

IBM DB2 Spatial Reader/Writer

FORMAT NOTES:

- This format is not supported by FME Base Edition.
- Object writing is available only with FME DB2 Edition.
- DB2 7.2 and 8.1 are currently supported.
- DB2 Spatial currently provides support for 2D geometries only. For 3D and 3D with Measures support, please contact Safe Software.

FME's DB2 Spatial Reader/Writer module (referred to as *DB2 Spatial* in this chapter) provides the Feature Manipulation Engine (FME) with the ability to read spatial and attribute data from IBM's DB2 database, and write spatial and attribute data to the existing database.

Overview

DB2 Spatial can read from databases which are spatially enabled but the tables may or may not have spatial information stored. This module communicates directly with DB2 using CLI for maximum throughput.

Note: DB2 7.2 and 8.1 are currently supported. DB2 Spatial currently provides support for 2D geometries only. For 3D and 3D with Measures support, please contact Safe Software.

This section assumes familiarity with IBM DB2 Spatial Extender, the geometry types it supports, and its indexing mechanisms.

Tip:

- See the `QueryFactory` in the FME Functions and Factories manual. This factory also exploits the powerful query capabilities of DB2 Spatial.
 - See the `@SQL` function, also in the FME Functions and Factories manual. This function allows arbitrary Structured Query Language (SQL) statements to be executed against any DB2 database.
-

DB2 Spatial Quick Facts

Format Type Identifier	DB2
Reader/Writer	Both
Licensing Level	Reading: Professional Object Writing: DB2 Edition
Dependencies	None
Dataset Type	Data source name
Feature Type	Table name
Typical File Extensions	N/A
Automated Translation Support	Yes
User-Defined Attributes	Yes
Coordinate System Support	Yes
Generic Color Support	No
Spatial Index	Never
Schema Required	Yes
Transaction Support	Yes
Geometry Type	db2_type

Geometry Support			
Geometry	Supported?	Geometry	Supported?
aggregate	yes	point	yes
circles	no	polygon	yes
circular arc	no	raster	no
donut polygon	yes	solid	no
elliptical arc	no	surface	no
ellipses	no	text	no
line	yes	z values	n/a
none	yes		

Reader Overview

The FME considers a DB2 Spatial dataset to be a database containing a collection of relational tables together with their geometry. The tables to be read may be defined in the mapping file. If no tables are specified, then all tables are read. Arbitrary `WHERE` clauses and joins are fully supported. An entire arbitrary SQL `SELECT` statement may also be used as a source of results.

Reader Directives

The suffixes listed are prefixed by the current `<ReaderKeyword>` in a mapping file. By default, the `<ReaderKeyword>` for the DB2 Spatial reader is `DB2SPATIAL`.

DATASET**Required/Optional:** *Required*

This specifies the data source name for the DB2 Spatial database. The data source name must have been set up in the Client Configuration Assistant or on the command line.

Example:

```
DB2SPATIAL_DATASET citySource
```

Workbench Parameter: [<WorkbenchParameter>](#)

USER_NAME**Required/Optional:** *Required*

The name of the user who will access the database.

Example:

```
DB2SPATIAL_USER_NAME shadow
```

Workbench Parameter: [<WorkbenchParameter>](#)

PASSWORD**Required/Optional:** *Required*

The password to access the database.

Example:

```
DB2SPATIAL_PASSWORD puppy
```

Workbench Parameter: [<WorkbenchParameter>](#)

DEF**Required/Optional:** *Optional*

The syntax of the definition is:

```
DB2SPATIAL_DEF <tableName>                                     \
  [db2_type           <type>]                                 \
  [db2_envelope_minx   <xmin>]                                 \
  [db2_envelope_miny   <ymin>]                                 \
  [db2_envelope_maxx   <xmin>]                                 \
  [db2_envelope_maxy   <ymin>]                                 \
  [db2_spatial_predicate <spatialPredicate>]                \
  [db2_predicate_result <predicateResult>]                   \
  [db2_where_clause <whereClause>]                             \
  [db2_sql           <sqlQuery>]                               \
  [<fieldName> <fieldType>] +
```

The <fieldType> of each field must be given, but it is not verified against the database definition for the field. In effect, it is ignored.

The column(s) which has geometry should not be specified on the DEF line. In case a feature type has more than one registered geometry column or layer, than DB2 Spatial Reader module will arbitrarily choose one as the primary geometry column and consider the other(s) as attribute columns.

The <tableName> can be either fully qualified or not. A fully qualified table name consist of two parts separated by a period (.). The first part is the <schema name> and second part is the <table name>. The <table name> part must match a table in the schema specified by the <schema name> part of the <tableName>. If a schema name is not provided as part of the table name, then the username will be considered the schema name. This will be used as the feature type of all the features read from the table. For example, if a user wants to read a table from its own schema then only the table name can be provided, but if the user wants to read from a different user's schema, then table name should be qualified with schema name.

The definition allows specification of separate search parameters for each table. If any of the configuration parameters are given, they will override, for that table, whatever global values have been specified by the reader directives listed above. If any of these parameters is not specified, the global values will be used.

The following table summarizes the definition line configuration parameters:

Parameter	Contents
db2_type	This specifies the type of geometry the features to be read from the layer will have.
db2_geometry_column	This specifies the spatial layer or geometry column to use for reading spatial data in case the table has multiple geometry/spatial columns.
db2_envelope_minx db2_envelope_miny db2_envelope_maxx db2_envelope_maxy	These specify the spatial extent of the features to be read from the layer. If these are not all specified, the values from the <ReaderKeyword>_SEARCH_ENVELOPE directive are used.
db2_spatial_predicate	This specifies the spatial predicate to be tested for this layer. Its default value is set to INTERSECTS. Note: This DEF line option is valid only if there is a valid spatial envelope specified by db2_envelope_minx, db2_envelope_miny, db2_envelope_maxx and db2_envelope_maxy.
db2_predicate_result	This specifies the result to be used for the Spatial predicate specified in db2_spatial_predicate option.
db2_where_clause	This specifies the SQL WHERE clause applied to the attributes of the layer's features to limit the set of features returned. If this is not specified, the value of the <ReaderKeyword>_WHERE_CLAUSE directive is used.

Parameter	Contents
db2_sql	<p>This specifies an SQL SELECT query to be used as the source for the results. If this is specified, the DB2 Spatial reader will execute the query, and use the resulting rows as the features instead of reading from the table <layerName>. All returned features will have a feature type of <layerName>, and attributes for all columns selected by the query.</p> <p>The db2_where_clause and all parameters which specify a spatial constraint – db2_envelope_minx, db2_interaction, and so on – are ignored if db2_sql is supplied.</p>

If no <whereClause> is specified, all rows in the table will be read and returned as individual features. If a <whereClause> is specified, only those rows that are selected by the clause will be read. Note that the <whereClause> does not include the word “where”.

The db2_sql parameter allows a user to specify an arbitrary SQL SELECT query. If this is specified, FME will execute the query, and use each row of data returned from the query to define a feature. Each of these features will be given the feature type named in the DEF line, and will contain attributes for every column returned by the SELECT. In this case, all DEF line parameters regarding a WHERE clause or spatial querying is ignored, as it is possible to embed this information directly in the text of the <sqlQuery>.

The following example joins the tables ROADS and ROADNAMES, placing the resulting data into FME features with a feature type of MYROADS. Imagine that ROADS defines the geometry for the roads, and has a numeric field named ID, and that ROADNAMES joins the numeric field ID with character arrays with the roads’ names.

```
DB2SPATIAL_DEF MYROADS                                     \
  db2_sql "SELECT * FROM ROADS,                            \
          ROADNAMES WHERE ROADS.ID = ROADNAMES.ID"
```

Workbench Parameter: [<WorkbenchParameter>](#)

IDs

Required/Optional: *Optional*

This optional specification is used to limit the available and defined database tables files that will be read. If no IDs are specified, then all defined and available tables are read. The syntax of the IDs directive is:

```
DB2SPATIAL_IDS <featureType1>                             \
  <featureType2>                                          \
  <featureTypeN>
```

The feature types must match those used in DEF lines.

The example below selects only the ROADS table for input during a translation:

```
DB2SPATIAL_IDS ROADS
```

Workbench Parameter: [<WorkbenchParameter>](#)

SIMPLIFY_AGGREGATES**Required/Optional:** *Optional*

This directive specifies whether multi-geometry or aggregate features with one member are read as stored or simplified and read as single member. e.g. an aggregate of points or multipoint features with only one point will be returned as a simple point if the value of this directive is YES.

Values: YES | NO**Default value:** NO**Example:**

The syntax of the **DB2SPATIAL_SIMPLIFY_AGGREGATES** directive is:

```
DB2SPATIAL_SIMPLIFY_AGGREGATES YES
```

Workbench Parameter: [<WorkbenchParameter>](#)

SEARCH_ENVELOPE**Required/Optional:** *Optional*

This directive specifies the spatial extent of the feature retrieval. Only features that have relationships specified by **INTERACTION** with the area defined by the bounding box are returned. If this is not supplied, all features will be returned.

Example:

The syntax of the **DB2SPATIAL_SEARCH_ENVELOPE** directive is:

```
DB2SPATIAL_SEARCH_ENVELOPE <minX> <minY> <maxX> <maxY>
```

The example below selects a small area for extraction:

```
DB2SPATIAL_SEARCH_ENVELOPE -130 49 -128 50.1
```

Workbench Parameter: [<WorkbenchParameter>](#)

SEARCH_ENVELOPE_COORDINATE_SYSTEM**Required/Optional:** *Optional*

This directive specifies the coordinate system of the search envelope if it is different than the coordinate system of the data. The **COORDINATE_SYSTEM** directive, which specifies the coordinate system associated with the data to be read, must always be set if the **SEARCH_ENVELOPE_COORDINATE_SYSTEM** directive is set.

If this directive is set, the minimum and maximum points of the search envelope are reprojected from the **SEARCH_ENVELOPE_COORDINATE_SYSTEM** to the reader **COORDINATE_SYSTEM** prior to applying the envelope.

The syntax of the **SEARCH_ENVELOPE_COORDINATE_SYSTEM** directive is:

```
<ReaderKeyword>_SEARCH_ENVELOPE_COORDINATE_SYSTEM <coordinate system>
```

Workbench Parameter: [Search Envelope Coordinate System](#)

SPATIAL_PREDICATE**Required/Optional:** *Optional*

This specifies the type of spatial relationship which must exist between the search envelope and the geometry in the target layer. Any supported relationship, in combination with the `SPATIAL_PREDICATE_RESULT` directive, can be used to filter the features being read.

Values: *CONTAINS, CROSSES, DISJOINT, EQUALS, INTERSECTS, ORDERINGEQUALS, OVERLAPS, TOUCHES, WITHIN*

Default value: *INTERSECTS*

For example,

```
DB2SPATIAL_SPATIAL_PREDICATE INTERSECTS
DB2SPATIAL_SPATIAL_PREDICATE_RESULT FALSE
```

This would result in a spatial filter using DB2 Spatial's native spatial function

```
DB2GSE.ST_Intersects( g1 geometry, g2 geometry) = 0
```

where `g1` is the search envelope and `g2` is the target feature. This will cause FME to return only those features that satisfy the spatial predicate above.

The following table lists the valid spatial predicate relationships.

Search Method	Description
CONTAINS	Determines whether the search envelope is completely contained by the target feature.
CROSSES	Determines whether the intersection of search envelope and the target feature results in a geometry object whose dimension is one less than the maximum dimension of the source geometries. Also determines if the intersection object contains points that are interior to both source geometries and are not equal to either of the source objects.
DISJOINT	Determines whether the intersection of search envelope with the target feature is an empty set.
EQUALS	Determines whether the search envelope and target feature are of the same type and have identical x,y coordinate values.
INTERSECTS	Determines whether the intersection of search envelope and target feature does not result in an empty set. This is the exact opposite of DISJOINT.
ORDERINGEQUALS	Determines whether the search envelope and target feature are equal and the coordinates are in the same order.
OVERLAPS	Determines whether the search envelope and target feature overlap each other.

Search Method	Description
TOUCHES	Determines whether any of the points common to search envelope and target feature intersect the interiors of both geometries. At least one geometry must be a linestring, polygon, multilinestring, multipolygon.
WITHIN	Determines whether the target feature is completely within the search envelope. This is exactly opposite to CONTAINS .

For more details on Spatial predicate, please refer to the *IBM DB2 Spatial Extender User's Guide and Reference*.

CLIP

Required/Optional: *Optional*

This directive specifies whether or not to clip the returned features with respect to Search envelope.

Values: YES | NO

Default value: NO

Workbench Parameter: [<WorkbenchParameter>](#)

WHERECLAUSE

Required/Optional: *Optional*

This specifies an SQL WHERE clause, which is applied to the table's columns to limit the resulting features. This feature is currently limited to apply only to the attributes of the target table, and does not allow for joining multiple tables together. The effect of table joins can be achieved using the object model, by specifying the entire queries in the DEF line with a db2_sql parameter.

By default, there is no WHERE clause applied to the results, so all features in the layer are returned.

Example:

```
DB2SPATIAL_WHERECLAUSE "se_row_id > 45"
```

Workbench Parameter: [<WorkbenchParameter>](#)

TRANSACTION_INTERVAL

Required/Optional: *Optional*

The features can be read from the DB2 Spatial database using a bulk reading technique to maximize performance. Normally 1000 rows of data are read from the database at a time. However, when we are reading LOB (BLOBs or CLOBs) data , we are restricted to a transaction interval of size 1. Since geometry columns are normally BLOB types, reading of spatial features will not be affected by this directive.

This directive allows users to tune the performance of the reader. It specifies how many rows are read from the database at a time.

Example:

```
DB2SPATIAL_TRANSACTION_INTERVAL "se_row_id > 45"
```

Workbench Parameter: [<WorkbenchParameter>](#)

BEGIN_SQL{n}

Required/Optional: *Optional*

Occasionally a user must execute some ad-hoc SQL statements prior to opening a DB2 Spatial table. For example, it may be necessary to ensure that a view exists prior to attempting to read from it.

Upon opening a connection to read from an DB2 Spatial database, the DB2 Spatial reader looks for the directive `<ReaderKeyword>_BEGIN_SQL{n}` (for $n=0, 1, 2, \dots$), and executes each such directive's value as an SQL statement on the database connection. Any errors occurring during the execution of these SQL statements will terminate the reader with an error.

Example:

```
DB2SPATIAL_BEGIN_SQL{0} "DELETE FROM myschema.mytable WHERE id=1"
DB2SPATIAL_BEGIN_SQL{1} "DELETE FROM myschema.yourtable WHERE id=1"
```

Workbench Parameter: [<WorkbenchParameter>](#)

END_SQL{n}

Required/Optional: *Optional*

Occasionally a user must execute some ad-hoc SQL after closing a set of DB2 Spatial tables. For example, it may be necessary to clean up a temporary view after writing to the database.

Just prior to closing a connection on an DB2 Spatial database, the DB2 Spatial reader looks for the directive `<ReaderKeyword>_END_SQL{n}` (for $n=0, 1, 2, \dots$), and executes each such directive's value as an SQL statement on the database connection. Any errors occurring during the execution of these SQL statements will terminate the reader with an error.

Example:

```
DB2SPATIAL_END_SQL{0} "DELETE FROM myschema.mytable WHERE id=1"
DB2SPATIAL_END_SQL{1} "DELETE FROM myschema.yourtable WHERE id=1"
```

Workbench Parameter: [<WorkbenchParameter>](#)

PERSISTENT_CONNECTION

A user may want to keep a connection to a database for reuse during a particular FME session. For example, when running a batch of 100 mapping files on the same database connection, it may be desirable to keep a connection open and save the processing time required to make and break a database connection.

A database connection will be determined to be the same when the database name, the username, the password, and the transaction interval are the same.

Values: YES | NO

Default value: NO

Example:

```
DB2SPATIAL_PERSISTENT_CONNECTION YES
```

Workbench Parameter: <WorkbenchParameter>

RETRIEVE_ALL_SCHEMAS

This specification is only applicable when generating a mapping file, generating a workspace or when retrieving schemas in a FME Objects application.

This optional specification is used to tell the reader to retrieve the names and the schemas of all the tables in the source database. If this value is not specified, it is assumed to be "No".

The syntax of the RETRIEVE_ALL_SCHEMAS directive is:

```
DB2SPATIAL_RETRIEVE_ALL_SCHEMAS Yes
```

RETRIEVE_ALL_TABLE_NAMES

This specification is only applicable when generating a mapping file, generating a workspace or when retrieving schemas in a FME Objects application.

Similar to RETRIEVE_ALL_SCHEMAS; this optional specification is used to tell the reader to only retrieve the table names of all the tables in the source database. If RETRIEVE_ALL_SCHEMAS is also set to "Yes," then RETRIEVE_ALL_SCHEMAS is chosen. If this value is not specified, it is assumed to be "No".

The syntax of the RETRIEVE_ALL_TABLE_NAMES directive is:

```
DB2SPATIAL_RETRIEVE_ALL_TABLE_NAMES Yes
```

Writer Overview

The DB2 Spatial writer module stores both geometry and attributes into DB2 Spatially enabled databases. The DB2 Spatial writer provides the following capabilities:

- **Table Creation:** Uses the information within the FME mapping file to automatically create database tables as needed.
- **Coordinate System:** Checks for a coordinate system and if a matching one is not found then it will create one automatically. The matching criteria is the OGC WKT definition of the coordinate system.

Note: When writing to DB2 V7.2, the exact coordinate system definition of the incoming features should exist in the database; otherwise FME will not be able to create a new coordinate system if the user does not have enough privileges to write to the DB2GSE.GSE_COORD_REF metadata table. User need to have either DBADB authority or at least have INSERT privileges for the coordinate system metadata table DB2GSE.GSE_COORD_REF for the translation to succeed.

- **Spatial Reference System:** Uses an existing Spatial reference system with matching parameters or create new one as required when registering spatial layer(column).

- **Spatial Grid Index Creation:** Creates spatial indexes only if valid values are specified for different levels of grid.
- **ESRI's ArcExplorer 3.0 JDBC Edition:** In order to view spatial data written by DB2 Spatial writer in ArcExplorer 3.0 each spatial table must have a column named "SE_ROW_ID" of type `integer`. It does not really matter whether the column is populated or not. Also note that ArcExplorer expects users to at least have execute privileges on certain functions in DB2GSE schema or have DBADM authority.

Note: Bulk Loading: The DB2 Spatial writer does not use a bulk loading technique due to certain limitations in DB2.

Writer Directives

The directives processed by the DB2 Spatial writer are listed below. The suffixes shown are prefixed by the current `<WriterKeyword>` in a mapping file. By default, the `<WriterKeyword>` for the DB2 Spatial writer is `DB2SPATIAL` when using the object model.

DATASET, USER_NAME, PASSWORD, PERSISTENT_CONNECTION, TRANSACTION_INTERVAL, BEGIN_SQL{ }, and END_SQL{ }

The `DATASET`, `USER_NAME`, `PASSWORD`, `PERSISTENT_CONNECTION`, `TRANSACTION_INTERVAL`, `BEGIN_SQL{ }`, and `END_SQL{ }` directives operate in the same manner as they do for the DB2 Spatial reader. The remaining writer-specific directives are discussed in the following sections.

DEF

Required/Optional: *Optional*

Each DB2 Spatial table must be defined before it can be written. The general form of a DB2 Spatial definition statement is:

```
DB2SPATIAL_DEF <tableName>                                     \
  [db2_overwrite_table <YES|NO|TRUNCATE>]                     \
  [db2_multi_geometry <YES|NO|FIRST_FEATURE>]                 \
  [db2_geometry_column <geometry>]                            \
  [db2_offset_x <x offset value>]                              \
  [db2_offset_y <y offset value>]                              \
  [db2_scale_x <x scale value>]                                \
  [db2_scale_y <y scale value>]                                \
  [db2_grid_0 <finest grid size>]                              \
  [db2_grid_1 <middle grid size>]                              \
  [db2_grid_2 <coarsest grid size>]                            \
  [db2_sql <sql statement>]                                    \
  [db2_update_key_columns <column>[,<column>]...]             \
  [db2_delete_key_columns <column>[,<column>]...]             \
  [<fieldName> <fieldType>]*
```

The table definition allows complete control of the layer that will be created. If the layer already exists, the majority of the `DEF` line parameters will be ignored and need not be given. As well, if the table already exists in the database, then it is not necessary to list the fields and their types – FME will use the schema information in the database to

determine this. FME will ignore the field names and types specified on the DEF line, except for the one with type `geometry`.

If the table does not exist, then the field names and types are used to first create the table. In any case, if a `<fieldType>` is given, it may be any field type supported by the target database.

The DB2 Spatial writer will use `db2_geometry_column` parameter to set the name of geometry column for the new table. If the `db2_geometry_column` parameter is not specified then a default name "geometr" will be used for the geometry column.

The configuration parameters present on the definition line are described in the following table:

Parameter	Contents
<code>db2_overwrite_table</code>	This parameter can have one of <code><YES NO TRUNCATE></code> option. If <code>YES</code> , then the table will be dropped and created again. If <code>TRUNCATE</code> , then all the rows from the table will be deleted. If <code>NO</code> , then data will be appended to the existing table.
<code>db2_multi_geometry</code>	This specifies whether the db2 types for point, linestring and polygon should be written as multi-geometries or single geometries. If <code>YES</code> , the table created has multi-geometries (that is, the geometry column type will be <code>ST_MULTIPPOINT</code> , and the features are coerced into multi-geometries if they are not already). If <code>NO</code> , the geometry column of the created table is singular (that is, <code>ST_POINT</code>), and multi-geometries are split. <code>FIRST_FEATURE</code> allows this setting to be based on the first feature in the table. This setting is used for DB2SPATIAL-to-DB2SPATIAL translations.
<code>db2_geometry_column</code>	This parameter can be used to specify name of the spatial layer (geometry column name). If it is not specified db2 spatial writer module will use default name "geometry" for the spatial layer.
<code>db2_offset_x</code>	The x offset value for the dataset, defaults to 0. If this parameter is non-zero, then it overrides the global <code>OFFSET_X</code> directive.
<code>db2_offset_y</code>	The y offset value for the dataset, defaults to 0. If this parameter is non-zero, then it overrides the global <code>OFFSET_Y</code> directive.
<code>db2_scale_x</code>	The x scale value for the dataset, defaults to 1. If this parameter is not equal to 1, then it overrides the global <code>SCALE_X</code> directive
<code>db2_scale_y</code>	The y scale value for the dataset, defaults to 1. If this parameter is not equal to 1, then it overrides the global <code>SCALE_Y</code> directive.
<code>db2_grid_0</code>	This parameter specifies the finest spatial index grid size. If 0, then a spatial index is not created.
<code>db2_grid_1</code>	This parameter specifies the middle spatial index grid size. If 0, then a spatial index is not created.
<code>db2_grid_2</code>	This parameter specifies the coarsest spatial index grid size. If 0, then a spatial index is not created.

Parameter	Contents
db2_sql	<p>This specifies an SQL <code>INSERT</code> or <code>UPDATE</code> query to be used to define the results. If this is specified, the DB2 Spatial writer will execute the query, defining one row for each feature from FME. The values in the query are specified by embedding “<code>?attrName</code>” in the query itself, where <code>attrName</code> is the name of the FME feature’s attribute. For example:</p> <pre>INSERT INTO MyTable VALUES(?ID,?NAME,?DESC)</pre> <p>In this example, the attributes named <code>ID</code>, <code>NAME</code> and <code>DESC</code> will be taken from each feature written to <code><tableName></code>.</p> <p>or</p> <pre>INSERT INTO MyTable (ID,NAME) VALUES(?ID,?NAME)</pre> <p>In this example, the attributes named <code>ID</code> and <code>NAME</code> will be taken from each feature written to <code><tableName></code>. If not all attributes are to be written, then a column list should be specified as shown in the second example statement where 2 out of 3 columns are being written.</p> <p>It is also very important that the attributes named in the query must be listed on the <code>DEF</code> line so that FME knows what type to use. There is no necessary or implied correlation between the FME attribute name and the db2 column name.</p>
db2_update_key_columns	<p>This instructs the DB2 Spatial writer to perform an <code>UPDATE</code> operation on the table, rather than performing an <code>INSERT</code>. The argument is a comma-separated list of the columns which are matched against the corresponding FME attributes’ values to specify which rows are to be updated with the other attribute values.</p> <p>For example:</p> <pre>db2_update_key_columns ID,NAME</pre> <p>In this case the FME attribute is always matched against the db2 column with the same name. Also, the target table is always the feature type specified in the <code>DEF</code> line. Each column listed with the <code>db2_update_key_columns</code> directive must be defined with a type on the <code>DEF</code> line, in addition to the columns whose values will be updated by the operation. This cannot be used with <code>db2_delete_key_columns</code>. Also, the keys cannot be of type <code>BLOB</code>, <code>CLOB</code>, or <code>LONG_VARCHAR</code>.</p>
db2_delete_key_columns	<p>This instructs the DB2 Spatial writer to perform a <code>DELETE</code> operation on the table, rather than performing an <code>INSERT</code>. The argument is a comma-separated list of the columns which are matched against the corresponding FME attributes’ values to specify which rows are to be deleted when their values match the other attribute values.</p> <p>For example:</p> <pre>db2_delete_key_columns ID,NAME</pre> <p>would delete those rows in the table whose values match the attribute values passed in through this <code>DEF</code> line. The FME attribute is always matched against the DB2 Spatial column with the same name. Also, the target table is always the feature type specified in the <code>DEF</code> line. Each column listed with the <code>db2_delete_key_columns</code> directive must be defined with a type on the <code>DEF</code> line, in addition to the columns whose values will be updated by the operation. This cannot be used with <code>db2_update_key_columns</code>. Also, the keys cannot be of type <code>BLOB</code>, <code>CLOB</code>, or <code>LONG_VARCHAR</code>.</p>

TRANSACTION_INTERVAL

This statement informs the FME about the number of features to be placed in each transaction before a transaction is committed to the database.

If the `DB2SPATIAL_TRANSACTION_INTERVAL` statement is not specified, then a value of 1000 is used as the transaction interval.

Parameter	Contents
<code><transaction_interval></code>	The number of features in a single transaction.

Default: *1000*

Example:

```
DB2SPATIAL_TRANSACTION_INTERVAL 2500
```

PERSISTENT_CONNECTION

Required/Optional: *Optional*

A user may want to keep a connection to a database for reuse during a particular FME session. For example, when running a batch of 100 mapping files on the same database connection, it may be desirable to keep a connection open and save the processing time required to make and break a database connection.

A database connection will be determined to be the same when the database name, the username, the password, and the transaction interval are the same.

Values: *YES | NO*

Default: *NO*

Example:

```
DB2SPATIAL_PERSISTENT_CONNECTION YES
```

Workbench Parameter: [<WorkbenchParameter>](#)

ABORT_ON_BAD_DATA

Required/Optional: *Optional*

Some features' geometries may fail DB2 Spatial Extender's check constraints based on the offset, scale, and coordinate system values. These features, as well as others with out-of-range or invalid attribute values, will be rejected and cannot be written to the database. If the value of this directive is YES then the translation will be aborted immediately after encountering such a problem. If this directive is set to NO then the translation will continue but the rejected features will not be written to the database.

Values: *YES | NO*

Default: *YES*

Example:

```
DB2SPATIAL_ABORT_ON_BAD_DATA YES
```

Workbench Parameter: [<WorkbenchParameter>](#)

OFFSET_X

Required/Optional: *Optional*

This directive can be used to set the global *x* offset for the entire translation. If a dataset contains many different tables but the same *x* offset applies to all of them, then this is a convenient way of setting the *x* offset. This value can be overridden by *DEF* line parameter *db2_offset_x*.

Default: *0*

Example:

```
DB2SPATIAL_OFFSET_X -12456
```

Workbench Parameter: [<WorkbenchParameter>](#)

OFFSET_Y

Required/Optional: *Optional*

This directive can be used to set the global *y* offset for the entire translation. If a dataset contains many different tables but the same *y* offset applies to all of them, then this is a convenient way of setting the *y* offset. This value can be overridden by *DEF* line parameter *db2_offset_y*.

Default: *0*

Example:

```
DB2SPATIAL_OFFSET_Y -1245
```

Workbench Parameter: [<WorkbenchParameter>](#)

SCALE_X

Required/Optional: *Optional*

This directive can be used to set the global *x* scale value for the entire translation. If a dataset may contains many different tables but the same *x* scale applies to all of them, then this is a convenient way of setting the *x* scale value. This value can be overridden by *DEF* line parameter *db2_scale_x*.

Default: *1*

Example:

```
DB2SPATIAL_SCALE_X 1000
```

Workbench Parameter: [<WorkbenchParameter>](#)

SCALE_Y

Required/Optional: *Optional*

This directive can be used to set the global y scale value for the entire translation. If a dataset contain many different tables but the same y scale applies to all of them, then this is a convenient way of setting the y scale value. This value can be overridden by DEF line parameter `db2_scale_y`.

Default: *1*

Example:

```
DB2SPATIAL_SCALE_Y 1000
```

Workbench Parameter: [<WorkbenchParameter>](#)

GRID_0

Required/Optional: *Optional*

This directive can be used to set the global finest grid size for the spatial grid index . If a dataset contains many different tables but the same finest grid size applies to all of them, then this is a convenient way of setting the finest grid size value. This value can be overridden by DEF line parameter `db2_grid_0`.

Default: *0*

Example:

```
DB2SPATIAL_GRID_0 10
```

Workbench Parameter: [<WorkbenchParameter>](#)

GRID_1

Required/Optional: *Optional*

This directive can be used to set the global middle grid size for the spatial grid index. If a dataset contains many different tables but the same middle grid size applies to all of them, then this is a convenient way of setting the middle grid size value. This value can be overridden by DEF line parameter `db2_grid_1`.

Default: *0*

Example:

```
DB2SPATIAL_GRID_1 100
```

Workbench Parameter: [<WorkbenchParameter>](#)

GRID_2

Required/Optional: *Optional*

This directive can be used to set the global coarsest grid size for the spatial grid index. If a dataset contains many different tables but the same coarsest grid size applies to all of them, then this is a convenient way of setting the coarsest grid size value. This value can be overridden by DEF line parameter `db2_grid_2`.

Default: 0

Example:

```
DB2SPATIAL_GRID_2 1000
```

Workbench Parameter: [<WorkbenchParameter>](#)

Feature Representation

Features read from DB2 Spatial consist of a series of attribute values and geometry. The feature type of each Database feature is as defined on its `DEF` line.

Features written to the database have the destination table as their feature type, and attributes as defined by on the `DEF` line.

In addition to the generic FME feature attributes that FME Workbench adds to all features (see *About Feature Attributes* on page 7), the DB2 Spatial module adds the format-specific attributes described below:

Attribute Name	Contents
db2_type	The type of geometric entity stored within the feature. The valid values for the object model are listed below: db2_nil db2_point db2_linestring db2_polygon

Features read from, or written to, DB2 Spatial also have an attribute for each column in the database table. The feature attribute name will be the same as the source or destination column name. The attribute and column names are not case-sensitive.

No Coordinates

db2_type: db2_nil

Features with no coordinates are tagged with this value when reading or writing to or from db2 Spatial.

Points

db2_type: db2_point

All DB2 Spatial point and multipoint features are read as `db2_point`. The only difference being the geometry type of feature, which will be set to `fme_aggregate` if it is a multipoint and `fme_point` if it is a point.

Lines

db2_type: db2_line

All DB2 Spatial linestring and Multilinestring features are read as `db2_line`. The only difference is the geometry type of feature, which will be set to `fme_aggregate` if it is a multilinestring and `fme_line` if it is a linestring.

Polygons

db2_type: db2_polygon

All DB2 Spatial polygon and Multipolygon features are read as `db2_polygon`. The only difference is the geometry type of feature, which will be set to `fme_aggregate` if it is a multipolygon and `fme_polygon` if it is a polygon. Polygon features include donut polygons with one or more holes.

Aggregates are written out as “multipolygon” geometry containing several polygonal elements, just as if the feature had been tagged with `db2_multiline`. Any non-polygonal elements contained in the aggregate are discarded.

The following table summarizes all of the `db2_type` values that are possible with DB2 Spatial geometry, and provides a description of each representation.

db2_type	DB2 Spatial type	Representation
db2_nil	N/A	No geometry
db2_point	POINT	Single point geometry. fme_geometry = fme_point fme_type = fme_point
	MULTIPOINT	Aggregate containing one or more points. fme_geometry = fme_aggregate fme_type = fme_point
db2_line	LINestring	Single line geometry. fme_geometry = fme_line fme_type = fme_line
	MULTILINestring	An aggregate of linestrings. fme_geometry = fme_aggregate fme_type = fme_line
db2_polygon	POLYGON	A single polygon or donut geometry. fme_geometry = fme_polygon or fme_donut fme_type = fme_polygon
	MULTIPOLYGON	An aggregate of simple polygons or donut polygons. fme_geometry = fme_aggregate fme_type = fme_polygon

Troubleshooting

Problems sometimes arise when attempting to connect to an DB2 Spatial database. This is almost always due to a misconfiguration in the user's environment. The following suggestions can often help detect and overcome such problems.

- Ensure you can connect to the database with the data source name, username, and password using DB2 Command Line processor.
- Ensure that you have the correct version of the DB2 client software installed.
- Ensure that the appropriate version of DB2 Spatial Extender is installed and the database is 'Spatially enabled'. If you get an error which says something like "DB2GSE.*.. is an undefined name", then it is most likely that the database is not enabled for spatial operations. For enabling a DB2 database for spatial operations, please refer to *IBM DB2 Spatial Extender User's Guide and Reference*.
- Ensure that you have the appropriate privileges to perform the operations like creating, dropping, inserting into, and deleting from tables when writing. DBADM privileges may be required to create indexes. Please check the DB2 database manuals for more information.
- When reading/writing large volumes of data, please ensure that the database configuration parameters are set for large data processing. For example, when reading/writing large volumes of data, failure may occur due to "app_ctl_heap_sz" and/or "logprimary" parameters not set to appropriate values. Most database errors will be logged as obtained from the database. Some error messages may not immediately imply the actual problem. For such messages, please refer to DB2 database manuals.
- If offset and scale values are not chosen appropriately for the dataset, some or all geometries may be rejected. Error messages may not indicate the actual problem. For example, not choosing an appropriate scale may result in duplicate coordinates and the error message may be "not enough points" or "polygon intersects itself". Please refer to the *IBM DB2 Spatial Extender User's Guide and Reference* manual for the resolution of such errors.

