

Import Price – Birth of a New Loader

by red1

The Best Lessons In Life Are From Mistakes.
Just Do Not Repeat Them.

- red1

Why do we need an Import Price?

So that we can import the client's Price Lists in text format into the Database's PriceList Versions. Such a Price List won't use the Compiere Price Engine as the client's case may be:

- Having unique PriceLists figures such as ending in 88 cts.
- Not following any rules that the PriceList Setup can handle
- Already having present long winded PriceLists in foreign format and too tedious to be setup again in Compiere.

Basically you want to create an Import Price just like the other Imports – Products, Business Partners, and Accounts.

Since Compiere do not have a PriceList Import at the moment (up to version 251e), you can get to learn how to go about creating one, which will come in handy whenever you feel you need to do it again for any other table.

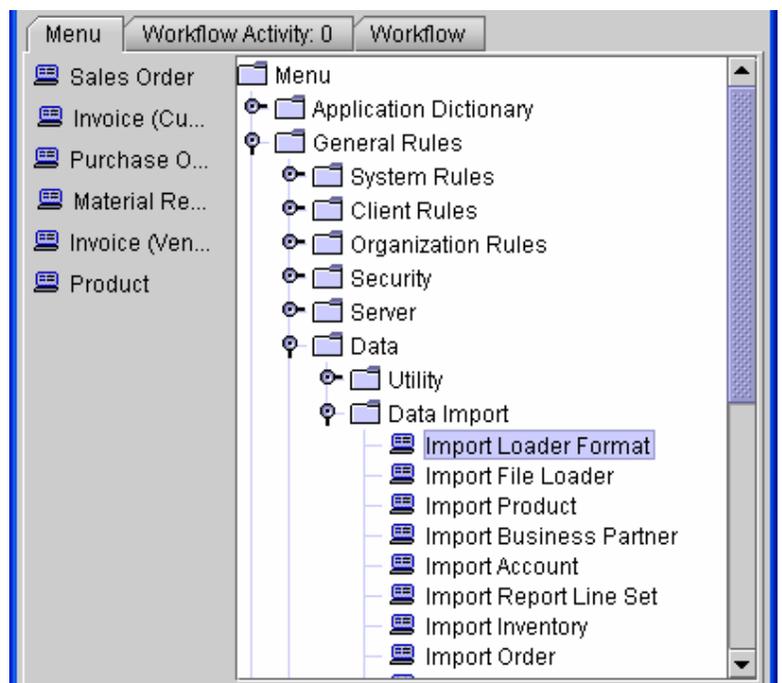
There are two items on the Compiere Menu that handles the Import Process as shown here:

- Import Loader Format
- Import File Loader

This Process basically takes the raw PriceList Data in text format and Load it into I_Price. Then a Java or SQL Process will bring it into the target PriceLists.

Now we shall begin.

If you open up the Import Loader Format, you will see that it does not have an Import Price. So we will need to create one.



NOTE: Its good if you can get hold of Compiere's User Manual that is sold on the Compiere's Webstore at www.compiere.org. In it, Chapter 17 on Utilities explains the definition and steps involved in the Import Process. Here we will only focus on the process of creating a new one. We will not overlap as this is weary work and redundant.

Creating new Table – I_Price

Open the Table and Column Window from the System Menu. You should log in as the System Administrator with System/System in the User-Id and Password.

The screenshot shows the 'Table and Column' configuration window. The 'Table' tab is selected. The 'Client' is set to 'System' and 'Organization' is marked with an asterisk. The 'DB Table Name' and 'Name' are both 'ImportPrice'. The 'Data Access Level' is 'Client only', 'Replication Type' is 'Local', and 'Entity Type' is 'User maintained'. The 'Active' checkbox is checked, while 'View', 'Maintain Change Log', and 'High Volume' are unchecked. The 'Records deleteable' checkbox is checked. A 'Create Columns from DB' button is visible at the bottom right. The status bar at the bottom left indicates 'Record saved' and the bottom right shows '[1/1]'.

Name it **ImportPrice**. Set the *Data Access Level* to Client only as its meant to be executed by the Admin of the Client and not by the Organisation or the System Administrator. Leave the *Entity-Type* as **User maintained** as you do not want your work to be overwritten when you upgrade your version through a migration later (another service that has to be purchased from Compiere and it requires an online connection when migrating).

You do not click on the *Create Columns from DB* as we do not have the DB Table, which is created later instead, by the *Synchronise* button at the next Tab.

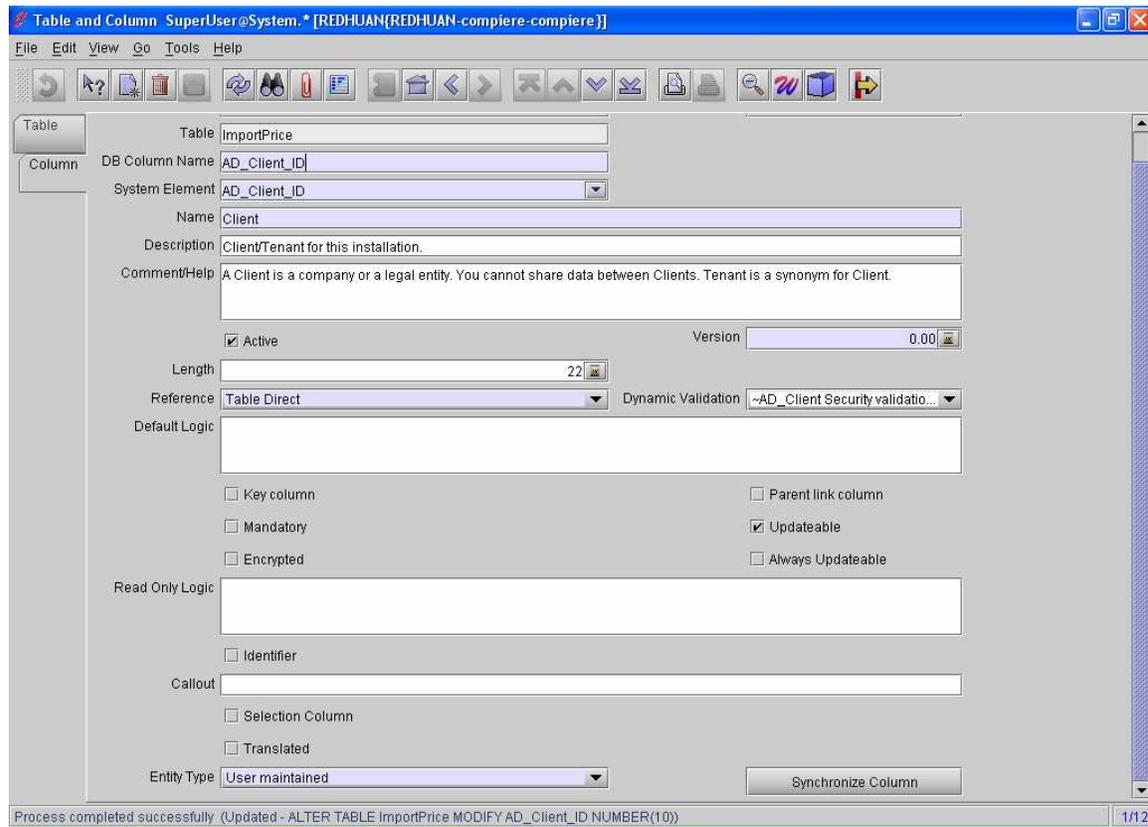
Now we go to the next Tab to populate the columns.

Creating The Columns

ization	Table	DB Column Name	System Element	Length	Name	Reference	Dynamic Validation
	ImportPrice	AD_Client_ID	AD_Client_ID	22	Client	Table Direct	~AD_Client Security validation~
	ImportPrice	AD_Org_ID	AD_Org_ID	22	Organization	Table Direct	AD_Org Security validation
	ImportPrice	Created	Created	7	Created	Date+Time	
	ImportPrice	CreatedBy	CreatedBy	22	Created By	Table	
	ImportPrice	I_IsImported	I_IsImported	1	Imported	Yes-No	
	ImportPrice	I_Price_ID	I_Price_ID	22	Import Price ID	ID	
	ImportPrice	IsActive	IsActive	1	Active	Yes-No	
	ImportPrice	PriceList	PriceList	22	List Price	Amount	
	ImportPrice	Processed	Processed	1	Processed	Yes-No	
	ImportPrice	ProductValue	ProductValue	40	Search Key	String	
	ImportPrice	Updated	Updated	7	Updated	Date+Time	
	ImportPrice	UpdatedBy	UpdatedBy	22	Updated By	Table	

You should create the columns as shown above. The **AD_Client_ID**, **AD_Org_ID**, **Created**, **CreatedBy**, **Updated**, **UpdatedBy**, **IsActive** fields are compulsory fields for Compiere database operations. You have to patiently create them. Take note of their *Reference* and other values as shown above. Our own fields are **PriceList** and **ProductValue**. These two fields are the foreign PriceList info that we are importing for. Your text file will thus has only those two which is what product? (ProductValue) has what price? (PriceList). The other fields – **Processed**, **I_IsImported** is for reference during the Import Process. The **I_Price_ID** is the Table's unique primary index key and you need not bother about it as it is handled by Compiere automatically. Just create it with the *Reference* set to **ID**. Otherwise Compiere wouldn't know.

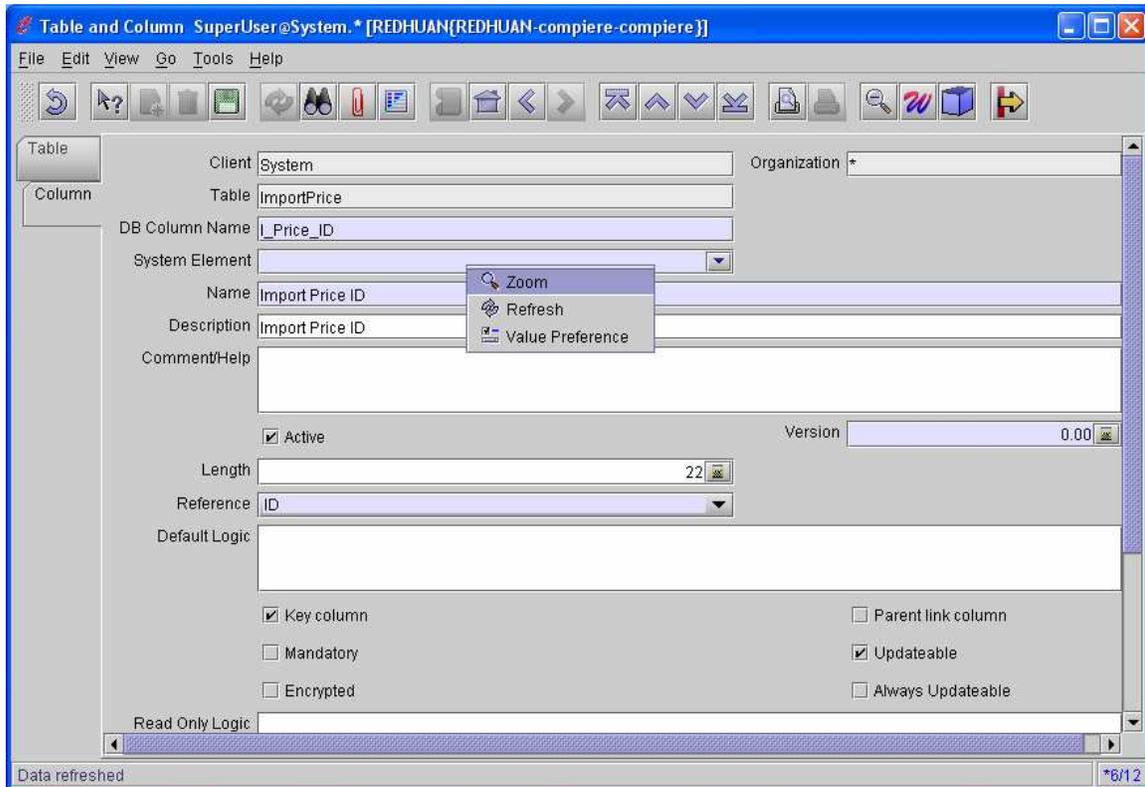
Next page is an example in single row view.



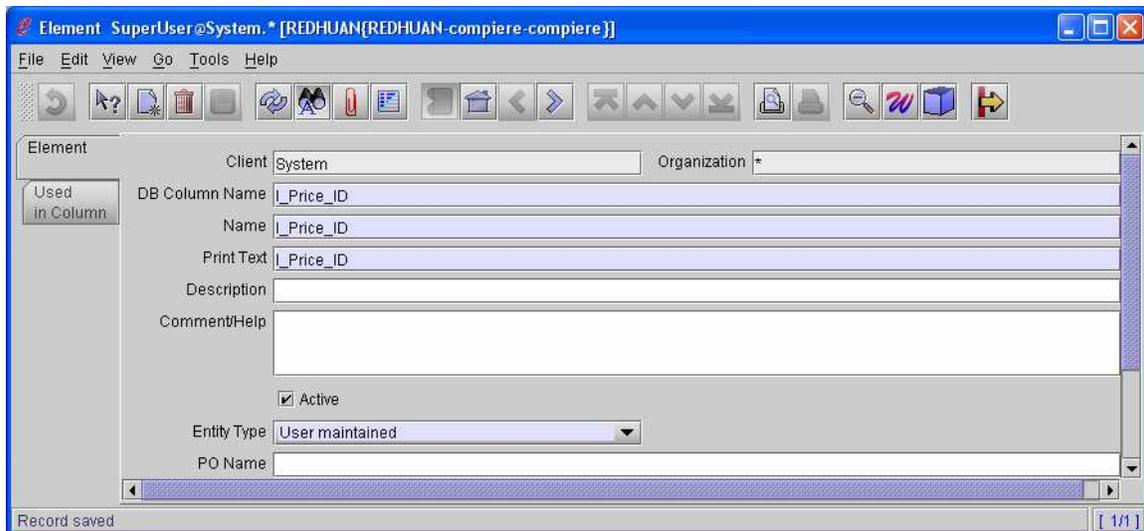
Again in review, take note of the Reference and Dynamic Validation needed for this **AD_Client_ID** field. Ensure that *System Elements* are filled in. They are for the persistency of the fields during integrative operations by Compiere such as during printing or query.

You can refer to other tables and see their settings for an idea how they need to be setup. Notice the *Synchronise Column*. This is the reverse of the *Create Columns from DB* earlier. You must now use this on all the columns created. Ensure that you get a successful reply at the bottom of the Window. You can see one now in the window above.

When you come to the **I_Price_ID**, you will have to create its own *System Element*. You can easily do that by right-clicking on the box and select **Zoom** to pull up the **Element Window**.

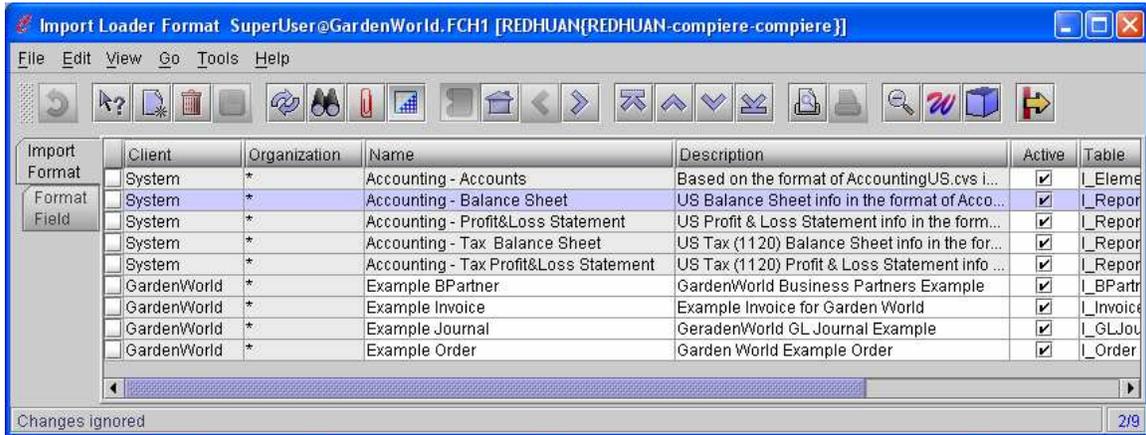


Here is the Element Window. Not much work here.

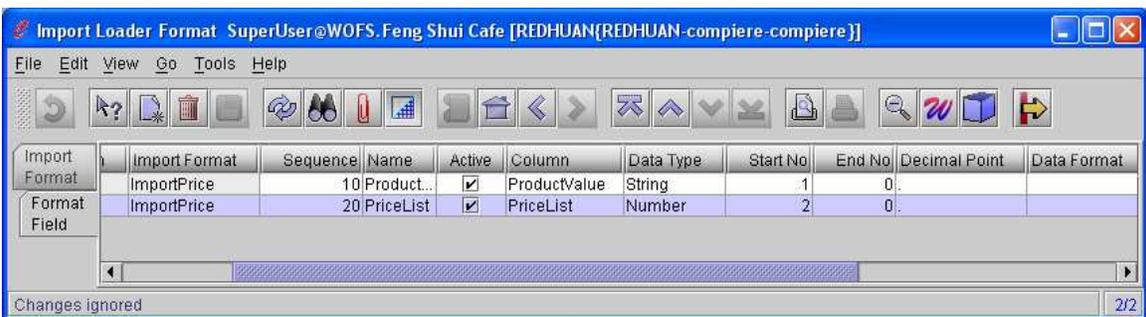
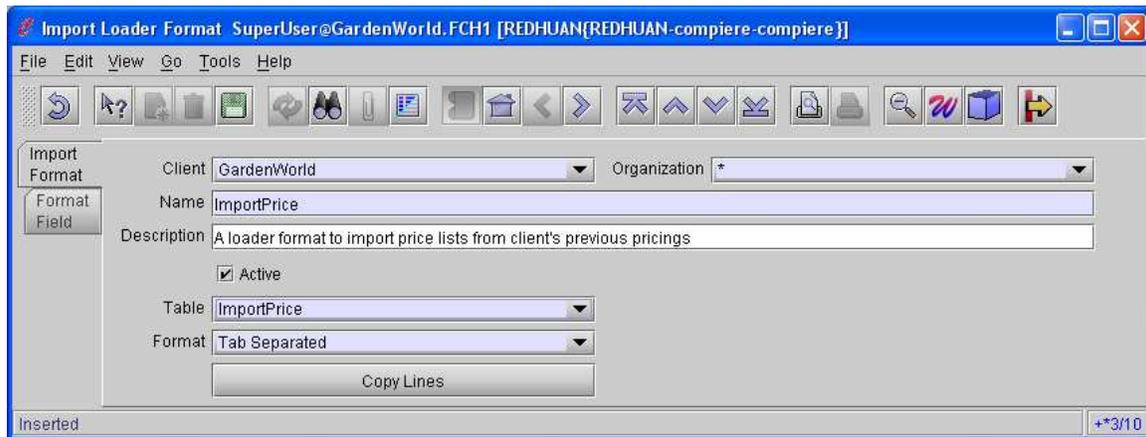


Return back to the **Column Window**. Check that its *Reference* is **ID**. Again remember to *Synchronise* or else the Table and Columns won't be created in the database.

Now we turn our attention to bringing the new Import Table into the Loader.

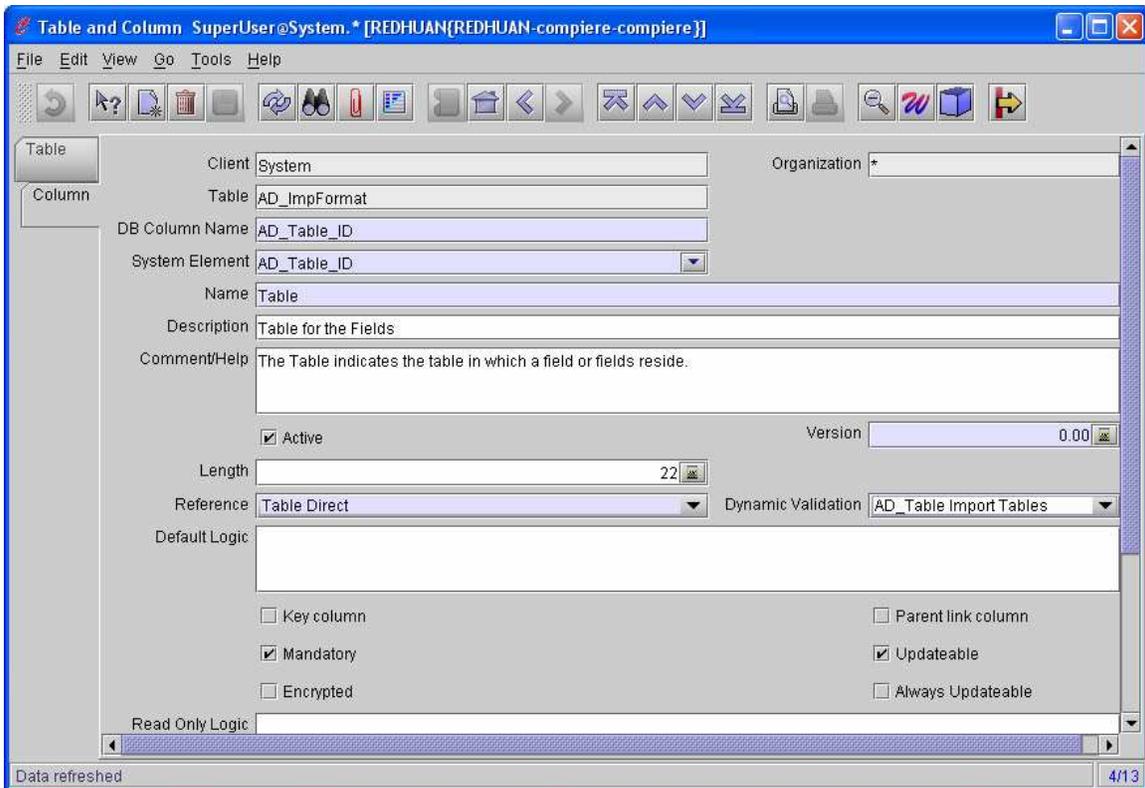


As you can see that the present Import Loader does not have the Import Price format so we will now create it in here. You may choose the format. Here it is Tab separated as our data has leading zeros which is harder to control in comma separated values.

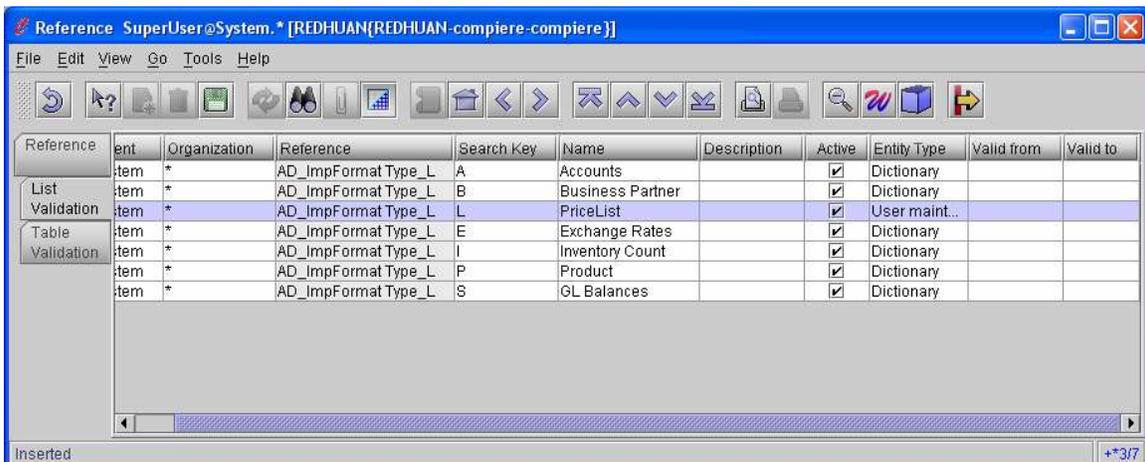


As explained prior, we create only two fields in this format. Note the Start No is important. Without them, your loading will not work. Keep them in sequence.

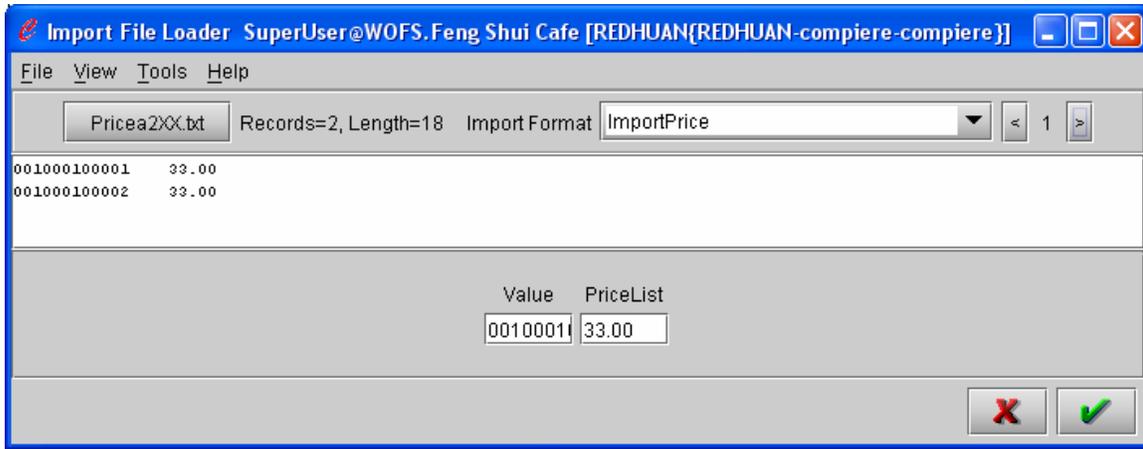
If you open up the Table and Column definition of the Import Loader, which defines the AD_ImpFormat table, you will see that the AD_Table_ID column is validated by AD_Table_Import Tables List.



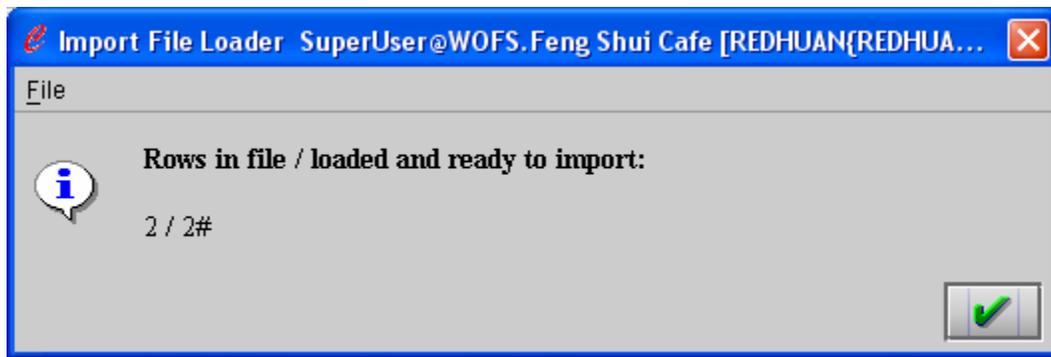
You can check that list under the Reference Window. You may create a new search key for Price List. I myself rush through this part, and aren't sure if this is needed and how it ties in. Perhaps you should skip this part first, do the next exercise, and carry on if things work out.



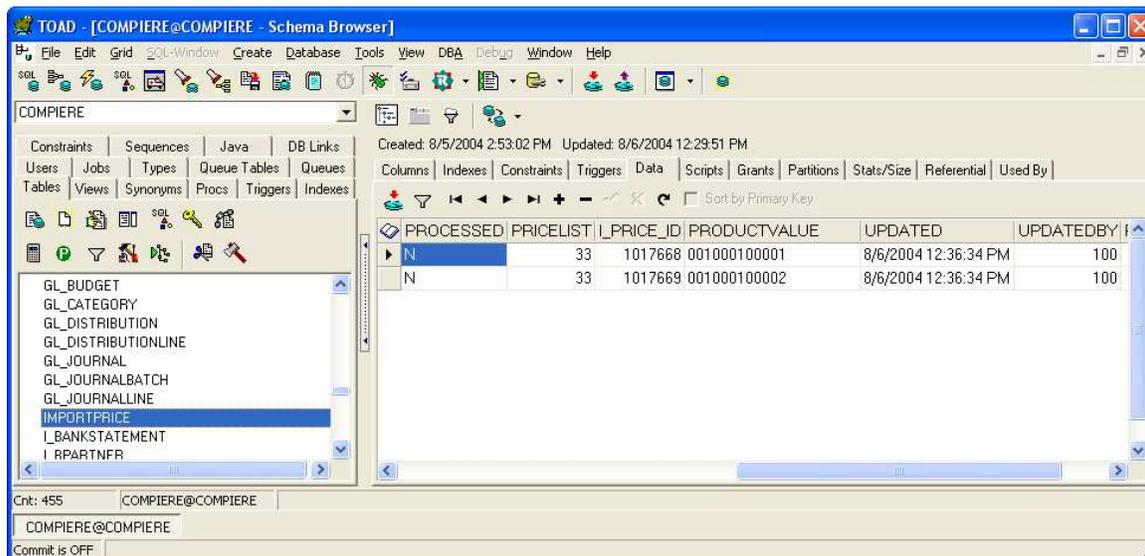
When we are ready to execute the Import Process into the I_Price table, we call the text source data. Here in the example below its Pricea2XX.txt which has only two test records. Then we select our Import Format which pulls from the Import Loader Format that we have created.



Pushing the process button, shall bring you to the next screen.



Success!! You can go into TOAD or Oracle EMC to examine the contents of your work.



Next will be the SQL Process to bring the I_Price data into the M_ProductPrice table according to the respective PriceListVersions.

Here are the two SQL Procedures that we used to achieve that. The first one populate the M_Product_ID for reference during the second process.

Create or Replace procedure SetProdID

AS

CURSOR PriceArray

IS

SELECT p.M_Product_ID, p.Value

FROM M_Product p

ORDER BY p.Value;

BEGIN

FOR r IN PriceArray LOOP

UPDATE ImportPrice SET Product_ID=r.M_Product_ID

WHERE ProductValue=r.Value;

DBMS_OUTPUT.PUT_LINE(' Processing ' || r.M_Product_ID);

END LOOP;

COMMIT;

END SetProdID;

This second procedure needs the PriceListVersion to be hardcoded. When tested successfully if needed to be a permanent feature in your Compiere application, you can incorporate them into a Java class and attached it to the ImportPrice window just like the other Import Windows. In my case, I didn't have the time and it isn't the priority in face of a long list of other work.

Create or Replace Procedure CopyPrice

```
AS
v_Mem1      NUMBER;
v_Mem2      NUMBER;

CURSOR PriceArray
IS
SELECT ip.Product_ID, ip.Price
FROM ImportPrice ip
ORDER BY ip.Product_ID;
BEGIN

FOR r IN PriceArray LOOP
v_Mem1:=r.Price;
v_Mem2:=r.Product_ID;
UPDATE M_ProductPrice SET PriceList=r.Price
      WHERE M_Product_ID=r.Product_ID
      AND M_PriceList_Version_ID=1000002 AND AD_Client_ID<>11;
--DBMS_OUTPUT.PUT_LINE(' Processing - ' ||v_Mem1 || v_Mem2);
      */
END LOOP;

COMMIT;
END CopyPrice;
```

That's all folks!

Please sign my guestbook to show any gratitude. <http://red1.org/guestbook> If you have money to spare, donate to any orphanage in your vicinity, in my name.

You can email me at red1@red1.org

If you aren't lazy like me, give me some challenge in the <http://red1.org/forum> but stick to the subject matter. I can't cover all the bases.

The Best Lessons In Life Are From Mistakes.
Just Do Not Repeat Them.

- red1

guess I just did! ☺