Provider Syntax for Replicate Endpoints

When you use an ODBC endpoint in Attunity Replicate, you can define a provider syntax that Attunity Replicate will use instead of the default provider syntax.

In this document:

Creating the Syntax

The syntax must be saved in a JSON file.

The available syntax options are described in the following sections:

- Creating a Query Template
- Creating Data Type Mapping between the provider data types and Attunity Replicate data types.
- Setting ODBC Native Error Codes
- Using CSV Files to Load Data into the Target Database

For further reference on creating this file, see List of Keywords

Creating a Query Template

Create a syntax file to change the default SQL queries. Note that default values are used for statements that are not included in the syntax file. The following is an example of a file that changes the default value for `Rename Table`.

```
{
    "name": "SYNTAX_DB2",
    "repository.provider_syntax":  {
        "name": "SYNTAX_DB2",
        "query_syntax":
             {
              "rename_table": "RENAME TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_
NAME}${QC} TO ${QO}${NEW_TABLE_NAME}${QC}"
              }

                ,

         "data_type_mapping": [
              {

               }

          ]
       }
}
```

Where:

`name` is the name of the syntax you are using.

`query_syntax` is the changes you are making to the query syntax.

Note: Only non-default queries are listed in this file. All other queries use their default value. The following describes the queries you can include in this file and their default values.

- **create_schema**: = "CREATE SCHEMA ${QO}${SCHEMA_NAME}${QC}"
- **create_table**: "CREATE TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} (${COLUMN_LIST})"
- **create_index**: " CREATE UNIQUE INDEX ${QO}${CONSTRAINT_NAME}${QC} on ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} (${COLUMN_LIST})"

- **create_primary_key**: "ALTER TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} ADD CONSTRAINT ${QO}${CONSTRAINT_NAME}${QC} PRIMARY KEY (${COLUMN_LIST})"
- **drop_table**: "DROP TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC}"
- **truncate_table**: "TRUNCATE TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC}"
- **add_column**: "ALTER TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} ADD COLUMN ${QO}${COLUMN_NAME}${QC} ${COLUMN_TYPE}"
- **drop_column**: "ALTER TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} DROP COLUMN ${QO}${COLUMN_NAME}${QC}"
- **rename_column**: "ALTER TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} RENAME COLUMN ${QO}${COLUMN_NAME}${QC} TO ${QO}${NEW_COLUMN_NAME}${QC}"
- **modify_column**: "ALTER TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} ALTER COLUMN ${QO}${COLUMN_NAME}${QC} ${COLUMN_TYPE}"
- **rename_table**: "ALTER TABLE ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC} RENAME TO ${QO}${NEW_TABLE_NAME}${QC}"
- **reorg** (used by DB2 LUW - default is empty): ""
- **create_clustered_index** (default is empty): ""
- **use_owner_name**: default is `true`. When set to `false`, all DML queries will be executed without owner name. Set this property to `false` when changing the above queries to not use owner name.

Creating Data Type Mapping

You can also define the data type mapping from the ODBC provider data to an Attunity Replicate data type. The following example shows how to enter the information:

```
{
    "name": "SYNTAX_DB2",
    "repository.provider_syntax":  {
        "name": "SYNTAX_DB2",
        "query_syntax":
             {

              }

                ,

         "data_type_mapping": [
              {
              "rep_type":     "kAR_DATA_TYPE_NUMERIC",
              "provider_data_type":    "NUMERIC(${PRECISION},${SCALE})"
               }
          ]
       }
}
```

Where:

`rep_type` is the name of the Attunity Replicate data type. Note that all Attunity Replicate data types are written as `kAR_DATA_TYPE_`*`name of data type`*

`provider_data_type` is the name of the data type from the ODBC provider.

To view a list of the default query and data type mappings for a specific database and optionally export them to a JSON file, open the Attunity Replicate command line prompt and issue the following command:

```
repctl.exe getprovidersyntax database_name=odbc_endpoint_name
syntax_name=endpoint_syntax_name [>> export_file_path]
```

where:

`odbc_endpoint_name` is the name defined for the endpoint in Attunity Replicate.

`endpoint_syntax_name` is the syntax name for the specific database. For a list of valid syntax names, see Overriding Target Endpoint Syntax Names and Properties.

`>> export_file_path` is the full path to the export file. Make sure to insert a space both before and after the double arrowheads (>>), as shown in the Example command above.

Note: If you specify the file name only, the file will be exported to **<product_dir>\bin**.

Example command:

```
repctl.exe getprovidersyntax database_name=mytargetsqlserver
syntax_name=MySQL >> C:\Exports\mysqlsyntax.json
```

Defining Conditional Mapping

You can specify conditional mapping according to length, scale or precision:

Example according to length:

```
{"rep_type": "kAR_DATA_TYPE_WSTR", "provider_data_type": "NVARCHAR(${LENGTH})",
"has_length_condition": true, "from_length": 1, "to_length": 16000},
{"rep_type": "kAR_DATA_TYPE_WSTR", "provider_data_type": "NVARCHAR(16000)", "has_
length_condition": true, "from_length": 16001, "to_length": 2147483647},
```

Example according to scale:

```
{"rep_type": "kAR_DATA_TYPE_TIMESTAMP", "provider_data_type": "TIMESTAMP", "has_
scale_condition": true, "from_scale": 0 , "to_scale": 6},
{"rep_type": "kAR_DATA_TYPE_TIMESTAMP", "provider_data_type": "VARCHAR(37)", "has_
scale_condition": true, "from_scale": 7 , "to_scale": 9},
```

Example according to precision:

```
 {"rep_type": "kAR_DATA_TYPE_NUMERIC", "provider_data_type":
"Numeric(${PRECISION}, ${SCALE})", "has_precision_condition": true, "from_
precision": 0,"to_precision": 18,"casting_data_type": ""},
 {"rep_type": "kAR_DATA_TYPE_NUMERIC","provider_data_type":
"Varchar(${PRECISION})", "has_precision_condition": true, "from_precision": 19,
"to_precision": 200, "casting_data_type": ""},
```

List of Keywords

Query and Data Type Mapping Keywords

The following is a list of keywords you can use when creating a query, defining data type mapping, or defining an external load utility.

- ${TABLE_OWNER}: The table owner.
- ${TABLE_NAME}: The table name.
- ${SCHEMA_NAME}: The schema name.
- ${COLUMN_NAME}: Column name (e.g. when adding, changing columns).
- ${LENGTH}: Length of column. This can be used in data type mapping when creating the table.
- ${LENGTHINBYTES}: The length of the column in bytes.
- ${PRECISION}: Precision of column. This can be used in data type mapping when creating the table.
- ${SCALE}: Scale of column. This can be used in data type mapping when creating the table.
- ${NEW_TABLE_NAME} New table name when renaming the table name.
- ${NEW_COLUMN_NAME}: New column name when renaming the column name.
- ${COLUMN_LIST}: All the table columns (name and data type).
- ${PK_COLUMN_LIST}: All the primary key columns.
- ${COLUMN_TYPE}: Column type, used when changing column data type.
- ${CONSTRAINT_NAME}: Constraint name (e.g. when creating primary key).
- ${QO}: The character string that is used as the starting delimiter of a quoted (delimited) identifier in SQL statements.
- ${QC}: The character string that is used as the ending delimiter of a quoted (delimited) identifier in SQL statements.
- ${LOB_MAX_SIZE}: When the **Limit LOB size to** task settings option is enabled, this is the maximum LOB size. For unlimited LOB size, set the value to "0".

External Load Utility Keywords

The following is a list of keywords you can use when loading data with an external loading utility:

- ${SERVER}
- ${PORT}
- ${DATABASE}
- ${USER_NAME}
- ${PASSWORD}
- ${FILENAME}
- ${BAD_RECORDS_FILE}: The file in which to store bad records (TASK_NAME/data_files/TABLE_ID.bad)
- ${OUPUT_DIR}: The Replicate log files directory.

Implementing the Query and Data Type Mapping

This sections describes how to implement the mapping you define when creating the JSON file as described in Creating a Query Template and Creating Data Type Mapping.

To implement the query and data type mapping

1. Create a JSON file with the query and/or data type mappings. Make sure that you include the syntax name that you are working with.

   For example:

   `"name":"SYNTAX_DB2".`

2. Save the JSON file and remember the file name you give it. The file name will be appended with the extension, `.json`.
3. Copy the JSON file to the Attunity Replicate `prod\data\imports` folder on the computer where you installed Attunity Replicate.
4. To import the file, open the Attunity Replicate command-line console and issue the following command:

   `run repctl.exe putobject data=`*`file_name`*

   Note: Do not append the extension `.json` in the file name.
5. In the Attunity Replicate Console, when you create the ODBC database you must include the **Provider syntax** field in the **Advanced** tab of the Add Database dialog box.

Using the GenericWithoutSchema Provider Syntax

A special syntax, called GenericWithoutSchema is provided for use. This syntax is useful when working with databases that do not support `owner`. It has the following query syntax values:

- **create_schema**: = ""
- **create_table**:

  "CREATE TABLE ${QO}${TABLE_NAME}${QC} (${COLUMN_LIST})"
- **create_index**: "CREATE UNIQUE INDEX ${QO}${CONSTRAINT_NAME}${QC} on ${QO}${TABLE_NAME}${QC} (${COLUMN_LIST})"
- **create_primary_key**: "ALTER TABLE ${QO}${TABLE_NAME}${QC} ADD CONSTRAINT ${QO}${CONSTRAINT_NAME}${QC} PRIMARY KEY (${COLUMN_LIST})"
- **drop_table**: " DROP TABLE ${QO}${TABLE_NAME}${QC}"
- **truncate_table**: "TRUNCATE TABLE ${QO}${TABLE_NAME}${QC}"
- **add_column**: "ALTER TABLE ${QO}${TABLE_NAME}${QC} ADD COLUMN ${QO}${COLUMN_NAME}${QC} ${COLUMN_TYPE}"
- **drop_column**: "ALTER TABLE ${QO}${TABLE_NAME}${QC} DROP COLUMN ${QO}${COLUMN_NAME}${QC}"
- **rename_column**: "ALTER TABLE ${QO}${TABLE_NAME}${QC} RENAME COLUMN ${QO}${COLUMN_NAME}${QC} TO ${QO}${NEW_COLUMN_NAME}${QC}"

- **modify_column**: "ALTER TABLE ${QO}${TABLE_NAME}${QC} ALTER COLUMN ${QO}${COLUMN_NAME}${QC} ${COLUMN_TYPE}"
- **rename_table**: "ALTER TABLE ${QO}${TABLE_NAME}${QC} RENAME TO ${QO}${NEW_TABLE_NAME}${QC}"
- **use_owner_name**: `false`. Ensures that all DML queries are executed without owner name.

To use the GenericWithoutSchema queries template

In the Attunity Replicate Console, when you define the ODBC endpoint (source or target), specify **GenericWithoutSchema** in the **Advanced** tab's **Provider syntax** field.

Setting ODBC Native Error Codes

In the vast majority of cases, Replicate handles error codes correctly. Occasionally, however, an error code may be handled incorrectly. In such cases, you can "force" Replicate to handle the error correctly by defining the ODBC native error code(s) for the error type.

Error codes can be defined for the following error types:

- `error_code_constraint_violation`
- `error_code_connection_failure`
- `error_code_data_failure`
- `error_code_table_error`
- `error_code_fatal_error`
- `load_exe_network_error_codes` - The error generated by the load utility when it is unable to establish a connection to the database.

Error Code Example

```
"error_code_data_failure": "7125,8114,6522,511"
```

Using CSV Files to Load Data into the Target Database

When using CSV files to load data into the target endpoint, the following properties can be changed in the syntax file:

File Properties

Delimiter properties:

- `"csv_delimiter"` - Default value: `","`
- `"csv_null_value"` - Default value: `""`
- `"csv_string_quote"` - Default value: `"\""`
- `"csv_string_escape"` - Default value: `""`

Loading Properties

Use the `"sql_load_data_statement"` to set the loading properties for CSV files:

Example:

```
"sql_load_data_statement":"COPY ${QO}${TABLE_OWNER}${QC}.${QO}${TABLE_NAME}${QC}
FROM LOCAL '${FILENAME}'  DELIMITER ',' ESCAPE AS '\\'  ENCLOSED BY '\"' NULL
'attNULL' DIRECT ABORT ON ERROR"
```

Properties for Loading using an External Utility

- `"load_data_exe_full_path"`

  Example:

  ```
  "load_data_exe_full_path":"",
  ```

- `"load_data_exe_name"`

  Example:

  ```
  "load_data_exe_name":"nzload.exe",
  ```

- `"load_data_exe_params"`

  Example:

  ```
  "load_data_exe_params":"-quotedValue DOUBLE -host ${SERVER} -db ${DATABASE}
  -nullvalue attN -u ${USER_NAME} -pw ${PASSWORD} -delim , -t \\\"${TABLE_
  OWNER}\\\".\\\"${TABLE_NAME}\\\" -df \"${FILENAME}\" -escapeChar \\ -outputDir
  \"${OUTPUT_DIR}\"",
  ```

Defining Syntax Values as Internal Parameters

To define syntax values as internal parameter, use the following format:

`$info.query_syntax.`*`property_name`*

Where:

`property_name` is the name of the query you want to set.

Example

To change the primary key statement:

```
$info.query_syntax.create_primary_key=ALTER TABLE ${QO}${TABLE_
OWNER}${QC}.${QO}${TABLE_NAME}${QC} ADD CONSTRAINT ${QO}${CONSTRAINT_NAME}${QC}
PRIMARY KEY ( ${COLUMN_LIST} )
```

Overriding Target Endpoint Syntax Names and Properties

This section lists which target endpoint syntax names are hard-coded and which can be changed.

Table 0–1 Endpoints with Hard-Coded Syntax Names

Endpoint Name	Syntax Name
Microsoft SQL Server	SQLServer
Pivotal Greenplum	Greenplum
Actian Vectorwise	Vectorwise
Teradata Database	Teradata
Amazon Redshift	Redshift

Note: Although a hard-coded syntax name cannot be changed, the syntax properties can still be overridden.

Table 0–2 Endpoints with Syntax Names that can be Overridden

Endpoint Name	Default Syntax Name
Sybase IQ	SybaseIQ
Vertica	Vertica
Sybase ASE	Sybase
IBM Netezza	Netezza
MySQL	MySQL
PostgreSQL	PostgreSQL
Micordot SQL Server PDW	SQLServerPDW

If you override an endpoint's syntax name, you need to add the following internal parameter to the endpoint definition:

`syntax=`*`syntax_name`*

For example:

```
syntax=CustomSyntaxName
```