

# Loading Data into salesforce.com



## Overview

Data can be uploaded into salesforce.com from Excel using the sforce connector, an open source plugin for excel, available here:

<http://sforce.sourceforge.net/excel/downloads.htm>

This plug-in's functionality is accessible from a drop down menu via the Excel GUI, as well as from the VB editor in Excel. Note that salesforce.com support has not been trained on assisting with this process.

As with any database data load, there may be dependencies in place, such as parent-child relationships, on the objects into which data is to be loaded. There are in general 3 levels of complexity for data loads in salesforce.com:

Case I. No dependency loads, for example:

- Loading leads
- Loading accounts & contacts
- Loading independent custom objects

Case II. Simple dependency loads, for example

- Loading tasks related to objects
- Loading opportunities related to accounts and contacts
- Loading cases related to accounts and contacts
- Loading product line items related to products and opportunities

Case III. Complex dependency loads, for example

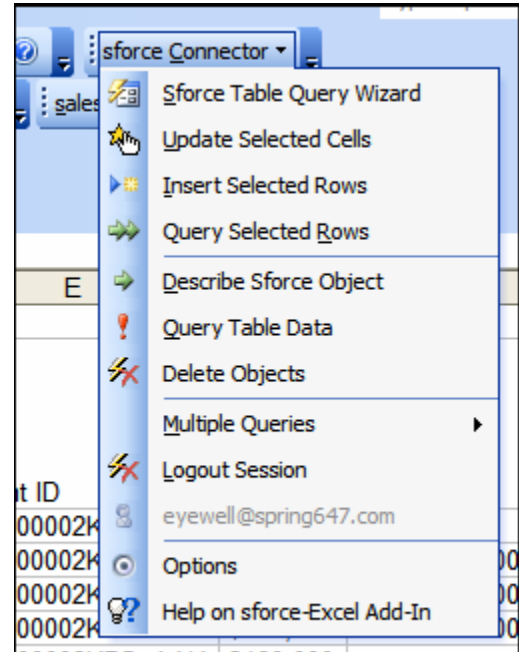
- Loading pricebooks

In Case I, data can be loaded using either the built-in import tools, the Bulk Data Loader, or the sforce connector using the standard data insert functionality: load the file with the values to be inserted, and run the import/insert function. There are no data dependencies, so the data can be completely loaded in one go, with no other references to data necessary.

In Case II, the records to be inserted are related to other data in the system (e.g.: opportunities have a relationship with contacts and accounts). These relationships are tracked in salesforce.com, not by putting the name of the parent object (the account) on the child object (the opportunity), but instead, by putting the *record ID* of the parent object on the child object. Here is an example, from data placed in an excel spreadsheet by an sforce connector query. Notice how the accountID value has nothing to do with a name of an account:

Opportunity ID	Close Date	Name	Stage	Account ID
00630000001BFxmAAG	3/3/2004	Dickenson - 15k	Closed	00130000002K7GdAAK
00630000001BFxrAAG	2/2/2004	UCSD - 90k	Closed	00130000002K7GeAAK
00630000001BFxwAAG	1/1/2004	United Partners - 45k	Closed	00130000002K7GgAAK
00630000001BFxkAAG	4/11/2004	Edge - SLA	Closed	00130000002K7GaAAK

**Figure 2: Relationships to other objects are represented by the object ID, not the object name**



**Figure 1: The sforce connector Excel toolbar menu**

# Loading Data into salesforce.com



This fact is what makes the 2<sup>nd</sup> case of imports a bit more complex. If we want to upload child data into salesforce.com, and at the same time define the parent ID for each opportunity, then we have to:

1. lookup the parent ID based on the parent object name,
2. feed this parent ID value into the parent ID column in the Excel spreadsheet,
3. upload the child data

In Case III, data must be loaded into certain tables before others. In the case of Pricebooks, there are 3 tables that, together, make up the data viewable in each pricebook. These tables are:

1. Product2
2. Pricebook2
3. PricebookEntry

In this case, the tables product2 and pricebook2 need to be populated, before PricebookEntry can be populated (see below).

	Price Book Entry ID	Price Book ID	Product ID	List Price	Active	Use Star Price
3	01u300000003YXLAA2	01s300000000FTQAA2	01t300000001IXiAAI	\$1	FALSE	FA
4	01u300000003rjFAAQ	01s300000000FTQAA2	01t300000001qvjAAA	\$50	FALSE	FA
5	01u300000004SuYAAU	01s300000000FTQAA2	01t300000001zEvAAI	\$0	TRUE	FA

**Figure 3: The PricebookEntry table. Notice how each entry requires a PricebookID and ProductID**

# Loading Data into salesforce.com



## How to Manually Load Data

Excel has a function that can help us with this process: VLOOKUP. See the Appendix for a help on the VLOOKUP function. VLOOKUP and the sforce connector can be used together to populate the “complete” set of child data to upload: the existing child record data, plus the parent ID field. This process requires 2 excel worksheets.

1. **Parent names and IDs:** using the sforce connector, query the table with the parent records into one worksheet in a workbook. NOTE: The rows in this list need to be alphabetized by the parent ID values. This is *required* in order for the VLOOKUP function to work correctly

2	Account ID	Account Name
3	00130000002aMWsAAM	BancTec
4	00130000002fbzdAAA	Neogov
5	00130000002h5F9AAI	Highland Transportation Ltd.
6	00130000002K7G2ΔΔK	Edge Communications

Parent Names and IDs / Child data / Sheet3

**Figure 4: parent object reference data can be in one worksheet, with the child data to be completed in another worksheet**

2. **Child data:** after switching to a new worksheet, using the sforce connector, load the child record data into a second worksheet, with Parent record names in one column, and a blank column for the parent record IDs, which we will fill using the VLOOKUP function (See below). At this time, paste any new record data into this sheet, matching the worksheet columns with the data columns

Once this VLOOKUP has been used to completely populate the parent object ID data column on the Child data worksheet, then the child data can be uploaded into salesforce.com: using the sforce connector,

1. Enter “New” in the child record ID column,
2. Highlight the rows to upload,
3. Select the “insert selected rows” menu option of the sforce connector.
4. The data will be uploaded, and the relationships with the parent data will be defined.

	A	B	C	D	E	F
1	Opportunity	System Modstamp >		9/19/2005		
2	Opportunity ID	Close Date	Name	Stage	Account ID	Account Name
3	New	3/3/2004	Dickenson - 15k	Closed	001300000088FfLAAU	Dickenson
4	New	2/2/2004	UCSD - 90k	Closed		UCSD
5	New	1/1/2004	United Partners - 45k	Closed		United Partners
6	New	4/11/2004	Edge - SLA	Closed		Edge

**Figure 5: the VLOOKUP function in action: the account ID for "Dickenson" is found by looking up the account name from cell F3, in the reference tables on the “Account Names and IDs” worksheet.**

# Loading Data into salesforce.com



**Note:** The format of the VLOOKUP function is:

***VLOOKUP(lookup\_value,table\_array,col\_index\_num,range\_lookup)***

The 4<sup>th</sup> parameter, the range\_lookup, while optional, needs to be "FALSE" for the most accurate pattern matching. See the Appendix for more information on the VLOOKUP function.

## How to Enhance the Process

The process defined above works very well, and is good for technical users. However, if this will be a frequently used process, performed by non-technical people, the user interface could be made much simpler, and much more user friendly.

With a little bit of Visual Basic, a nice GUI front end can be tacked on to the front of this workbook to make it a very useful tool. The various functions used by the sforce connector are accessible from any VB code you would write (just add sforce-connector as a reference to your Excel VB project). Depending on the application, you may need to develop any of the following screens:

1. A Login Screen, so the user's salesforce.com login can be captured
2. An upload option screen, where the user is asked what type of data to upload. Behind the scenes, based on the selection, the appropriate salesforce.com tables will be queried for reference data, and a screen will be presented to the user where they can paste in the data to upload.
3. A parent object selection screen, if the upload will be assigning a series of data records to a single parent
4. Any others you may need

# Loading Data into salesforce.com



## Appendix A: VLOOKUP

(copied for reference from MS Excel Help)

Searches for a value in the leftmost column of a table, and then returns a value in the same row from a column you specify in the table. Use VLOOKUP instead of HLOOKUP when your comparison values are located in a column to the left of the data you want to find.

The V in VLOOKUP stands for "Vertical."

### Syntax

**VLOOKUP**(lookup\_value,table\_array,col\_index\_num,range\_lookup)

**Lookup\_value** is the value to be found in the first column of the array. Lookup\_value can be a value, a reference, or a text string.

**Table\_array** is the table of information in which data is looked up. Use a reference to a range or a range name, such as Database or List.

- If range\_lookup is TRUE, the values in the first column of table\_array must be placed in ascending order: ..., -2, -1, 0, 1, 2, ..., A-Z, FALSE, TRUE; otherwise VLOOKUP may not give the correct value. If range\_lookup is FALSE, table\_array does not need to be sorted.
- You can put the values in ascending order by choosing the **Sort** command from the **Data** menu and selecting **Ascending**.
- The values in the first column of table\_array can be text, numbers, or logical values.
- Uppercase and lowercase text are equivalent.

**Col\_index\_num** is the column number in table\_array from which the matching value must be returned. A col\_index\_num of 1 returns the value in the first column in table\_array; a col\_index\_num of 2 returns the value in the second column in table\_array, and so on. If col\_index\_num is less than 1, VLOOKUP returns the #VALUE! error value; if col\_index\_num is greater than the number of columns in table\_array, VLOOKUP returns the #REF! error value.

**Range\_lookup** is a logical value that specifies whether you want VLOOKUP to find an exact match or an approximate match. If TRUE or omitted, an approximate match is returned. In other words, if an exact match is not found, the next largest value that is less than lookup\_value is returned. If FALSE, VLOOKUP will find an exact match. If one is not found, the error value #N/A is returned.

### Remarks

- If VLOOKUP can't find lookup\_value, and range\_lookup is TRUE, it uses the largest value that is less than or equal to lookup\_value.
- If lookup\_value is smaller than the smallest value in the first column of table\_array, VLOOKUP returns the #N/A error value.
- If VLOOKUP can't find lookup\_value, and range\_lookup is FALSE, VLOOKUP returns the #N/A value.

# Loading Data into salesforce.com



## Example

The example may be easier to understand if you copy it to a blank worksheet.

1. Create a blank workbook or worksheet.
2. Select the example in the Help topic. Do not select the row or column headers

Selecting an example from Help

3. Press CTRL+C.
4. In the worksheet, select cell A1, and press CTRL+V.
5. To switch between viewing the results and viewing the formulas that return the results, press CTRL+` (grave accent), or on the **Tools** menu, point to **Formula Auditing**, and then click **Formula Auditing Mode**.

The example uses values for air at 1 atm pressure.

	A	B	C
<b>1</b>	<b>Density</b>	<b>Viscosity</b>	<b>Temperature</b>
<b>2</b>	0.457	3.55	500
<b>3</b>	0.525	3.25	400
<b>4</b>	0.616	2.93	300
<b>5</b>	0.675	2.75	250
<b>6</b>	0.746	2.57	200
<b>7</b>	0.835	2.38	150
<b>8</b>	0.946	2.17	100
<b>9</b>	1.09	1.95	50
<b>10</b>	1.29	1.71	0

Formula	Description (Result)
=VLOOKUP(1,A2:C10,2)	Looks up 1 in column A, and returns the value from column B in the same row (2.17)
=VLOOKUP(1,A2:C10,3,TRUE)	Looks up 1 in column A, and returns the value from column C in the same row (100)
=VLOOKUP(.7,A2:C10,3,FALSE)	Looks up 0.746 in column A. Because there is no exact match in column A, an error is returned (#N/A)
=VLOOKUP(0.1,A2:C10,2,TRUE)	Looks up 0.1 in column A. Because 0.1 is less than the smallest value in column A, an error is returned (#N/A)
=VLOOKUP(2,A2:C10,2,TRUE)	Looks up 2 in column A, and returns the value from column B in the same row (1.71)