Publish Job Configuration Tables

Database Tables required for Jobs

GO Publisher supports four types of Data Selection which can be used standalone or in combination with two types of Data Chunking

Data Selection

1. Geographic Selection (using Extent Geometry or Extent ID*)
2. Feature ID Selection*
3. Attribute Selection
4. Temporal Selection

Data Chunking

1. Geographic Chunking*
2. Attribute Chunking

* If you require **Geographic Selection (using Extent ID)**, **Feature ID Selection** or **Geographic Chunking** then you will need to create the appropriate Configuration tables in THE SOURCE DATABASE. The source database is the database holding the source data you wish to publish.

If you require Attribute Selection, Temporal Selection and / or Attribute Chunking then you do not need to create any Configuration tables.

The easiest place to create Coniguration tables is alongside your source data tables in the same schema. If IT policy permits this then Configuration tables can be created in a separate schema.

Configuration tables in a separate schema must be in the same database and must also have the appropriate permissions to be queried across the two database schemas. If you decide to store the Configuration tables in a separate schema you will need to configure your product's Datasource to connect to the separate schema (see Configure products).

In order to execute any of the jobs outlined above you need to manually create the appropriate Configuration table(s) below.

GO Publisher Workflow supports the ASCII character-encoding scheme. Publish Job Configuration Tables can contain ASCII characters only.

Data Selection Jobs

The following tables are required to perform Data Selection jobs.

Geographic Selection

Geographic selection requires the GP_GEOGRAPHIC_SELECTION table to exist in the source database.

Table Name	GP_GEOGRAPHIC_SELECTION
Target Database	THE SOURCE DATABASE
Creation	Manual
Reason	The GP_GEOGRAPHIC_SELECTION table stores the geometries that can be used to query the database using geographic selection.

Creation SQL	**Example SQL for Oracle**
	`----------`
	`-- Delete any existing record of the table from the user_sdo_geom_metadata table`
	`----------`
	`DELETE FROM USER_SDO_GEOM_METADATA where TABLE_NAME = 'GP_GEOGRAPHIC_SELECTION';`
	
	`----------`
	`-- Create the GP_GEOGRAPHIC_SELECTION table`
	`----------`
	`CREATE TABLE GP_GEOGRAPHIC_SELECTION(ID varchar(50), EXTENT MDSYS.SDO_GEOMETRY);`
	
	`----------`
	`-- Insert into user_sdo_geom_metadata table with the relevant SRS and appropriate extents for the SRS`
	`----------`
	`INSERT INTO user_sdo_geom_metadata VALUES('GP_GEOGRAPHIC_SELECTION', 'EXTENT', MDSYS.SDO_DIM_ARRAY (MDSYS.SDO_DIM_ELEMENT('X',-10000000,10000000,0.0010), MDSYS.SDO_DIM_ELEMENT('Y', -10000000,10000000,0.0010)),2157);`
	
	Example SQL for PostgreSQL
	`----------`
	`-- Create the GP_GEOGRAPHIC_SELECTION table`
	`----------`
	`CREATE TABLE schemaname."GP_GEOGRAPHIC_SELECTION" (`
	`  "ID" varchar(50),`
	`  "EXTENT" geometry(Geometry,-1)`
	`);`

Features can then be added to the table to allow for geographic selection:

Example SQL for Oracle

```
----------
-- Insert records into the GP_GEOGRAPHIC_SELECTION table
----------
INSERT INTO GP_GEOGRAPHIC_SELECTION values ('QueryGeom1',SDO_GEOMETRY(2003,2157,NULL,SDO_ELEM_INFO_ARRAY
(1,1003,1),SDO_ORDINATE_ARRAY(605251,789746,605451,789746,605451,789946,605251,789946,605251,789746)));
```

Example SQL for PostgreSQL

```
----------
-- Insert records into the GP_GEOGRAPHIC_SELECTION table
----------
INSERT INTO schemaname."GP_GEOGRAPHIC_SELECTION" VALUES ('Treasure2',ST_GeomFromText('POLYGON((36878
22912,36878 23578,37369 23578,37369 22912,36878 22912))', -1));
```

This SQL is provided as an example. The values should be edited depending on your own data.

Feature ID Selection

Feature selection requires the GP_FEATURE_SELECTION table to exist in the source database.

Table Name	GP_FEATURE_SELECTION
Target Database	THE SOURCE DATABASE
Creation	Manual

<table>
<tr><td>Reason</td><td>The GP_FEATURE_SELECTION table stores 'key => feature' mappings that can be used to query the database using a feature selection key.</td></tr>
<tr><td>Example</td><td>
<table>
<tr><th></th><th>SELECTION_ID</th><th>FEATURE_ID</th><th>FEATURE_TABLE_NAME</th><th>FEATURE_KEY_COLUMN_NAME</th></tr>
<tr><td>Column definition</td><td>selection ID (used in the publish job)</td><td>feature(s) to be included in the selection ID</td><td>table that contains the feature (s)</td><td>column that contains the FEATURE_ID in the FEATURE_TABLE_NAME</td></tr>
<tr><td>Example</td><td>airport</td><td>3535</td><td>TABLE1</td><td>airport_id</td></tr>
<tr><td></td><td>airport</td><td>8536</td><td>TABLE1</td><td>airport_id</td></tr>
</table>
In this example, features with an airport_id of 3535 and 8536 in TABLE1 are grouped together with a SELECTION_ID 'airport.'
</td></tr>
<tr><td>Creation SQL</td><td>
Example SQL for Oracle
<pre>

-- DDL for Table GP_FEATURE_SELECTION

CREATE TABLE "GP_FEATURE_SELECTION" (
 "SELECTION_ID" VARCHAR2(50) NOT NULL,
 "FEATURE_ID" VARCHAR2(50) NOT NULL,
 "FEATURE_TABLE_NAME" VARCHAR2(30) NOT NULL,
 "FEATURE_KEY_COLUMN_NAME" VARCHAR2(30) NOT NULL
);

COMMIT;
</pre>
Example SQL for PostgreSQL
<pre>
CREATE TABLE schemaname."GP_FEATURE_SELECTION"
(
"SELECTION_ID" varchar(50),
"FEATURE_ID" varchar(50),
"FEATURE_TABLE_NAME" varchar(30),
"FEATURE_KEY_COLUMN_NAME" varchar(30)
);
</pre>
</td></tr>
</table>

Features can then be added to the table to allow for feature ID selection:

Example SQL for Oracle

```
-----------------------------
-- Insert records into the GP_FEATURE_SELECTION table
-----------------------------
INSERT INTO gp_feature_selection (selection_id, feature_id, feature_table_name, feature_key_column_name) VALUES
('airport',3535,'TABLE1','airport_id');
```

Example SQL for PostgreSQL

```
INSERT INTO schemaname."GP_FEATURE_SELECTION" ("SELECTION_ID","FEATURE_ID","FEATURE_TABLE_NAME","
FEATURE_KEY_COLUMN_NAME") VALUES ('airport',3535,'TABLE1','airport_id');
```

This SQL is provided as an example. The values should be edited depending on your own data.

The values are case sensitive.

For example, if the table name is capitalised in your database (such as TABLE1), it will need to be capitalised in the SQL.

Data Chunking Jobs

The following tables are required to perform Data Chunking jobs.

Geographic Chunking

For geographic chunking two tables are required:

- GP_CHUNK_SCHEME
- GP_CHUNKS

Table Name	GP_CHUNK_SCHEME
Target Database	THE SOURCE DATABASE
Creation	Manual
Reason	The GP_CHUNK_SCHEME table stores the names of chunking schemes available to GP-Workflow.

Creation SQL	**Example SQL for Oracle**
	`----------`
	`--  DDL for Table GP_CHUNK_SCHEME table under GPWORKFLOW`
	`----------`
	`CREATE TABLE "GP_CHUNK_SCHEME" (`
	`  "NAME" VARCHAR2(50 BYTE)`
	`);`
	
	`COMMIT;`
	
	⚠ **Note for Oracle SQL**
	You will need to create primary keys
	
	Example SQL for PostgreSQL
	`----------`
	`--  DDL for Table GP_CHUNK_SCHEME table`
	`----------`
	`CREATE TABLE schemaname."GP_CHUNK_SCHEME" (`
	`  "NAME" character varying(50)`
	`);`
	`----------`
	`--  DDL for Index PK_GP_CHUNK_SCHEME`
	`----------`
	`CREATE UNIQUE INDEX "PK_GP_CHUNK_SCHEME" ON schemaname."GP_CHUNK_SCHEME" ("NAME");`

Features can then be added to the table to define the chunking scheme name(s).

Example SQL for Oracle

```
----------------------------
-- Insert records into the GP_CHUNK_SCHEME table
----------------------------
INSERT INTO gp_chunk_scheme (name) VALUES ('large');
```

Example SQL for PostgreSQL

```
-----------------------------
-- Insert records into the GP_CHUNK_SCHEME table
-----------------------------
INSERT INTO schemaname."GP_CHUNK_SCHEME" ("NAME") VALUES ('large');
```

This SQL is provided as an example. The values should be edited depending on your own data.

Table Name	GP_CHUNKS
Target Database	THE SOURCE DATABASE
Creation	Manual
Reason	The GP_CHUNKS table stores '*key* => *chunk*' mappings that can be used to chunk the output of a job.

Creation SQL

Example SQL for Oracle

```
-------------------------------------------------------
--  DDL for Table GP_CHUNKS table under GPWORKFLOW
-------------------------------------------------------
CREATE TABLE "GP_CHUNKS" (
  "SCHEME" VARCHAR2(50 BYTE),
  "CHUNK_ID" VARCHAR2(50 BYTE),
  "PATH" VARCHAR2(50 BYTE),
  "EXTENT" "MDSYS"."SDO_GEOMETRY"
);

-------------------------------------------------------
--  DDL for Index PK_GP_CHUNK
-------------------------------------------------------
CREATE UNIQUE INDEX "PK_GP_CHUNK" ON "GP_CHUNKS" ("SCHEME", "CHUNK_ID");

-------------------------------------------------------
--  Constraints for Table GP_CHUNKS
-------------------------------------------------------
ALTER TABLE "GP_CHUNKS" ADD CONSTRAINT "PK_GP_CHUNK" PRIMARY KEY ("SCHEME", "CHUNK_ID") ENABLE;
CREATE INDEX "PKS_GP_CHUNK" ON "GP_CHUNKS" ("SCHEME");

COMMIT;
```

Example SQL for PostgreSQL

```
-------------------------------------------------------
--  DDL for Table GP_CHUNKS table
-------------------------------------------------------
CREATE TABLE schemaname."GP_CHUNKS" (
  "SCHEME" character varying(50),
  "CHUNK_ID" character varying(50),
  "PATH" character varying(50),
  "EXTENT" geometry(Geometry,-1)
);

-------------------------------------------------------
--  DDL for Index PK_GP_CHUNK
-------------------------------------------------------
CREATE UNIQUE INDEX "PK_GP_CHUNK" ON schemaname."GP_CHUNKS" ("SCHEME", "CHUNK_ID");

-------------------------------------------------------
--  Constraints for Table GP_CHUNKS
-------------------------------------------------------
CREATE INDEX "PKS_GP_CHUNK" ON schemaname."GP_CHUNKS" ("SCHEME");
```

Features can then be added to the table to allow for geographic chunking:

Example SQL for Oracle

```
-----------------------------
-- Insert records into the GP_CHUNKS table
-----------------------------
Insert into GP_CHUNKS (SCHEME,PATH,CHUNK_ID,EXTENT) values ('large','N','NE',MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY
(35000,24600,37000,24600,37000,26600,35000,26600,35000,24600)));
Insert into GP_CHUNKS (SCHEME,PATH,CHUNK_ID,EXTENT) values ('large','N','NW',MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY
(37000,24600,39000,24600,39000,26600,37000,26600,37000,24600)));
Insert into GP_CHUNKS (SCHEME,PATH,CHUNK_ID,EXTENT) values ('large','S','SE',MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY
(35000,22600,37000,22600,37000,24600,35000,24600,35000,22600)));
Insert into GP_CHUNKS (SCHEME,PATH,CHUNK_ID,EXTENT) values ('large','S','SW',MDSYS.SDO_GEOMETRY(2003,NULL,NULL,
MDSYS.SDO_ELEM_INFO_ARRAY(1,1003,1),MDSYS.SDO_ORDINATE_ARRAY
(37000,22600,39000,22600,39000,24600,37000,24600,37000,22600)));
```

This SQL is provided as an example based on the Treasure Island data and training we offer. The values should be edited depending on your own data.

Example SQL for PostgreSQL

```
-----------------------------
-- Insert records into the GP_CHUNKS table
-----------------------------
INSERT INTO schemaname."GP_CHUNKS" ("SCHEME","PATH","CHUNK_ID","EXTENT") VALUES
        ('large','N','NE',ST_GeomFromText('POLYGON((35000 24600,37000 24600,37000 26600,35000 26600,35000
24600))', -1));
INSERT INTO schemaname."GP_CHUNKS" ("SCHEME","PATH","CHUNK_ID","EXTENT") VALUES
        ('large','N','NW',ST_GeomFromText('POLYGON((37000 24600,39000 24600,39000 26600,37000 26600,37000
24600))', -1));
INSERT INTO schemaname."GP_CHUNKS" ("SCHEME","PATH","CHUNK_ID","EXTENT") VALUES
        ('large','S','SE',ST_GeomFromText('POLYGON((35000 22600,37000 22600,37000 24600,35000 24600,35000
22600))', -1));
INSERT INTO schemaname."GP_CHUNKS" ("SCHEME","PATH","CHUNK_ID","EXTENT") VALUES
        ('large','S','SW',ST_GeomFromText('POLYGON((37000 22600,39000 22600,39000 24600,37000 24600,37000
22600))', -1));
```

This SQL is provided as an example and the values should be edited depending on your own data.

Further Reading

Now you have set up the Configurable Tables needed to perform certain publishing job types read about how to Create a Publishing Job.