**Spatial Data Provider:**

**Functions and Flag Mappings**

Version: PreRelesase

Last Revised: March 29, 2016

Author: Patrick Wingo

The Spatial Data Provider (SDP) interface uses a series of flags to inform the parent environment of a given SDP implementation's capabilities. Descriptions of the flags and their associated functions are described below.

OLE-object

**Table 1.** .

Some functions are tied to the specific “grid mode” assigned to the Spatial Data Provider instance. This functions make use of the gridmode flags, as outlined in table 2. The optional grid modes are as follows:

**grid4 –** A square grid is represented, with only the edge neighbors considered (N, E, S, W).

**grid8 –** A square grid is represented, with edge and corner neighbors considered (N, NE, E, SE, S, SW, W, NW).

**hex –** A hexagon grid is represented with one of two neighbor orientations: either (N, NE, SE, S, SW, NW), or (NE, E, SE, SW, W, NW)

|  |  |  |
| --- | --- | --- |
| **Flag** | **Direction** | **Applicable Grid Modes** |
| SDP\_N | North | grid4, grid8, hex |
| SDP\_S | South | grid4, grid8, hex |
| SDP\_E | East | grid4, grid8, hex |
| SDP\_W | West | grid4, grid8, hex |
| SDP\_NE | Northeast | grid8, hex |
| SDP\_SE | Southeast | grid8, hex |
| SDP\_NW | Northwest | grid8, hex |
| SDP\_SW | Southwest | grid8, hex |

**Table 2.** Grid mode flags.

*C++ API functions Descriptions*

The following are brief summaries of the functions associated with the Spatial Data Provider interface. For a more detailed description of these functions, see the OME source code, or the auto-generated Doxygen documentation.

**AddFieldCol(<label>, <...>)** - Add a new column/record field with with **<label>** and initialized according to the additional arguments in **<...>**, which vary depending on which overloaded function is being used.

**GetCapabilities()** - Queries the SpatialDataProvider for its supported capabilities, which returns any combination of the flags in the above table.

**GetCellSize(<width>,<height>)** - On return, sets **<width>** and **<height>** to a grid cell's standard width and height, or returns false if SDP is not representing a gridded coverage.

**GetColumnCount()** - Returns the total number of fields per spatial record. Required for all implementations.

**GetData(<record index>,<attribute index>,<value>)** - If both **<record index>** and **<attribute index>** are valid, populates **<value>** on return. Otherwise, returns false.

**GetDataMinMax(<attribute index>,<minValue>,<maxValue>)** - If **<attribute index>** is a valid index **<minValue>** and **<maxValue>** are populated by the minimum and maximum values found in that attribute across all records, respectively. If **<attribute index>** is -1, the minimum and maximum values across all attributes and records are retrieved. If **<attribute index>** is invalid and not -1, false is returned.

**GetExtents(<xMin>, <xMax>, <yMin>, <yMax>)** - Populates **<xMin>**, **<xMax>**, **<yMin>**, and **<yMax>** with the minimum and maximum extents of of the x and y axes. Required for all implementations.

**GetFieldCol(<label>)** - Retrieve the index of the column who is named **<label>** or -1 if no column with that name exists. Required for all implementations.

**GetIterator(<kind>)** - returns an iterator that moves either forward or backward through records based on the type of iterator specified by **<kind>**.

**GetNeighborCount(<record index>)** - Returns the total number of neighbors for the record at

**<record index>**.

**GetNeighborRecord(<record index>, <neighbor>)** - Returns the record index of the nth **<neighbor>** of the record at **<record index>**. This can be used in conjunction with **GetNeighborCount()** to iterate through all neighboring records.

**GetNeighbors(<record index>,<count>)** - Returns a list of indices for the total number of neighbors for the record at **<record index>**, with **<count>** containing the total number of indices returned.

**GetNeighborForDirection(<record index>,<grid\_dir>,<neighbor index>)** – Returns true if there is a neighbor for **<record index>** in the direction of **<grid\_dir>**. If the specified neighbor exists, the **<neighbor index>** is set to the neighbor’s index on return.

**GetNeighborRecord(<record index>,<nth neighbor>)** – Retrieve the index of the **<nth neighbor>** record for the record at **<record index>**, or -1 if no such neighbor exists.

**GetNeighbors(<record index>,<count>)** – Returns a list containing the indices for each neighbor record for **<record index>**. On return, **<count>** is set to the total number of entries in the returned list.

**GetNextTo(<record index>,<count>)** - retrieves an iterator to a record of all neighbors of the record at **<record index>**, optionally populating **<count>** with the total number of records found.

**GetRowCount()** - Returns the total number of spatial records. Required for all implementations.

**GetWithin(<record index>,<query>,<distance>)** - Retrieve an iterator to records within **<distance>** from the record at **<record index>** which satisfy **<query>**, whose syntax is SDP-dependent.

**HasNeighborForDirection(<record index>,<grid\_dir>,<neighbor index>)** – Returns true if there is a neighbor for **<record index>** in the direction of **<grid\_dir>**.

**Load(<filename>,<optional expression parser>)** – Attempts to load parser data from a file pointed to by **<filename>**, with an optional expression parser being passed in as well. Returns a flag indicating success or failure.

**NextTo(<index>, <query>)** - Returns true if the spatial component at **<index>** has any neighbors that satisfy the **<query>** whose syntax is SDP-dependent. Otherwise, returns false.

**NextToArea(<index>,<query>)** - Return the total area of all neighbors of the record at **<index>** that satisfy **<query>**, whose syntax is SDP-dependent.

**Save(<filename>,<flags>)** - Attempts to save data to a file at **<filename>**, using any optional **<flags>** that are interpreted by the specific SDP. Returns a flag indicating success or failure in saving the file.

**SetData(<record index>,<attribute index>,<value>)** - Set the value of the attribute at **<attribute index>** for the record at **<record index>** to **<value>**.

**Within(<index>,<query>,<distance>)** - Returns true if any records within **<distance>** from the record at **<index>** satisfy **<query>**, whose syntax is SDP-dependent. Otherwise, returns false.

**WithinArea(<index>,<query>,<distance>)** - Returns the total are of all records within **<distance>** from the record at **<index>** that satisfy **<query>**, whose syntax is SDP-dependent.

*Model Component Expression SDP Functions*

The following functions are for use within the expressions used to for updating model component values. All expression functions that interact directly with the loaded SDP are prefixed with “SDP”. A “spatial representation unit” refers to the base unit of representing a section of space; typically this is a grid cell or a polygon.

**OLE-object**

**Table 2.** . The usage groups are: 1) querying details about the whole coverage, 2) querying about details for a specific attribute across all spatial records, 3) getting and/or setting values specific attributes in a specific spatial record, and 4) querying about spatial relations between spatial records.

**SDPGetBooleanData(<row>,<column>)** - Return boolean value for **<column>** in **<row>**, or nil/NULL if it doesn't exist.

**SDPGetCapabilityFlags()** - Return the flag markers for capabilities of SDP. See Table 5 for a list of possible flags.

**SDPGetCellSize()** - Return the extents (width, height) for a given cell in a gridded coverage.

**SDPGetColumnCount()** - Return the number of columns of attributes within the SDP.

**SDPGetExtents()** - Return the four values representing the extents of the spatial coverage: the minimum x-value, the maximum x-value, the minimum y-value, and the maximum y-value.

**SDPGetFieldCol(<label>)** - Return index of column with the header matching **<label>**, or -1 if no column matches.

**SDPGetDataMinMax(<index>)** - Return the minimum and maximum values within the column at **<index>**, or the minimum and maximum values for all numeric columns in the coverage if **<index>** is -1.

**SDPGetIntegerData(<row>,<column>)** - Return the integer value for the attribute in **<column>** and **<row>**, or nil/NULL if it doesn't exist.

**SDPGetNumberData(<row>,<column>)** - Return the floating-point value for the attribute in **<column>** and **<row>**, or nil/NULL if it doesn't exist.

**SDPGetNeighborCount(<record>) –** Return the number of neighbors for **<record>**.

**SDPGetNeighborForDirection(<record>,<grid direction>) –** Return the index of the neighbor of **<record>** in the direction of **<grid direction>**.

**SDPGetRowCount()** - Return the number of rows in spatial data provider.

**SDPGetStringData(<row>,<column>,<maxChars>)** - Return a string value representation for the attribute in **<column>** and **<row>**, limiting the length of the returned string to **<maxChars>**. If the record or value does not exist, nil/NULL is returned instead.

**SDPGetNeighborForDirection(<record>,<grid direction>) –** Return true if **<record>** has a neighbor in the direction of **<grid direction>**.

**SDPListCapabilities()** - Return A string listing all flagged capabilities.

**SDPNextTo(<row>,<query>)** - Return true if any neighbors next to the spatial representation unit at **<row>** validate **<query>**.

**SDPNextToArea(<row>)** - Return the total area of all neighbors next to the spatial representation unit at **<row>**.

**SDPNextToIDs(<row>)** - Return array of row ids for all neighbors of the spatial representation unit at **<row>**.

**SDPNextToValues(<row>,<label>)** - Return array of values for the attribute column whose header matches **<label>** for all neighbors next to the spatial representation unit at **<row>**.

**SDPSetData(<row>,<column>,<value>)** - Set value of the attribute in **<column>** for **<row>** to **<value>**.

**SDPWithin(<row>,<query>,<dist>)** - Return true if any spatial representation units are within a distance of **<dist>** from the spatial representation unit at **<row>** validate **<query>**.

**SDPWithinArea(<row>,<query>,<dist>)** - Return area of all spatial representation units are within a distance of **<dist>** from the spatial representation unit at **<row>** validate **<query>**.