

DB2 Pricing DLL Instructions



Table of Contents

Purpose	3
Before you begin.....	3
Must be on current DB2 version:.....	3
Technical Documentation	3
Setting Pricing Parameters.....	3
Initialize/Test DLL:	4
Set Dealer Location Settings:.....	4
Set Customer Location Settings:.....	4
Loading and Saving Settings:	4
Testing the Pricing Functions	4
Pricing DLL Returned Data	4
Sell Price:.....	4
Price Type:.....	5
Discount:.....	5
Contract:	5
List Price:.....	5
Cost Price:.....	5
Error Code:	5
Error Description:	5
Error Message:.....	5
User Database Pricing DLL Stub Functions	5
CustomerPrice Stub Function:	6
[DealerLocation].....	6
[CustomerNumber]	6
[CustomerDepartment]	6
[CustomerLocation].....	6
[ItemNumber]	6
[ItemCompany].....	6
[ItemLocation]	6
[SellQuantity].....	7
ItemPrice Stub Function:	7
[ItemNumber]	7
[ItemCompany].....	7
[ItemLocation]	7
[SellQuantity].....	7
Sample Reports and Queries	7
OE – Price DLL Check	7
HI – What If.....	8
Common Problems	8
Additional Assistance	8

Purpose

These instructions detail the DB2 Pricing DLL, its purpose, how to use the functions in your user database, and how to include the functions in your reports and queries.

Before you begin

You must perform the User Database Update to copy the DB2 Pricing DLL functions to your user database. See the DB2 User Database Instructions on how to use this program.

Your DB2 databases must be current. Make sure you have fully populated all your relevant DB2 databases before trying to use the Pricing DLL. The Pricing DLL uses all your System Database, Customer Database, Item Database, and Contract Database.

Must be on current DB2 version:

Make sure you perform a server update before following these steps. Recent changes to the DB2 User Database and the maintenance programs are required for successful operation.

Technical Documentation

This Pricing DLL documentation is technical in nature and may be outside the reader's scope of expertise. We strongly recommend taking advantage of the many professional consultants available to assist in achieving your goals. Please contact us if you need to take advantage of the consulting services available.

Setting Pricing Parameters

Open your user database. Choose the Forms Button under the Access Objects Panel. You should see two forms: Startup and Pricing. If you do not see the Pricing form, then you need to perform the User Database Update. Double-click the Pricing form. The following screen will be shown:

The screenshot shows a software interface for setting pricing parameters. At the top, there are five buttons: "Load Settings", "Initialize/Test DLL", "Set Dealer Location Settings", "Set Customer Location Settings", and "Save Settings". Below these buttons are several rows of input fields:

- Dealer Location: Start Sale Contract: End Sale Contract:
- Customer No: Dept: Customer Location:
- Best Price Flag: Discount Type: Discount Percent:
- Contract 1: Contract 2: Contract 3: Contract 4:
- Cost Type: Margin Hold Flag: Margin Hold Percent:

At the bottom, there are fields for "Item Number:", "Company:", and "Quantity:", followed by a "Show Price" button. Below that is a row of fields for "Sell Price:", "Price Type:", "Discount:", "Contract:", "List Used:", and "Cost:".

Initialize/Test DLL:

First, click the Initialize/Test DLL Button. You should get a message box: DB2 Setup DLL Initialized. If you get an error message, then re-run the User Database Update making sure to select all objects to be transferred from the OPSoftware distribution database to your User Database.

Set Dealer Location Settings:

Enter a location code in the Dealer Location box (do not enter zero one (01) when one (1) is your primary location) and then click the Set Dealer Location Settings Button. The form should automatically fill in your Start Sale Contract and End Sale Contracts. These contracts are your global sale contracts and are used in determining customer pricing.

Set Customer Location Settings:

Enter a valid customer number in the Customer No box, a department in the Department Box, if any, a location code if different from the Dealer Location, and then click the Set Customer Location Settings Button. If the customer cannot be located, a message will appear, otherwise the Best Price Flag, Discount Type, and the other customer pricing setting boxes will automatically be updated with the selected customer settings.

Loading and Saving Settings:

Two additional buttons are provided for loading and saving your pricing parameters. Clicking the Load Settings Button will load the last saved settings into the Pricing Form. Clicking the Save Settings Button will save the current Pricing Form values and close the Pricing Form. Make sure you choose the Save Settings Button if you want to keep any changes you make in this screen.

Testing the Pricing Functions

Make sure you have entered and set your pricing parameters (Dealer Location Settings and Customer Location Settings). Also, make sure you have saved the settings (Save Settings Button) before testing the pricing functions.

Open the Pricing Form. Your saved settings should appear. Down below the Customer Pricing Parameters, there are three boxes you can use to test the pricing functions: Item Number, Company, and Quantity.

Enter an existing item number and company in the provided boxes. Important! The item number and company you enter must match exactly! Case does not matter. You can also enter a quantity if desired as some pricing is determined by the quantity ordered. Once a valid item number AND company have been entered, click the Show Price Button. The row below the item number and company boxes will contain the pricing information.

Pricing DLL Returned Data

When a call is made to the pricing DLL, several items are returned along with the actual selling price. Below is a complete listing of the different items returned from DLL calls.

Sell Price:

DLL name "PriceSell", this DLL returned item contains the actual sell price based on your pricing parameters. The price is for an individual item (unit price) and is extended to three decimal places.

Price Type:

DLL name “PriceType”, this DLL returned item contains the price type used when calculating the price. The price types are: “N” = Net price, no discount, “S” = Standard discounted price, “P” = Cost-plus (the discount field contains the cost plus percent used), “C” = Contract price used, and “*” = Sale contract price used.

Discount:

DLL name “PriceDiscount”, this DLL returned item contains the discount applied when calculating the price. In the case of cost plus (indicated by a “PriceType” of “P”), this field will contain the cost-plus percentage used.

Contract:

DLL name “PriceContract”, this DLL returned item contains the contract ID used when calculating the price. This item is only set when the price type is “C” for standard contract, or “*” for sale contract.

List Price:

DLL Name “PriceList”, this DLL returned item contains the list price used in calculating the sell price. Remember that depending on your parameter settings, a different list price can be used when calculating customer sell prices. This returned item reflects the actual list price used in the price calculations

Cost Price:

DLL Name “PriceCost”, this DLL returned item contains the cost price used in calculating the sell price. Also remember that depending on your parameter settings, a different cost price can be used when calculating customer sell prices. This returned item reflects the actual cost price used in the price calculations.

Error Code:

DLL Name “PriceErrCode”, this DLL returned item contains the error code, if any, generated during the price calculation. This item is for OPSoftware debugging purposes and not particularly useful to normal users.

Error Description:

DLL Name: “PriceErrDescription”, this DLL returned item contains the error description associated with any returned error code. This item is also primarily for OPSoftware debugging purposes.

Error Message:

DLL Name: “PriceErrMsg”, this DLL returned item contains a user message indicating why a price could not be calculated. Most common error messages are: “Item Not On File”, “Customer Not On File”, etc. You can use this returned item to check for and adjust any calls you make to the DLL.

User Database Pricing DLL Stub Functions

To simplify the process of using the Pricing DLL directly from your DB2 User Database, two stub functions are included in the Pricing code module of your User Database. Stub functions simply mean a

lead-in, or path, to a much more complicated process of calling a DLL from Access. You can use these two powerful stub functions directly in your Access queries as well as displaying the returned data.

CustomerPrice Stub Function:

The CustomerPrice function allows you to provide customer and item parameters in a single call to the DLL. This function is especially designed to provide prices when the customer number and item number is known. It can be used to check your open order database against current pricing for discrepancies. The point to remember before choosing to use this function is that both the customer number and item number must be available in your query before a price can be obtained. Below is the complete function call to access the CustomerPrice stub function followed by details of each parameter:

Function CustomerPrice([DealerLocation], [CustomerNumber], [CustomerDepartment], [CustomerLocation], [ItemNumber], [ItemCompany], [ItemLocation], [SellQuantity]) as Currency

[DealerLocation]

Optional, you can provide your primary store location code sometimes called GL location. If this parameter is not provided, the DLL will default to location 1.

[CustomerNumber]

Required, you must provide a valid customer number to use in calculating the price. The pricing parameters for the provided customer, discount, contracts, etc., will be used to determine the price.

[CustomerDepartment]

Required if you want to select a particular customer department, otherwise the DLL will default to the primary account (blank department).

[CustomerLocation]

Optional, you can provide a location code for the customer. Some customers can have different pricing based on location code. If provided, then the provided location code will be used. If this parameter is not provided, then the [DealerLocation] code will be used.

[ItemNumber]

Required, you must provide the item number you want to price. The item must exist in your database. Case does not matter, and dashes and slashes will not effect the matching.

[ItemCompany]

Required, you must provide the item company code, if the item has one, that you want the DLL to price. Case does not matter.

[ItemLocation]

Optional, you can provide a item location code to use when calculating the price. Some items may have different prices based on location. If you provide a location code, then that code will be used to calculate the price. If you do not provide an item location code, then the [DealerLocation] code will be used as the default.

[SellQuantity]

Optional, you can provide a selling quantity to use when calculating the price. Some items may have different prices depending upon the quantity ordered. If you provide a quantity, then it will be used to determine the price. If no quantity is provided, then a quantity of one will be used to calculate the price.

ItemPrice Stub Function:

The ItemPrice Stub function is designed to provide pricing when only an item number is available. This behavior makes this function especially suited to ‘what if’ calculations and pricing of item lists and bids. The ItemPrice function gets its missing customer parameters from the pricing form discussed earlier in this document. Before you use this function, make sure you set and save the pricing parameters using the Pricing form. You can then create queries or reports calling this function and change the pricing parameters using the form to generate different results. The key points to remember before choosing this function is that only an item number has to be available and the other pricing parameters are obtained from the saved settings in the Pricing Form. . Below is the complete function call to access the ItemPrice stub function followed by details of each parameter:

Function ItemPrice([ItemNumber], [ItemCompany], [ItemLocation], [SellQuantity]) as Currency

[ItemNumber]

Required, you must provide the item number you want to price. The item must exist in your database. Case does not matter, and dashes and slashes will not effect the matching.

[ItemCompany]

Required, you must provide the item company code, if the item has one, that you want the DLL to price. Case does not matter.

[ItemLocation]

Optional, you can provide a item location code to use when calculating the price. Some items may have different prices based on location. If you provide a location code, then that code will be used to calculate the price. If you do not provide an item location code, then location one (1) will be used as the default.

[SellQuantity]

Optional, you can provide a selling quantity to use when calculating the price. Some items may have different prices depending upon the quantity ordered. If you provide a quantity, then it will be used to determine the price. If no quantity is provided, then a quantity of one (1) will be used to calculate the price.

Sample Reports and Queries

There are two sample reports and queries, one each of both stub function calls, contained in the latest OPSsoftware distribution database.

OE – Price DLL Check

This query and report uses the CustomerPrice stub function to compare your open order database to the current customer price as calculated by the DLL. You can see the function call in the query and how it passes the required fields (the open order file contains both a customer number and item number, the requirements for using the CustomerPrice stub function) to retrieve the current price.

HI – What If

This query and report uses the ItemPrice stub function to compare existing sales history by customer to pricing parameters used in the Price form. Although the customer number is available in the history file, it is not used in this report. Instead, the ItemPrice function is used along with the pricing parameters in the Pricing Form to calculate a ‘what if’ price using the item number only.

Common Problems

The most common problem is a missing customer or item number. In the initial testing, using actual dealer databases, many items in the open order files were no longer good items and contained in the item file. The easy way to spot problem items is a returned price of negative one (-1). Since negative prices are not normal, we decided to use this indication of an error condition.

Additional Assistance

Due to the technical nature, and the general complexities of pricing on your system, we strongly recommend you take advantage of the many professional consultants available. Please contact us if you need to take advantage of these professional services.