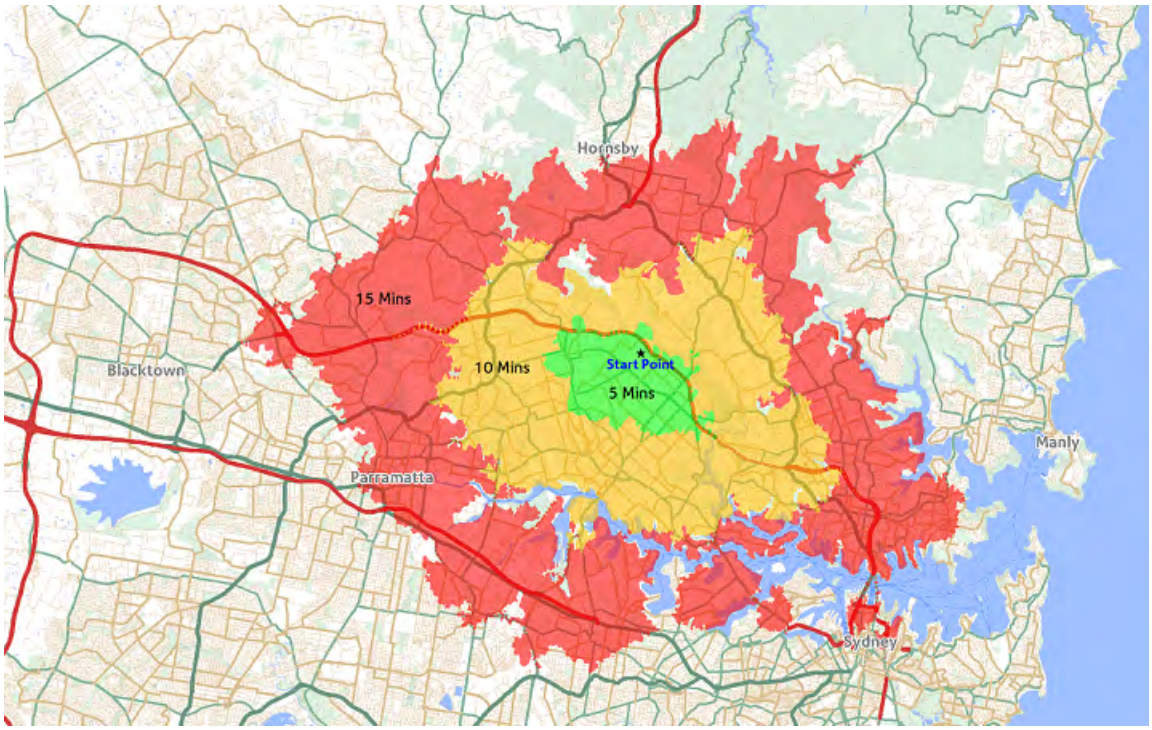
Hive requires that all values in a MAP must be of the same type:

```
SELECT PointToPointRoute(-77.750333, 38.736103, -77.693102, 38.677640,
 map('pb.routing.engine.timeout', '300', 'pb.routing.allowFallback',
'false')) FROM <table_name>;
```

IsoChrone

Description

An isochrone is a polygon connecting points that take the same amount of time to reach from a particular point. The IsoChrone 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 calculation. For more information, see the ***Global Routing SDK User Guide***, which is also available on the **Spectrum Spatial for Big Data** documentation landing page.

> **Note:** The function may time out when using a large number of datasets that are stored in remote locations (such as HDFS and S3). If you are using Hive with the MapReduce engine, you can adjust the value of the `mapreduce.task.timeout` property.

Syntax

```
IsoChrone(WriteableGeometry startPoint, Number|String cost, Map<String,
 String> options, Map<String, String> ambientOptions)
```

```
IsoChrone(WriteableGeometry startPoint, Array<Number|String> costs,
Map<String, String> options, Map<String, String> ambientOptions)
```

Parameters

Parameter	Type	Description
startPoint	WriteableGeometry	the location from which travel starts. This must be a point for those UDFs that require it.
cost	Number or String	contains a single value for the maximum travel time (default unit is minute).
costs	**Array**	contains multiple values for the maximum travel time (default unit is minute). When this parameter is provided, the UDF will return an array of responses, one for each cost, in the same order as the costs parameter.
options	**Map**	**options** that affect the generation of the isochrone and **engine options** that affect the operation of the routing engine
ambientOptions	**Map**	**ambient road type options** that affect the distance traveled when going off-network

Syntax

```
IsoChrone(Number|String startX, Number|String startY, Number|String
cost, Map<String, String> options, Map<String, String> ambientOptions)
```

```
IsoChrone(Number|String startX, Number|String startY,
Array<Number|String> costs, Map<String, String> options, Map<String,
String> ambientOptions)
```

Parameters

Parameter	Type	Description
startX	Number or String	x ordinate of the location from which travel starts in WGS 84
startY	Number or String	y ordinate of the location from which travel starts in WGS 84
cost	Number or String	contains a single value for the maximum travel time (default unit is minute).
costs	**Array**	contains multiple values for the maximum travel time (default unit is minute). When this parameter is provided, the UDF will return an array of responses, one for each cost, in the same order as the costs parameter.
ambientOptions	**Map**	**ambient road type options** that affect the distance traveled when going off-network

Options

The *options* and *ambientOptions* keys cannot be derived from a column in the table that the UDF is executed on. The keys of the *options* and *ambient-options* maps must also be of type String or convertible to String, and the values should be convertible to a type appropriate for the key's value (typically this means that all values must be of type String). Keys and values of any type can be enclosed in either double or single quotes; a number or Boolean value will be processed correctly with or without quotes. For information on options that affect the calculation of the route, see **Engine Options** on page 10.

Key	Type	Description
pb.routing.bandingStyle	BandingStyle	The style of banding to be used in the result. Banding styles are the types of multiple distance bands that can be displayed based on multiple costs. The available values are: **Donut** Each boundary is determined by subtracting out the next smallest boundary. **Encompassing** Each boundary is determined independent of all others. This is the default method.
pb.routing.historicSpeedBucket	HistoricSpeedBucket	Specifies whether the routing calculation uses the historic traffic speeds. These speeds are based on different time-of-day buckets. The data must have historic traffic speeds included in order to use this feature. The data for each country/region has the same bucket definitions, where the speeds for these bucket values may vary. The available values are: **None** The default value. Historic traffic data is not used in the calculation. Instead an averaged speed value is used. **AMPeak** Calculate routes with the peak AM speeds. The AMPeak time bucket is from 07:00 to 10:00. **PMPeak** Calculate routes with the peak PM speeds. The PMPeak time bucket is from 16:00 to 19:00. **OffPeak** Calculate routes with the off peak (daytime) speeds. The OffPeak time bucket is from 10:00 to 16:00. **Night** Calculate routes with the nighttime speeds. The Night time bucket is from 22:00 to 04:00.
pb.routing.majorRoads	Boolean	Whether to include all roads in the calculation or just major roads. If you choose to include only major roads (that is, set the value to true), performance will improve but accuracy may decrease. The default value is false.

Key	Type	Description
pb.routing.returnHoles	Boolean	Specifies whether you want to return holes, which are areas within the larger boundary that cannot be reached within the desired time or distance, based on the road network. The default value is false.
pb.routing.returnIslands	Boolean	Specifies whether you want to return islands, which are small areas outside the main boundary that can be reached within the desired time or distance. The default value is false.
pb.routing.simplificationFactor	Number	What percentage of the original points should be returned or upon which the resulting complexity of geometries should be based. A number between 0.0 and 1.0 is accepted, exclusive of 0.0 but inclusive of 1.0. Complexity increases as the value increases, therefore 1.0 means the most complex. The default value is 0.5.
pb.routing.maxOffroadDistance	Number	The maximum distance to allow travel off the road network using the *pb.routing.maxOffroadDistanceUnit*. Examples of off-network roads include driveways and access roads. For example, if you specify a maximum off road distance of 1 mile the travel boundary will extend no further than one mile from the network road. If you specify a value of 0 the travel boundary will not extend beyond the road itself. Use the ambient speed options to specify the speed of travel along non-network roads.

Key	Type	Description
pb.routing.maxOffroadDistanceUnit	String	The distance unit defining the *pb.routing.maxOffroadDistance*. You must also define *pb.routing.maxOffroadDistance* if you define this key. Available linear units are: • in • inch • ft • foot • yd • yard • mi • mile • mm • millimeter • cm • centimeter • m • meter • km • kilometer
pb.routing.timeUnit	String	The unit to return time. If not specified, the default value is minute. Available time unit values are: • msec • millisecond • sec • s • second • m • min • minute • h • hr • hour • day • week

Key	Type	Description
pb.routing.defaultAmbientSpeed	Number	The speed to travel when going off a network road to find the travel boundary (for all road types). To control how off-network travel is used in the travel boundary calculation, you need to specify the speed of travel off the road network (the ambient speed). Ambient speed can affect the size and shape of the travel boundary polygon. In general, the faster the ambient speed, the larger the polygon. For example, if you were at a point with 5 minutes left, and if the ambient speed were 15 miles per hour, boundary points would be put at a distance of 1.25 miles. If the ambient speed were reduced to 10 miles per hour, boundary points would be put at a distance of 0.83 miles. The default value is 15.
pb.routing.ambientSpeedUnit	String	The unit of measure to use to calculate the ambient speed. Available speed units are: MPH (miles per hour), KPH (kilometers per hour), MTPS (meters per second), and MTPM (meters per minute). The default value is MPH.
pb.routing.error.limit	Number	The number of routing errors to allow before failing a task for "too many errors". This prevents a task (and thus the job) from continuing in the case of a likely configuration error. The default value is 10. Optionally you can use the equivalent Hive variable, `pb.routing.error.limit`.
pb.routing.error.limit.disabled	Boolean	Disables the error limit. All errors will be logged but will not cause the task or job to fail. Optionally you can use the equivalent Hive variable, `pb.routing.error.limit.disabled`.

Ambient Road Type Options

This option lets you adjust the distance traveled when going off-network

Key	Type	Description
RoadType	Number	Specifies the ambient speed to use for off-network travel based on the road type. You must specify both the road type and the new speed for that road type. The speed is defined in the *options*. Road types can be returned in all supported types. For a list of road type enumerations, see **RoadType Enumeration** on page 36.

Return Value

This function returns struct values described in the table below. In the case where a single *cost* is input, a single struct is returned. In the case where an array of multiple *cost* values are input, then an array of structs are returned.

Geometry	The `WritableGeometry` of an isochrone.
Error	Any error that occurs; if none, then null.

Examples

This query returns a polygon geometry comprising all the points that can be reached from the specified starting point in the specified time of three minutes. Because the return is in the form of a struct, you need to access the geometry field. You can use the `WritableGeometry` returned by the IsoChrone UDF with the Spatial UDFs for further processing, as shown in the example given below:

```
SELECT ToWKT(IsoChrone(ST_Point(-77.088217, 38.937072), 600,
map('pb.routing.timeUnit', 'second')).geometry);
```

The Routing Hive UDFs are also interoperable with the Spatial and Geocoding Hive UDFs. To avoid returning a geometry from the top of the query which can be problematic, the geometry is wrapped in a call to ToWKT in order to get a text representation:

```
SELECT ToWKT(IsoChrone(Geocode('485A Watervliet Shaker Rd, Latham, NY
12110','usa').geometry, 600).geometry);
```

> **Note:** See the *Spectrum™ Geocoding for Big Data User Guide* and *Spectrum™ Location Intelligence for Big Data User Guide* on the **Spectrum Spatial for Big Data** documentation landing page for more information on how to use the Geocoding and Spatial UDFs.

Sometimes calculating a route takes longer than expected and times out. You can adjust this using the engine options:

```
SELECT ToWKT(IsoChrone(x, y, cost, map(pb.routing.engine.timeout,
300)).geometry) FROM <table_name>;
```

The previous example had the time-out specified as an number. It can also be specified as a string:

```
SELECT ToWKT(IsoChrone(x, y, cost, map(pb.routing.engine.timeout,
'300')).geometry) FROM <table_name>;
```

Hive requires that all values in a MAP must be of the same type:

```
SELECT ToWKT(IsoChrone(x, y, cost, map(pb.routing.engine.timeout, '300',
 'pb.routing.allowFallback', 'false')).geometry) FROM <table_name>;
```

Example showing `ambient-options` set:

```
SELECT ToWKT(IsoChrone(ST_Point(-77.088217, 38.937072), 3, null,
map('footpath', '5')).geometry);
```

Example with both `options` and `ambient-options` set:

```
SELECT ToWKT(IsoChrone(ST_Point(-77.088217, 38.937072), 3,
map('pb.routing.timeUnit', 'minute'), map('footpath', '5')).geometry);
```

Example showing multiple costs specified in an array for an input point. Hive requires that all values in an array must be of the same type:

```
SELECT costs[costidx] cost, ToWKT(iso.geometry) wkt, iso.error
FROM points
LATERAL VIEW explode(array(array(5,10,20))) c as costs
LATERAL VIEW OUTER
posexplode(IsoChrone(points.longitude,points.latitude,costs)) isoresult
 as costidx, iso;
```

IsoDistance

Description

An isodistance is a polygon representing specific distance intervals from a particular point extended out along all possible paths. The IsoDistance 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 isodistance calculation. For more information, see the ***Global Routing SDK User Guide***, which is also available on the **Spectrum Spatial for Big Data** documentation landing page.

> **Note:** The function may time out when using a large number of datasets that are stored in remote locations (such as HDFS and S3). If you are using Hive with the MapReduce engine, you can adjust the value of the `mapreduce.task.timeout` property.

Syntax

```
IsoDistance(WriteableGeometry startPoint, Number|String cost, Map<String,
 String> options, Map<String, String> propagationFactorOptions)
```

```
IsoDistance(WriteableGeometry startPoint, Array<Number|String> costs,
Map<String, String> options, Map<String, String>
propagationFactorOptions)
```

Parameters

Parameter	Type	Description
startPoint	WriteableGeometry	the location from which travel starts. This must be a point for those UDFs that require it.
cost	Number or String	contains a single value for the maximum travel distance (default unit is meter)

Parameter	Type	Description
costs	**Array**	contains multiple values for the maximum travel distance (default unit is meter) . When this parameter is provided, the UDF will return an array of responses, one for each cost, in the same order as the costs parameter.
options	**Map**	**options** that affect the generation of the isodistance and **engine options** that affect the operation of the routing engine
propagationFactorOptions	**Map**	**propagation factor options** that affect the distance traveled when going off a network road

Syntax

```
IsoDistance(Number|String startX, Number|String startY, Number|String
cost, Map<String, String> options, Map<String, String>
propagationFactorOptions)
```

```
IsoDistance(Number|String startX, Number|String startY,
Array<Number|String> costs, Map<String, String> options, Map<String,
String> propagationFactorOptions)
```

Parameters

Parameter	Type	Description
startX	Number or String	x ordinate of the location from which travel starts in WGS 84
startY	Number or String	y ordinate of the location from which travel starts in WGS 84
cost	Number or String	contains a single value for the maximum travel distance (default unit is meter)

Parameter	Type	Description
costs	**Array**	contains multiple values for the maximum travel distance (default unit is meter) . When this parameter is provided, the UDF will return an array of responses, one for each cost, in the same order as the costs parameter.
options	**Map**	**options** that affect the generation of the isodistance and **engine options** that affect the operation of the routing engine
propagationFactorOptions	**Map**	**propagation factor options** that affect the distance traveled when going off a network road

Options

The *options* and *propagationFactorOptions* keys cannot be derived from a column in the table that the UDF is executed on. The keys of the *options* and *propagationFactorOptions* maps must also be of type String or convertible to String, and the values should be convertible to a type appropriate for the key's value (typically this means that all values must be of type String). Keys and values of any type can be enclosed in either double or single quotes; a number or Boolean value will be processed correctly with or without quotes. For information on options that affect the calculation of the route, see **Engine Options** on page 10.

Key	Type	Description
pb.routing.bandingStyle	BandingStyle	The style of banding to be used in the result. Banding styles are the types of multiple distance bands that can be displayed based on multiple costs. The available values are: **Donut** Each boundary is determined by subtracting out the next smallest boundary. **Encompassing** Each boundary is determined independent of all others.This is the default method.

Key	Type	Description
pb.routing.historicSpeedBucket	HistoricSpeedBucket	Specifies whether the routing calculation uses the historic traffic speeds. These speeds are based on different time-of-day buckets. The data must have historic traffic speeds included in order to use this feature. The data for each country/region has the same bucket definitions, where the speeds for these bucket values may vary. The available values are: **None** The default value. Historic traffic data is not used in the calculation. Instead an averaged speed value is used. **AMPeak** Calculate routes with the peak AM speeds. The AMPeak time bucket is from 07:00 to 10:00. **PMPeak** Calculate routes with the peak PM speeds. The PMPeak time bucket is from 16:00 to 19:00. **OffPeak** Calculate routes with the off peak (daytime) speeds. The OffPeak time bucket is from 10:00 to 16:00. **Night** Calculate routes with the nighttime speeds. The Night time bucket is from 22:00 to 04:00.
pb.routing.majorRoads	Boolean	Whether to include all roads in the calculation or just major roads. If you choose to include only major roads (that is, set the value to true), performance will improve but accuracy may decrease. The default value is false.
pb.routing.returnHoles	Boolean	Specifies whether you want to return holes, which are areas within the larger boundary that cannot be reached within the desired time or distance, based on the road network. The default value is false.
pb.routing.returnIslands	Boolean	Specifies whether you want to return islands, which are small areas outside the main boundary that can be reached within the desired time or distance. The default value is false.

Key	Type	Description
pb.routing.simplificationFactor	Number	What percentage of the original points should be returned or upon which the resulting complexity of geometries should be based. A number between 0.0 and 1.0 is accepted, exclusive of 0.0 but inclusive of 1.0. Complexity increases as the value increases, therefore 1.0 means the most complex. The default value is 0.5.
pb.routing.maxOffroadDistance	Number	The maximum distance to allow travel off the road network using the *pb.routing.maxOffroadDistanceUnit*. Examples of off-network roads include driveways and access roads. For example, if you specify a maximum off road distance of 1 mile the travel boundary will extend no further than one mile from the network road. If you specify a value of 0 the travel boundary will not extend beyond the road itself. Use the ambient speed options to specify the speed of travel along non-network roads.
pb.routing.maxOffroadDistanceUnit	String	The distance unit defining the *pb.routing.maxOffroadDistance*. You must also define *pb.routing.maxOffroadDistance* if you define this key. Available linear units are: • in • inch • ft • foot • yd • yard • mi • mile • mm • millimeter • cm • centimeter • m • meter • km • kilometer

Key	Type	Description
pb.routing.distanceUnit	String	The unit to return distance. If not specified then this value is meter. Available linear units are: • in • inch • ft • foot • yd • yard • mi • mile • mm • millimeter • cm • centimeter • m • meter • km • kilometer
pb.routing.error.limit	Number	The number of routing errors to allow before failing a task for "too many errors". This prevents a task (and thus the job) from continuing in the case of a likely configuration error. The default value is 10. Optionally you can use the equivalent Hive variable, `pb.routing.error.limit`.
pb.routing.error.limit.disabled	Boolean	Disables the error limit. All errors will be logged but will not cause the task or job to fail. Optionally you can use the equivalent Hive variable, `pb.routing.error.limit.disabled`.
pb.routing.propagationFactor	Number	The distance ratio to travel when going off a network road to find the travel boundary (for all road types). To control how off-network travel is used in the distance travel boundary calculation, you need to specify the propagation ratio. The propagation factor can affect the size and shape of the travel boundary polygon. In general, the higher the propagation factor the larger the polygon. For example, if there is 10 miles left to travel and the propagation factor was 0.15, then boundary points would be put at a distance of 1.5 miles.The default value is 0.16.

Propagation Factor Override Road Type Options

This option lets you adjust the distance traveled when going off-network.

Key	Type	Description
RoadType	Number	Specifies the propagation factor to use for off-network travel based on the road type. You must specify both the road type and the new propagation factor for that road type. The propagation factor is defined in the *options*. For a list of road type enumerations, see **RoadType Enumeration** on page 36.

Return Values

This function returns struct values described in the table below. In the case where a single *cost* is input, a single struct is returned. In the case where an array of multiple *cost* values are input, then an array of structs are returned.

Geometry	Returns the `WritableGeometry` of an isodistance.
Error	Any error that occurs; if none, then null.

Note: For values of distance linear units, see the ***Global Routing SDK User Guide***, which is also available on the **Spectrum Spatial for Big Data** documentation landing page.

Examples

This query returns a polygon geometry comprising all the points that lie at the specified distance of 3 kilometers from the starting point. Because the return is in the form of a struct, you need to access the geometry field. You can use the `WritableGeometry` returned by the IsoDistance UDF with the Spatial UDFs for further processing, as shown in the example given below:

```
SELECT ToWKT(IsoDistance(ST_Point(-77.088217, 38.937072), 1,
map('pb.routing.distanceUnit', 'km')).geometry);
```

The Routing Hive UDFs are also interoperable with the Spatial and Geocoding Hive UDFs. To avoid returning a geometry from the top of the query which can be problematic, the geometry is wrapped in a call to ToWKT in order to get a text representation:

```
SELECT ToWKT(IsoDistance(Geocode('485A Watervliet Shaker Rd, Latham, NY
 12110','usa').geometry, 500).geometry);
```

Note: See the *Spectrum™ Geocoding for Big Data User Guide* and *Spectrum™ Location Intelligence for Big Data User Guide* on the **Spectrum Spatial for Big Data** documentation landing page for more information on how to use the Geocoding and Spatial UDFs.

Sometimes calculating a route takes longer than expected and times out. You can adjust this using the engine options:

```
SELECT ToWKT(IsoDistance(x, y, cost, map(pb.routing.engine.timeout,
300)).geometry) FROM <table_name>;
```

The previous example had the time-out specified as an number. It can also be specified as a string:

```
SELECT ToWKT(IsoDistance(x, y, cost, map(pb.routing.engine.timeout,
'300')).geometry) FROM <table_name>;
```

Hive requires that all values in a MAP must be of the same type:

```
SELECT ToWKT(IsoDistance(x, y, cost, map(pb.routing.engine.timeout,
'300', 'pb.routing.allowFallback', 'false')).geometry) FROM <table_name>;
```

Example showing multiple costs specified in an array for an input point. Hive requires that all values in an array must be of the same type:

```
SELECT costs[costidx] cost, ToWKT(iso.geometry) wkt, iso.error
FROM points
LATERAL VIEW explode(array(array(5,10,20))) c as costs
LATERAL VIEW OUTER
posexplode(IsoDistance(points.longitude,points.latitude,costs)) isoresult
 as costidx, iso;
```

Example showing *propagationFactorOptions* set:

```
SELECT ToWKT(IsoDistance(ST_Point(-77.088217, 38.937072), 3, null,
map('footpath', '0.45')).geometry);
```

Example with both *options* and *propagationFactorOptions* set:

```
SELECT ToWKT(IsoDistance(ST_Point(-77.088217, 38.937072), 3,
map('pb.routing.timeUnit', 'minute'), map('footpath', '0.45')).geometry);
```

3 - Appendix

In this section

RoadType Enumeration

Use the following enumerations when defining the *RoadType* for ambient speeds:

- access way
- back road
- connector
- ferry
- footpath
- limited access dense urban
- limited access rural
- limited access suburban
- limited access urban
- local road dense urban
- local road rural
- local road suburban
- local road urban
- major local road dense urban
- major local road rural
- major local road suburban
- major local road urban
- major road dense urban
- major road rural
- major road suburban
- major road urban
- minor local road dense Urban
- minor local road rural
- minor local road suburban
- minor local road urban
- mormal road dense urban
- normal road rural
- normal road rural
- normal road urban
- primary highway dense urban
- primary highway rural
- primary highway suburban
- primary highway urban
- ramp dense urban
- ramp limited access

- ramp major road
- ramp primary highway
- ramp rural
- ramp secondary highway
- ramp urban
- ramp suburban
- secondary highway dense urban
- secondary highway rural
- secondary highway suburban
- secondary highway urban

Download Permissions

Setting the download permissions allows multiple services to download and update the downloaded data when required. You should have a common operating system group of which all the service users who need to download the data are part of. For example, if Hive and YARN jobs are required to download data and use the same download location, then both the Hive and YARN operating system users should be part of a common operating system group. The group of the download directory should be the common operating system group, one that has Read, Write, and Execute (775) permissions for the owner and group.

Your group should contain services and users that will run jobs in your cluster. You may skip services you will not use or do not have installed. Services include YARN, Hive, Zeppelin, and Hue.

You also should include all operating system users who will run jobs such as pbuser and *<myOtherUser>*.

1. Add the group.

```
sudo groupadd pbdownloads
```

2. Add users to the group.

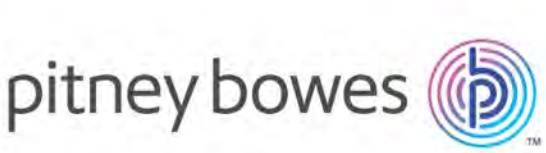
```
sudo usermod -a -G pbdownloads hive
sudo usermod -a -G pbdownloads yarn
sudo usermod -a -G pbdownloads zeppelin
sudo usermod -a -G pbdownloads hue
sudo usermod -a -G pbdownloads pbuser
sudo usermod -a -G pbdownloads <myOtherUser>
```

3. Using a window where no job is running, restart all the services whose operating system users were added to the new group.
4. Using a window where no job is running, restart the session of all the operating system users that were added to new group (for example, pbuser).

5. Update the group to the common operating system group and update permissions to 775 for the download directory specified in `pb.download.location` property.

```
sudo chgrp pbdownloads /pb/downloads
sudo chmod 775 /pb/downloads
```

Copyright



3001 Summer Street
Stamford CT 06926-0700
USA

www.pitneybowes.com