pitney bowes

EasyLoader

Version 15.2

User Guide

Table of Contents

1 - Uploading Data with MapInfo EasyLoader

1 - Uploading Data with MapInfo EasyLoader

The EasyLoader utility lets you to upload MapInfo® TAB files to a remote database, such as Oracle, SQL Server, Microsoft Access or PostgresSQL/PostGIS. The spatial information in the TAB files is maintained in the remote database and is available for viewing and analyzing in MapInfo® Pro.

In this section

Introduction

EasyLoader is a utility available from EasyLoader that allows you to upload MapInfo TAB files to a remote database. The spatial information in the TAB files is maintained in a SQL Server database and is available for viewing and analyzing in MapInfo Pro. EasyLoader can also upload a text object to SQL Server if text-supported SpatialWare (4.6 or later) is on the server.

EasyLoader is installed into the `\Tools` directory during the MapInfo Pro installation process. EasyLoader supports the following databases:

- Oracle
- SQL Server
- Microsoft Access
- PostgreSQL / PostGIS

For spatial database support, the Database Management System (DBMS) must be able to handle spatial geometry, either by itself (as in Oracle Spatial) or through MapInfo SpatialWare (for SQL Server). If one of the above DBMS does not have spatial object type support, the table can only be uploaded as XY data: XY or XY with MapInfo Key (MICode). Only one server connection may be open at any one time.

EasyLoader works with both 32-bit and 64-bit versions of MapInfo Pro. However, EasyLoader is a 32-bit software product that requires having 32-bit ODBC drivers installed, such as the PostgreSQL ANSI ODBC driver. Ensure you have the correct drivers when using EasyLoader with MapInfo Pro 64-bit.

To obtain the latest copy of EasyLoader, go to **www.mapinfo.com** and search for the EasyLoader download page where you can download both EasyLoader and the *EasyLoader User Guide*.

System Requirements

This product is tested on the following Microsoft Windows Desktop Operating Systems:

- Windows 10 64-bit
- Windows 8.1 64-bit
- Windows 7 Ultimate 64-bit SP1
- Windows 2012 Server R2 64-bit SP1
- Windows 2012 Server R2 64-bit with XenApp 7.5
- Windows 2008 Server R2 64-bit SP1
- Windows 2008 Server R2 64-bit SP1 with XenApp 6.0

History of EasyLoader Changes

This section provides a history of features and enhancements that have been added to EasyLoader since version 11.0.

New in EasyLoader 15.2

EasyLoader 15.2 is a 32-bit release that works with the MapInfo Pro 15.2 64-bit release. There are no new features or fixes.

MapInfo Pro 15.2 supports unicode and requires the PostgreSQL Unicode ODBC drive. EasyLoader does not yet support unicode and requires the PostgreSQL ANSI ODBC drive. You must have both drivers installed when both products are on the same machine.

New in EasyLoader 15.0

EasyLoader 15.0 is a 32-bit release that works with the MapInfo Pro 15.0 32-bit release. There are no new features or fixes.

New in EasyLoader 12.5.1

EasyLoader 12.5.1 is a 32-bit release that works with the MapInfo Pro 12.5.1 64-bit release. There are no new features or fixes.

New in EasyLoader 12.5

There are no new features or fixes.

New in EasyLoader 12.0

EasyLoader 12.0 supports the following database versions:

- **PostGIS 2.0** – Note that when uploading files to a PostGIS 2.0 database, spatial objects upload as a geometry spatial data type.
- **SQL Server 2012** – EasyLoader processes any invalid geography using the SQL Server 2012 MakeValid( ) function, so that the geography is valid.

The server table processing option **Grant Public Access to Table** has been changed to **Grant Public Full Access to Table**.

This release fixes the following issues:

- **MIPRO-12309**: A message displays when appending a native table with a MI_PRINX column in an Oracle Spatial database with an OCI connection, "Table cannot be Appended—It has a different table structure."
- **MIPRO-12320**: When uploading a table with no geometry into EasyLoader 11.5, an error message appears: "Table cannot be Appended—It has a different table structure."

- **MIPRO-32642**: EasyLoader is unable to import to “geography” data type in SQL Server 2008 SP2.

  You will need to upgrade to SQL Server 2012. SQL Server 2012 uses the function STIsValid( ) to check if a geography data type is well formed and recognized as a valid object based on its OGC type. The MakeValid( ) function is also used that converts an invalid geography data type into a valid one. In MapInfo Pro 12.0, you can use the extended spatial methods in SQL Server 2012 to manage valid geography data types. Refer to the **SQL Server 2012 MSDN library** for more information.

New in EasyLoader 11.5.1

EasyLoader 11.5.1 lets you connect to Oracle using your Operating System (OS) authentication (Windows credentials). The **MapInfo Oracle Connect** dialog box has a new **Use Operating System Authentication** check box to set this option.

When running EasyLoader from the command line with the Connection parameter and connecting with OS authentication, the parameter should contain only the server name (*@ServerName*) and not a user ID or password. For the database authentication, provide the user ID, password, and server name (`UserName/Password@ServerName`).

The **/Z** parameter changed in EasyLoader to set the **Auto Select** option instead of **Always Geometry**.

This release fixes the following issues:

- **MIPRO-19759**: In EasyLoader, a connection string created in a tab file does not seem to work every time.
- **MIPRO-21235**: EasyLoader cannot upload geography from the command line in SQL Server 2008.
- **MIPRO-24515**: Uploading a table with a boolean field to PostGIS does not put the correct TYPE_NAME on the server side. There is an inconsistency between EasyLoader and MapInfo Professional on how they save data on PostGIS. EasyLoader uploads a logical field as smallint, whereas, MapInfo Professional uploads a logical field as boolean. When you open a first table, uploaded by EasyLoader, in MapInfo Professional the field shows up as smallint. When you open a second table, uploaded by MapInfo Professional, in MapInfo Professional the field shows up as char[5].

Uploading Data with EasyLoader

You must set up your ODBC connections prior to uploading TAB files to remote databases.

Note: You cannot replace version-enabled tables in the Oracle database. It makes any child versions of these tables obsolete.

To upload MapInfo TAB files using EasyLoader:

1. In MapInfo Pro, on the **HOME** tab, in the **Tools** group, click **Tools** to open the **Tools Manager** window.
2. On the **Running** tab, double click **EasyLoader** to open the **EasyLoader** dialog box.

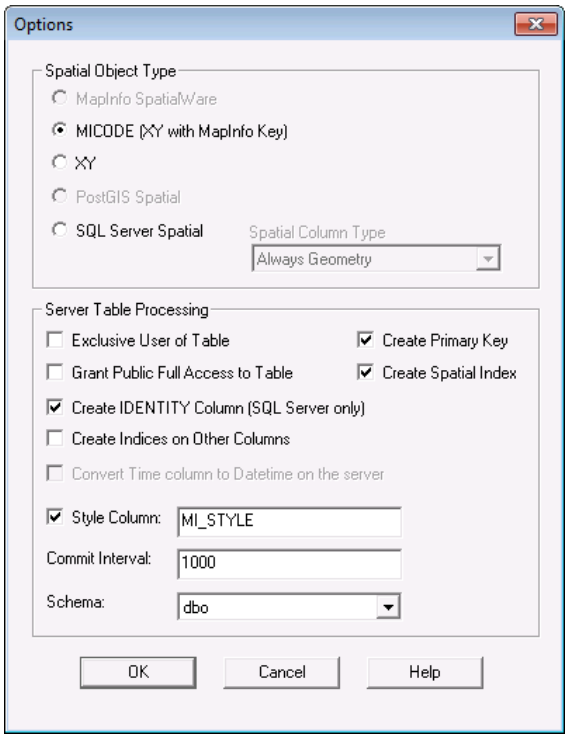
   If you do not see it listed on the **Running** tab, then locate it on the **Registered** tab.

   To load EasyLoader into the current session of MapInfo Pro, click the **Load Tool (Run)** icon beside it.

   To reload EasyLoader for all subsequent sessions, select the **AutoLoad** check box.

   To load EasyLoader for the current session and all subsequent sessions, select both.
3. Under **Connection Information**, click the appropriate button (**ODBC** or **Oracle Spatial**) to connect to your database. Provide the necessary connection information (for example, data source name or User ID, password, and server name). Click **OK** to return to the EasyLoader dialog box.
4. Click **Source Tables** to display a list of MapInfo tables from a single directory. When you have selected the tables for uploading, the names display in the **MapInfo Table** list box.
5. Choose the tables you want to upload and select the appropriate Server Table processing task (Create new table, Append to existing table, Replace existing Table).

   Note: The **Upload** button is not available until one or more tables are chosen.
6. To create local TAB files, provide a directory or browse to its location. By default, EasyLoader does not generate these files. The file naming convention for these tables is `yourServerTableName_srv.tab`.
7. To set options for the upload process, click **Options**.



Note: See **Understanding the Options Dialog Box** on page 15 for an explanation of available options.

8. In the **Options** dialog box, select the appropriate options and click **OK**.
9. Click **Upload** to start the upload process.
10. Close EasyLoader after the upload process is finished.

If you haven't already created the spatial index during the upload process, do so now by either executing a create index statement or re-uploading the table, making sure this time to select the **Create Spatial Index** check box and replace the table (see steps 1-3).

Unsupported Geometries in Oracle, SQL Server, and PostGIS

Some times when you are creating a Map in MapInfo Pro and you are storing the results in Oracle, SQL Server Spatialware, SQL Server Spatial, or PostGIS, you create maps which use geometries that are not supported by these DBMS engines.

- **Oracle** does not support arcs, ellipses, rectangles, and rounded rectangles.
- **SQL Server Spatial** does not support arcs, and lines that do not have distinct points

- **SQL Server SpatialWare** does not support arcs, ellipses, and rounded rectangles.
- **PostGIS** does not support arcs, ellipses, rectangles, and rounded rectangles

EasyLoader will skip the unsupported object type and insert the attribute data, it will not dislay a message.

SQL Server Restriction on Geography Objects

SQL Server uploads invalid Geometry objects and converts them to valid using a SQL server command, but SQL Server does not upload invalid Geography objects. Attempting to upload invalid Geography objects causes EasyLoader to fail. This is a limitation of SQL Server and not of EasyLoader.

Due to this SQL Server limitation, our recommendation is to upload only as Geometry.

Loading Oracle Spatial Data

You can upload Oracle Spatial data using EasyLoader.

Loading for Oracle Locator

EasyLoader loads data for the Oracle Locator. Loading data for the Oracle Locator is the same as loading data for Oracle Spatial.

Loading Long/Lat Tables into Oracle 9i

When uploading tables that use the Longitude/Latitude coordinate system (Geodetic Data) to Oracle 9i, it is important to verify that all geometry coordinates are between (-180,180) longitude and (-90, 90) latitude. Geodetic data coordinates beyond that range are not supported in Oracle Spatial and may cause problems. You may check your data using MapInfo Pro before loading, or by using the Oracle Spatial SDO_GEOM.VALIDATE_LAYER( ) function on the table after loading it to Oracle Spatial.

Note: EasyLoader assigns the default datum World Geodetic System 1984 (WGS84) to any “datumless” Long/Lat coordinate system when loading Oracle 9i.

Loading Native Tables that Contain Text Objects into Oracle 11g

If you are using Oracle 11g or later, you can use EasyLoader to upload a native table (TAB file) that contains text objects to an Oracle Spatial database table. (Previous versions of EasyLoader discarded

any text objects when uploading a TAB file to Oracle.) When you upload the TAB file, EasyLoader uploads the text objects to Oracle as annotation text fields.

Oracle 11g stores MapInfo geometry objects (such as points, polylines, and polygons) as SDO_GEOMETRY types and stores MapInfo text objects as ANNOTATION_TEXT types. A TAB file may contain both geometry and text objects. MapInfo Pro, however, only supports database tables that have one object column. Because of this limitation, if the TAB file you are uploading contains both types of objects, EasyLoader will prompt you to select which object type you want to upload: **Text Objects** or **Geometry**.

When you upload a TAB file that contains text objects to Oracle 11g, you are prompted to specify a numeric value representing the map scale to be used along with the text attributes. Enter a value in the **Default Map Base Scale** field or keep the value shown.

You can define map base scale as a value at which the text will be drawn on the map at the size specified in the attributes.

Validating Oracle Data

There are two functions that allow you to validate data on Oracle:

1. SDO_GEOM.VALIDATE_GEOMETRY( )
2. SDO_GEOM.VALIDATE_LAYER( )

These functions may result in validation errors due to the tolerance level set by EasyLoader. You may get error messages such as:

"ORA-13356 adjacent points in a geometry are redundant", or "ORA-13022 polygon crosses itself"

To resolve these errors, reset the tolerance within the USER_SDO-GEOM_METADATA by adjusting them downwards (by a factor of 10) and rerun the validation.

Note: If you adjust the tolerance, you must re-create the spatial indexes because they use the tolerance when they are created.

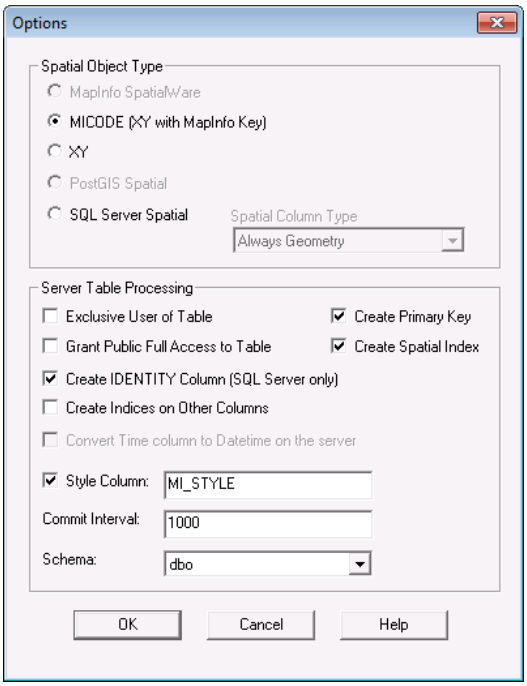
To view the Unsupported Geometries in Oracle See **Unsupported Geometries in Oracle, SQL Server, and PostGIS**.

Loading Microsoft SQL Server Spatial Data

Microsoft SQL Server Spatial includes Geography and Geometry data types. Geography fields hold geometries specified in the Lat/Long coordinate system. Geometry fields hold geometries specified in all other coordinate systems.

If you choose to upload to a Geometry field type, then EasyLoader runs the SQL Server Spatial MakeValid( ) function to fix any geometry that SQL Server Spatial deems invalid. This may shift point locations for certain geometries.

In the **Options** dialog box, EasyLoader enables you to control how the geometry data in a native table is uploaded to SQL Server Spatial.



Keep the default selection of **Always Geometry**if you want EasyLoader to always upload geometry objects to Geometry fields regardless of the coordinate system they use.

Select **Auto Select**if you want EasyLoader to automatically decide whether the geometry object is uploaded to a Geography field (in cases where the geometry object uses the Lat/Long coordinate system) or a Geometry field (in cases where the geometry object does not use the Lat/Long coordinate system).

Note: When the Auto Select option is selected, if for any reason a geometry object that uses the Lat/Long coordinate system cannot be uploaded to a Geography field, it will instead be uploaded to a Geometry field.

To view the Unsupported Geometries in SQL, see **Unsupported Geometries in Oracle, SQL Server, and PostGIS** on page 8.

EasyLoader supports uploading MapInfo native tables into SQL Server Spatial. SQL Server Spatial allows spatial data to be stored into two data types; Geometry and Geography.

SQL Server's rules that define what a valid geometry or geography are different than MapInfo Pro's rules.

The SQL Server Spatial's Geography data type does not support:

- polygons that contain self intersecting boundaries
- geography instances that span more than one hemisphere
- lines/polygon boundaries that have two successive, identical nodes

The SQL Server Spatial's Geometry type does not support:

- polygons that contain self intersecting boundaries
- lines/polygon boundaries that have two successive, identical nodes

Refer to SQL Server books online for complete information on the rules that define these two types at:

http://www.microsoft.com/en-us/sqlserver/default.aspx

This means that you may have geometry data that are valid in a MapInfo TAB format that cannot be loaded into SQL Server Spatial without correction/edits.

EasyLoader aborts the upload if it comes across the geometry that SQL Server rejects. It will display the primary key of the record that contains the rejected object. A "MapInfo Upload Utility Error" message displays.

You may want to try the following steps to make the geometry acceptable to SQL Server.

- Use MapInfo Pro's **Clean** operation on the **Object** menu to correct the geometry.
- If your upload was to Geography type and it fails, try loading it to Geometry, correct the instance using the MakeValid( ) method on the Geometry type, and uploading the instance again. You will need a working knowledge of SQL and access to a SQL Server Spatial client to use this approach.

Consider the following example. Let's suppose the Geometry update failed for the row (state = 'Florida').

- selectively update the offending geometry, using the MakeValid( ) method

```
update states_geom set geom = geom.MakeValid( ) where state
= 'Florida'
```

- force an operation that does not alter the geometry, but forces a topology construction. here, we perform a union of the Geometry with its own start point

```
update states_geom set geom =
geom.STUnion(geom.STStartPoint()) where state = 'Florida'
```

- now this query will attempt to create a Geography instance from the reformed Geometry

```
select state_name, sw_member,
Geography::STGeomFromWKB(geom.STAsBinary(), 4269) from
states_geom, where state = 'Florida'
```

Loading PostGIS Spatial Data

EasyLoader supports uploading MapInfo native tables into PostGIS. PostGIS stores spatial data in a geometry column called sp_geometry. Unsupported data types are ignored resulting in an empty record in the sp_geometry geometry column.

For a list of unsupported data types, see **Unsupported Geometries in Oracle, SQL Server, and PostGIS** on page 8.

When upload a table to a PostGIS database, EasyLoader uploads the table with a Serial datatype for the primary key column. This allows the database to automatically increment the key column.

To check if a geometry object is valid in PostGIS, use the function called isvalid. The following is an example SQL statement:

```
select isvalid(sp_geometry) from test
```

Understanding the EasyLoader Dialog Box

This section explains the options in the **EasyLoader** dialog box.

Connecting to a Remote Database

ODBC Button

The **ODBC** button lets you connect to a server using ODBC and prompts you for a data source to identify where the data you want to upload is located. You must have your data connections set up before using the **ODBC** button to connect to the remote database in this manner.

To connect to a database using the **ODBC** button:

1. Click **ODBC** in the **EasyLoader** dialog box.

   The **Select Data Source** dialog box displays.

2. Do one of the following:

   - Click the **File Data Source** tab and use the **Look in** drop-down list or the **Up** button to locate the database you want to upload from and click it. Click **OK** to display your selection in the **EasyLoader** dialog box.

- Click the **Machine Data Source** tab and double-click the data source driver you want to connect to and navigate to the database you want to upload. Click it and click **OK** to display your selection in the **EasyLoader** dialog box.

Database Options

This section describes the button options and input fields in the EasyLoader for connecting to a database.

ODBC button	Click this button to connect to a server using ODBC. This displays the **Select Data Source** dialog box where you can select a file or machine data source. You must have your data connections set up before selecting the **ODBC** option. For more information, see **Connecting to a Remote Database** on page 13.
Oracle Spatial button	Click this button to connect to an Oracle Spatial server. Enter your user name, password, and the server name to complete the connection. This button is only enabled if the Oracle Client software is installed. Oracle Client software does not come with EasyLoader.
Source Tables button	Click this button to identify the source tables you want to upload. The **Source Tables** button is available only after you have made a connection to a server. This action enables you to select one or more MapInfo tables from a single directory.
Server Table Name	Type the name of the database server table to which you are uploading the selected tables.
Append All Tables to One Server Table	Click this option to upload all MapInfo tables listed in the **MapInfo Table** list to a single server table. The server table name is the one visible in **Server Table** box. This feature should be used to upload tables with the same structure and style to one table. Example: Instead of creating a new table for each street layer, check the **Append All to One** check box, and only one table is created. Then all of the tables are appended to this table.
TAB File Directory for Server Table(s)	Generates TAB files to access remote DBMSs when you provide the TAB file directory. By default, an empty directory, the loader does not generate .tab files. The newly generated .tab file is the Server Table Name plus _srv.tab (`yourServerTableName_srv.tab`). Click **Browse** to search for the directory you need.
Map Catalog button	Click this button to add a new Map Catalog or to unregister a table from the Map Catalog. For more information, see **Creating a New Map Catalog** on page 24.
Upload button	Click this button when you have set all of the parameters you want for uploading the table(s) you have specified.

Options button	Click this button to specify the spatial object types and the server processing options for the current upload. For instructions about using the **Options** dialog box, see **Understanding the Options Dialog Box** on page 15.

Table Processing Options

This section provides a comprehensive description of the table options available in EasyLoader. There are four table processing options associated with the main **EasyLoader** dialog box.

Create New Table	A server table is created with the name that you specify. If this option is chosen and a table with the same name already exists on the server, an error message displays, making you aware of this problem. Use a different name or choose the option **Replace Existing Table to upload** the table.
Replace Existing Table	When this option is selected, if a server table of the same name already exists, it is dropped and a new table is created to match the MapInfo table being uploaded.
Append to Existing Table	The MapInfo table is appended to the server table if the server table exists and the structure of the two tables match. Otherwise, you get an error and the table is not uploaded. The tables must have the same table structure and be in the same projection for Oracle Spatial.
Append All Tables to One Server Table	All MapInfo tables listed are uploaded to a single server table. The server table name is the one visible in the **Server Table** box. This feature is meant to be used to upload tables with the same structure and projection to one table. For example, instead of creating a new table for each street layer, check the **Append All Tables to One Server Table** check box, and only one table is created. All of the tables are then appended to this table. **Note:** It is possible that some tables will not be appended if their table structure differs. If you choose the **Replace Existing Table** option and this check box is selected, the server table is dropped, a new table is created, and all tables listed are appended to that one. If you select the **Create New Table** option and this check box is selected, the server table is created, and all tables listed are appended to that one. **Note:** If this option is chosen, all tables must have the same table structure and be in the same projection.

Understanding the Options Dialog Box

This section explains the settings in the Options dialog box.

Spatial Object Type Options

Choose from **MapInfo SpatialWare** or **Oracle Spatial** (depending on the type of connection), **MICODE** (XY with MapInfo Key), **XY**, **PostGIS Spatial**, or **SQL Server Spatial**. The default for loading spatial data is **MapInfo SpatialWare** or **Oracle Spatial**, if this option is available; otherwise the **MICODE** option is the default.

MapInfo SpatialWare	To select this option, the server must be Oracle Spatial or have MapInfo SpatialWare installed. Tables are uploaded as spatial data. This option is disabled (grayed) if SpatialWare is not installed on the server, or if it is not available to the currently selected database.
MICODE (XY with MapInfo Key)	Use this option if the server is not Oracle Spatial or does not have MapInfo SpatialWare installed. This option stores the data as XY coordinates on the server and creates the server table as a point table. If the MapInfo table to be uploaded is not a point table and this option is chosen, the centroid is abstracted and stored on the server table, if you instruct it to do so. The difference between **XY** and **MICODE** is that the **MICODE** provides a MapInfo key as the spatial index, making its performance superior to **XY**.
XY	This option is the same as the **MICODE** option, except that this option does not create a spatial index.
PostGIS Spatial	Use this option with a spatialized PostGIS database.
SQL Server Spatial	Use these options if the server is Microsoft SQL Server Spatial: **Auto Select** – EasyLoader automatically decides whether to upload the geometry object to a Geography field (if the geometry object is using a Lat/Long coordinate system) or to a Geometry field (if the geometry object is using another coordinate system). **Always Geometry** – EasyLoader always uploads the geometry object to a Geometry field, regardless of the coordinate system specified for the geometry object.

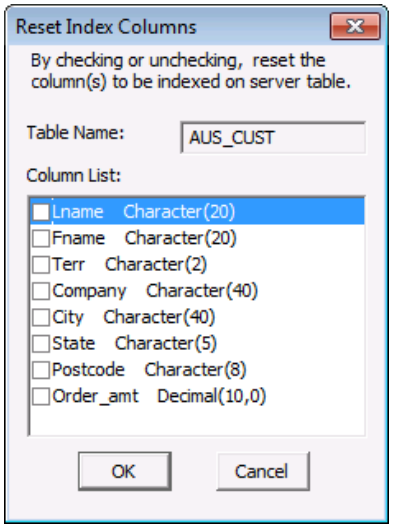
Note: When the **Auto Select** option is selected, if for any reason a geometry object that uses the Lat/Long coordinate system cannot be uploaded to a Geography field, it will instead be uploaded to a Geometry field.

Server Table Processing Options

Exclusive User of Table – You can significantly speed up load time on large tables if you know that you are the only one attempting to update or upload the table. If you do not select this option, EasyLoader verifies after each commit that no other updates are made to the table while it is being

uploaded. Selecting this option prevents this test from occurring, which can improve runtime performance for large tables.

Create Primary Key	By selecting this check box, a primary key is created for the **Create New Table** and **Replace Existing Table** operations. This primary key is created in the column SW_MEMBER for SpatialWare, MI_PRINX for Oracle, or MI_SQL_REC_NUM for XY and MICODE. These columns are sequential numbers that are generated by EasyLoader. These columns are always created, but do not have to be a primary key. For the Append To Existing Table operation, the primary key is not created.
Grant Public Full Access to Table	The PUBLIC is granted all access to the server table.
Create Spatial Index	By selecting this check box, a spatial index (called <tablename>ind) is created for the tables on the geometry column. You may also build your own spatial index to suit your specific needs. If you choose to do this, clear this check box to save loading time. By selecting this check box, a spatial index (called <tablename>_SX) is created for Oracle Spatial tables on the geometry column . The index tiling level is based on the SDO_TUNE.ESTIMATE_TILING_LEVEL function. For tables with fewer than 7500 rows, the tiling level is restricted to 8. After the index is built the ANALYZE table function is run on the index table. The spatial index is R-Tree for Oracle 8.17 or later.
Create IDENTITY Column (SQL Server only)	Select this check box if you wish to create the primary key column (SW_MEMBER) with IDENTITY properties. When this feature is in use, the primary key column values are generated automatically by SQL Server. You do not need to fill in the key manually when a new row is inserted.
Create Indices on Other Columns	Select this check box if you want to index additional columns when uploading a new table or replacing an existing table. By default this box is not selected. When you select this check box and click **OK**, the table upload begins and the following dialog box displays.



The **Table Name** text box shows which server table is selected for indexing. The **Column List** shows each column followed by its data type. Select a check box to select that column for indexing on the server table. Clear a check box if you do not want to index that column.

Note: If the column name length is too long, the index creation fails. The limit to the size of the index name varies per database.

Convert Date and Time columns to Datetime on the server

This option is automatically handled by the software, with selects it when connected to a DBMS, such as Oracle and Access, that does not have date or time data types and unselects it for a DBMS, such as SQL Server Spatial or PostGIS, that has both date and time data types.

Style Column

This option allows you to specify whether per-row styles are loaded with the data. You can also specify the name of the column in the text box next to the **Style Column** check box. The default column name is MI_STYLE.

Note: To load per-row styles, the MapInfo Map catalog for the database must contain the following columns: RENDITIONTYPE, RENDITIONCOLUMN, RENDITIONTABLE, and NUMBER_ROWS. For more information, see **MAPINFO_MAPCATALOG Table Structure** on page 26.

Commit Interval

Use this option to specify a commit interval for uploading. EasyLoader commits the inserted records when the commit interval is reached. The default commit interval is 1000. If the commit interval is set to 0 (zero), the whole range of records is inserted as a single transaction, before a commit is issued.

Schema

Use this option to specify a schema to which you want to upload the table. This is valid for SQL Server 2005 servers and PostgresSQL/PostGIS.

Uploading Tables with Time and DateTime Columns

Time and DateTime data types sometimes require conversion because of variations from server to server, and between the server and MapInfo data types. The following table shows how the data types are converted from MapInfo Pro to each server.

From MapInfo Pro	To Oracle	To MS Access	To MS SQL Server
DATE	DATE	DATETIME*	DATETIME*
TIME	TIMESTAMP(3)*	DATETIME*	DATETIME*
DATETIME	TIMESTAMP(3)	DATETIME	DATETIME

* The MapInfo data type will be extended on the servers. When the same data comes back to MapInfo Pro, the data type will be as is indicated on the server. This conversion is consistent with MapInfo Pro behavior when a MapInfo table is saved to a server using MapInfo Pro.

On servers that do not support DATE or TIME data types, the data is converted to a DATETIME type. In this conversion, part of the data will be missing because the MapInfo types contain either the date or the time, but not both. The server default values for the date or the time are used to fill in the missing data. Conversions to DATETIME are made for SQL Server versions earlier than version 10 (prior to SQL Server Spatial).

For example, if the server does not support the MapInfo DATE type, the upload process converts the DATE type to a DATETIME type. The date value comes from the MapInfo table, but the time value is filled in with the server default value for time. The following table shows how the MapInfo data types are converted when they are not supported on the server and what default value are used to fill in the missing data:

MapInfo Data Type	To Server Data Type	Server Default Values	Databases
MapInfo DATE type	DATETIME/TIMESTAMP type	midnight: 12:00.00.000 AM	All databases
MapInfo TIME type	DATETIME/STAMP type	current date	MS Access, MS SQL Server*
MapInfo TIME type	DATETIME/STAMP type	first day of current month and year	Oracle

* SQL Server versions earlier than version 10 (prior to SQL Server Spatial).

Using the Command Line Flags to Run EasyLoader

You can run the EasyLoader executable from the command line. For example:

```
easyloader.exe /T c:\data\states.tab;mystates /G
/Y
```

Note: Do not enclose file names in quotation marks. Command line arguments are interpreted correctly without quote marks, even if the filename includes spaces. If you use quotation marks, EasyLoader cannot parse the filenames correctly and errors occur.

EasyLoader supports the following flags to allow you to specify additional upload parameters from the command line:

- **/A Append All Tables to One** – Use this flag to upload multiple tables to a single table (as long as the table structures are the same).

  Syntax: /A

- **/B Schema Name** – Use this flag to specify a schema name when you upload tables to a SQL Server 2005 server. If you do not supply a schema name, then this flag uses the user's default schema on the server (on SQL Server the default is usually set to **dbo** for example).

  Syntax: /B SchemaName

- **/C Create Indices for All Locally-Indexed Columns** – Use this flag to mandate that only the columns that were indexed on the local table be automatically indexed on the server table, when you upload a new table or replace an existing table.

  This command line option does not allow you to make or reset column selections. That additional capability is supported by the EasyLoader user interface (see Create Indices on Other Columns under **Server Table Processing Options** on page 16 and **Mixing Command Line Flags with the EasyLoader User Interface** on page 24).

  Syntax: /C

- **/D .tab File Directory for Server Table(s)** – Use this flag to generate TAB files and provide the TAB file directory to access a remote DBMS. The default is an empty directory, in which case EasyLoader does not generate .tab files. The naming convention of your newly generated .tab file is yourServerTableName_srv.tab. The directory must be valid (empty is treated as valid) to upload a table.

  Command line option is /D PathName.

  Syntax: /D C:\temp

  Do not enclose file names in quotation marks.

- **/E Exclusive Use of Table** – Use this flag to improve load time on large tables significantly if you know that you are the only one attempting to update the table. Specifying this flag does not guarantee that EasyLoader can obtain exclusive use; you must guarantee that to the loader. EasyLoader checks on the current maximum value of the primary key column (MI_PRINX) after each commit to ensure that it detects any other entries that may have been made by other processes. This flag prevents that check from occurring, which can significantly improve the upload time for large tables.

  This flag may be placed within a shortcut, allowing the interactive use of the EasyLoader interface for other functions.

  Syntax: /E

- **/F Log File name** – Use this flag to specify the name and location of the log file. Whenever you upload a table, EasyLoader produces a log file. By default, a log file named `EasyLoader.log` is created in the Windows TEMP directory. If you specify a file name but do not provide a path, EasyLoader creates the log in the same directory as the `EasyLoader.EXE` file.

  Syntax: The first example shows just the name of the log file, which is written to the directory in which `EasyLoader.exe` is located; the second example specifies the full path for the log file.

  /F myLogFile.txt

  /F c:\temp\myLogFile.txt

  Do not enclose file names in quotation marks.

- **/G Grant all** – Use this flag to grant all rights to PUBLIC. This flag is turned OFF by default.

  Syntax: /G

- **/I Do Not Create a Spatial Index** – Use this flag to prevent EasyLoader from creating a spatial index on the uploaded table. By default EasyLoader creates a spatial index. This flag is turned OFF by default, meaning a spatial index is created. For IDS/UDO tables, EasyLoader creates a spatial index, and then issues the 'update statistics medium' statement. See the /U flag description, which controls the unique index. For Oracle Spatial tables, the spatial index is created on the geometry column and is called table_name_SX; for SpatialWare tables, the index is created on the column geometry column and is called hg table_name ind.

  Syntax: /I

- **/K Create Automated Key Column for SQL Server** – Use this flag to generate the unique key column values automatically in SQL Server. You do not need to fill the key manually when you insert a new row. The key column (SW_MEMBER) can be created with the IDENTITY property as an option.

  For SQL Server 2005, the key column is SW_MEMBER, because it needs SpatialWare. It also can be MI_SQL_REC_NUM depending on the spatial object type that is selected. If no SpatialWare is used (as with SQL Server versions 2008 or later), then the key column is MI_PRINX or MI_SQL_REC_NUM: MI_PRINX or SW_MEMBER is used for spatial types and MI_SQL_REC_NUM is used for XY types.

The key column (SW_MEMBER) is created with the IDENTITY property by default. Therefore, omitting the K option in the command line has the same action as specifying /K, (that is, it creates the key column with IDENTITY property). If you want to turn off the property, you must provide a keyword NO_IDENTITY following /K. See **Mixing Command Line Flags with the EasyLoader User Interface** on page 24.

Syntax: /K

Example: /K NO_IDENTITY

- **/L List of MapInfo tables** – Use this flag to specify a text file that contains a list of tables you want to upload. The format of each line is the same as the /T flag.

  Syntax: /L ListOfTables.txt

  Do not enclose file names in quotation marks.

- **/M MICODE/XY** – Use this flag to specify the object type to be used if it is SpatialWare. If the /M flag is used, provide MICODE (for XY with MapInfo key) or XY (for XY only) after /M. Any words other than MICODE or XY after /M are treated as errors, and EasyLoader does not run (the main EasyLoader dialog box does not appear). If you do not use the /M flag, EasyLoader uses SpatialWare as the default if the selected database has SpatialWare installed.

  Syntax: /M MICODE

  Example: /M X

- **/O Connection String** – Use this flag to set a connection string for Oracle Spatial to be passed to the program. See the /S flag for ODBC connections.

  Syntax: /O user_name/password@server_name

- **/P A | C | R** – Use this flag to specify what to do with the table(s) being loaded to the server.

  Use A to append to an existing server table.

  Use C to create a new server table. If you specify the C option and the table you are uploading has the same name as a table on the server, upload operation fails.

  Use R to replace an existing table.

  Syntax: /P A

- **/Q Quit** – Use this flag to exit EasyLoader when the upload is complete.

  Syntax: /Q

- **/R Replace the server table** – Use this flag to drop the server table and create and upload the new table. EasyLoader creates a table on the server even if the table did not exist previously.

  Syntax: /R

- **/S Connection String** – Use this flag to pass an ODBC connection string to the program. If enough information to connect is supplied, the ODBC connection dialog box does not appear. See the /O

flag for Oracle Spatial connections. The following examples illustrate the syntax of this flag. The first example uses a data source (DSN), the second supplies the full connection string.

```
/S DSN=MyDataSource
```

```
/S
UID=MyId;DATABASE=MyDB;HOST=MyServer;SERVER=MyServer_tli;SERVICE=sqlexec;PROTOCOL=onsoctcp
```

The following example shows the syntax using a data source (FileDSN) with a user ID (UID) and password (PWD).

```
/S FILEDSN=C:\Tenop\MyDataSource.dsn;UID=MyUserID;PWD=MyPassword
```

- **/T MapInfo Table Name;Server Table Name;Range** – Use this flag to pass a single table name to the program. Use a semicolon symbol as the separator between the MapInfo table name, the server name, and the range. The range is in the format starting number (COMMA) ending number. The server table name and the range are optional.

  Syntax: /T c:\data\states.tab;mystates;1,500

  Note: Do not enclose file names in quotation marks.

- **/U Do Not Create a Primary Index** – Use this flag to add a primary key constraint by default. This flag prevents a primary key from being created on the table. This flag is turned OFF by default, which means that a primary key is created by default. See /I which controls the spatial index. For Oracle Spatial tables, the primary key is created on the column MI_PRINX and is called table_name_PK. For SpatialWare tables, the primary key is created on the column SW_MEMBER and is called table_name_PK.

  Syntax: /U

- **/V Oracle Version** – Use this flag to load tables on an Oracle 8.1.6 server with the 8.1.5 format. This not generally recommended, but it is available if you have a special need to do this. If you want to accomplish this using the graphical interface, see **Mixing Command Line Flags with the EasyLoader User Interface** on page 24.

  Syntax: /V

- **/X Commit interval** – Use this flag to specify a commit interval. EasyLoader commits the inserted records when it reaches the commit interval you specify. The default commit interval is 1000. This same interval applies to the creation of the spatial index for Oracle Spatial. If the commit interval is set to 0 (zero), the whole range of records is inserted as a single transaction, before a commit is issued.

  Syntax: /X 500

- **/Y Style Column Name** – Use this flag to specify whether per-row styles are being loaded with the data. You can also specify the name of the column to be used. If you do not provide a name, the default MI_STYLE column name is created. If you specify the NO_STYLE keyword after the /Y flag, EasyLoader does not create a style column on the server table.

  Syntax: /Y [StyleColumnName | NO_STYLE]

- **/Z Always Geometry** – Use this flag to specify that the table is to be uploaded as a Geometry data type in Microsoft SQL Server Spatial, regardless of the coordinate system specified in the native table. For more information, see **Loading Microsoft SQL Server Spatial Data** on page 10.

  Syntax: /Z

Mixing Command Line Flags with the EasyLoader User Interface

Command line flags may be mixed with the EasyLoader user interface by using a Windows shortcut. This makes it easier to set flags as your default while being able to override them from the interface. These flags are only available from the command line.

- Create a shortcut to EasyLoader.
- Right-click and choose **Send To** and **Desktop** to create the shortcut on the desktop, or right-click and drag to move a shortcut to a folder of your choice.
- Right-click the shortcut and choose **Properties**.
- Under the **Shortcut** tab, within the **Target** edit box, add the appropriate command line flags to the end of the line, separated by spaces.

When EasyLoader is run from that shortcut, the specified flags will be in effect.

Creating a New Map Catalog

The Map Catalog stores the metadata used by MapInfo Pro and other MapInfo products to open tables with geometry data. You can create the Map Catalog without uploading a table at the same time.

Note: To create a Map Catalog successfully, you must have administrator privileges to the database server or the system administrator needs to grant you permission to create the Map Catalog.

There are two operations you can perform:

- If your database does not have a Map Catalog, you can use EasyLoader to create one.
- If your database has a Map Catalog already, you can delete entries in the catalog that are no longer valid.

To create a new Map Catalog and delete entries from an existing Map Catalog:

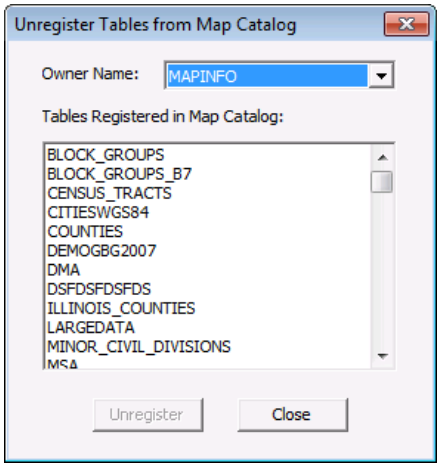
1. Open EasyLoader. The **EasyLoader** dialog box displays.

If the Map Catalog is present and there are no entries in it, the **Map Catalog** button does not enable.

2. In the **Connection Information** box, identify the connection you either create the Map Catalog for or delete Map Catalog entries from.
3. When you click the **Map Catalog** button, one of the following things happens:

   If there is no **Map Catalog** available for the current database, EasyLoader creates the MapInfo Owner and then creates the Map Catalog. This concludes the Map Catalog creation process.

   If there is a Map Catalog, use the **Unregister tables from Map Catalog** dialog box to delete obsolete tables from the list.



4. To identify the entries in the list you want to delete, select the data owner from the **Owner** drop-down list and click the table or tables you want to unregister from the Map Catalog list.
5. Click the **Unregister** button. Click **Close** when you have completed this process.

If you see an error when creating the Map Catalog (when you click the **Map Catalog** button), then you may not have the correct permissions to create a user and a Map Catalog table. When there is no `mapinfo` user, EasyLoader tries to create one using your credentials. An error occurs when your credentials do not have the correct permissions to create a user.

Have your Database Administrator do one of the following:

- Give you permission to create a user named `mapinfo` and create a table for the `mapinfo` schema.
- Create a user named `mapinfo` for you and give you permission to create a table for the `mapinfo` schema.
- Create the `mapinfo.mapinfo_mapcatalog` table for you and give you permission to write to it.

Using the MAPINFO_MAPCATALOG

The MAPINFO_MAPCATALOG is a registry table for databases that stores metadata about geometry tables in the database. Using the tablename and ownername as the key, the MAPINFO_MAPCATALOG identifies the geometry column, geometry type, projection, projection bounds, and table and feature level rendition information. The MAPINFO_MAPCATALOG is used by a number of MapInfo products, including MapInfo Pro, that access map data from databases.

If a Map Catalog does not exist, it can be created during the upload process when running EasyLoader. When using ODBC, EasyLoader will not issue public grants, which must be done by other means. If you do not have adequate permissions then creation will not succeed and the table will not be uploaded.

After the table is uploaded, an entry is made in the MAPINFO.MAPINFO_MAPCATALOG to represent that table. A separate entry is made for every table you upload.

If the table is made up of a single type of object, then the server object type is restricted to that type, otherwise the type is ALL. Also, the symbol clause generated is based on the server type. For example: After uploading the table 'States.tab' the server type will be X.2 (polygons), where X is a number that represents either IDS, SQL Server, or Oracle Spatial, and the symbol clause will have only the information for a polygon.

Note: See also, **Loading Microsoft SQL Server Spatial Data** on page 10.

MAPINFO_MAPCATALOG Table Structure

MAPINFO_MAPCATALOG has the following table structure:

```
SPATIALTYPE FLOAT
TABLENAME CHAR(32)
OWNERNAME CHAR(32)
SPATIALCOLUMN CHAR(32)
DB_X_LL FLOAT
DB_Y_LL FLOAT
DB_X_UR FLOAT
DB_Y_UR FLOAT
VIEW_X_LL FLOAT
VIEW_Y_LL FLOAT
VIEW_X_UR FLOAT
VIEW_Y_UR FLOAT
COORDINATESYSTEM CHAR(254)
SYMBOL CHAR(254)
XCOLUMNNAME CHAR(32)
YCOLUMNNAME CHAR(32)
RENDITIONTYPE INTEGER
RENDITIONCOLUMN CHAR(32)
```

```
RENDITIONTABLE CHAR(32)
NUMBER_ROWS INTEGER
```

The following script, which is shipped with EasyLoader, may be used to modify an existing MAPINFO.MAPINFO_MAPCATALOG to add the rendition columns if they do not exist. Run this script as user MAPINFO.

```
ALTER TABLE MAPINFO.MAPINFO_MAPCATALOG ADD RENDITIONTYPE INTEGER;
ALTER TABLE MAPINFO.MAPINFO_MAPCATALOG ADD RENDITIONCOLUMN CHAR(32);
ALTER TABLE MAPINFO.MAPINFO_MAPCATALOG ADD RENDITIONTABLE CHAR(32);
```

Notices

Index

3001 Summer Street
Stamford CT 06926-0700
www.pitneybowes.com