
UniVerse Data Structures with Report Writer and Mass Load



© 2007 Activant Solutions, Inc. All rights reserved. Unauthorized reproduction is a violation of applicable law. Activant and the Activant logo, among others, are registered trademarks and/or registered service marks of Activant Solutions, Inc. in the United States and other countries. Eclipse is a trademark and/or a service mark of Activant Solutions, Inc. in the United States and other countries. Other parties' trademarks or service marks are the property of their respective owners and should be treated as such.

Table of Contents

UniVerse Data Structures with Report Writer and Mass Load	1
The UniVerse Database	7
What is a Database?	7
Frequently Asked Questions	8
Lesson 1	9
Accounts in Eclipse	9
Eclipse Database Accounts	10
Lesson 2	11
Eclipse Files	11
Valid Files in Eclipse	12
File Definition Maintenance	18
Maintenance Logging	19
File Definition Parameters	20
File Definition Maintenance for a Branch Specific File	22
Lesson 3	24
Eclipse File Structure	24
Eclipse File Structure	25
Attributes, Values, and Sub-Values	26
Viewing Raw Data in a File using TCL	28
LIST.ITEM	28
CT	28
DISP	28
SORT.ITEM	28
LIST	28
Dictionary Maintenance Summary Screen	31
Field Descriptions	33
Hot Keys	34
Example of Multi-Values in Dictionary Maintenance Summary	35
Example of Sub-Values in Dictionary Maintenance Summary	37
Branch Specific Data File Structure	40
PROD.BR File	40
PROD.CALC.BR File	43
Lesson 4	45
Selecting and Listing Data in TCL	45
Using the Select Retrieve Verbs	46
Retrieve Sentence Operators	47
Examples	47
Multiple Selects	48
Wild Cards	49
Saved Lists	51
Listing Saved Data	52
Sorting	55

Setting Common Data.....	56
SET.COMMON	56
Activating Upper/Lower Case Sensitivity	58
UL Command.....	58
Lesson 5.....	60
Creating a Basic Report in Report Writer	60
Report Writer	61
Designing Your Report.....	61
Painting Your Columns.....	63
Using the Files Option on Report Writer.....	66
Advanced Search for Dictionaries in Other Files	70
How does the System Know what Files you have Access to?.....	72
Path Option on Report Writer	73
Column Data	77
Ignore Branch Hierarchy.....	78
Break and Total.....	80
Formatting Data	82
Lesson 6	85
Selecting and Sorting Your Report	85
Report Writer/Mass Load Selection Screen.....	86
Compare To Column on the Select Screen	88
Comparing to a Dictionary Item	88
Comparing to a Text String.....	89
Comparing to a Null Value.....	89
Comparing to a User-Defined Prompt	90
Sorting the Report.....	94
Report Writer Advanced Selection Screen	95
Report Driver: Running the Report.....	96
Lesson 7	99
Report Writer Options	99
Report Writer Options Screen.....	100
Creating Mailing Labels	102
Lesson 8	107
Creating a Mass Load.....	107
Mass Load.....	108
Designing the Mass Load.....	110
Identifying the Data Type	112
Setting a Replacement Value	113
Leaving the Default/Set Value Field Blank.....	114
Replacing a Value with a Text String	115
Setting a Value to Null.....	118
Replacing a Value with another Dictionary ID Value	119
Replacing a Value with a Text String	120
Replacing a Value with a Concatenated Value.....	121
Replacing a Value with a Numerical Expression	122
Word Wrapping	123

Lesson 9	124
Dictionary Maintenance and I-Descriptors	124
Dictionary Maintenance	125
Dictionary Field Definitions	126
Hotkeys in Dictionary Maintenance Screen	129
Creating an I-Descriptor	131
Elements in an I-Descriptor Formula	132
Field Names	133
Operators	134
Testing I-Descriptors	135
Lesson 10	136
Creating Mathematical I-Descriptors	136
I-Descriptors that use a Mathematical Formula	137
Lesson 11	141
Creating Internal Variable I-Descriptors	141
Internal Variables	142
Using the @RECORD Variable	143
Using the @VM Variable	148
Lesson 12	150
Creating Field Command I-Descriptors	150
FIELD Function	151
Using the PSUB File	153
Lesson 13	156
Creating TRANS Command I-Descriptors	156
TRANS Function	157
Using the ORDER.QUEUE File	163
Using the PSUB File	164
Using a Double TRANS Command	165
Using Notes Files	166
Lesson 14	168
Basic Functions	168
Basic Functions	169
LEN Function	169
IF THEN ELSE Operators	170
Using IF THEN ELSE to Fix a Problem	172
Examples of other IF THEN ELSE Dictionaries	173
Character Strip	174
STR Function	177
TRIM Function	178
Concatenation	179
OCONV Function	181
DCOUNT Function	182
Lesson 15	183
Using Subroutines	183
Subroutines	184

Subroutine Dictionary Examples	194
Appendix A	200
Answers to I-Descriptor Exercises	200
Lesson 12	201
Lesson 13	201
Lesson 14	202
Appendix B	204
File Layouts for Release 7	204
Entity File Layout	205
Product Dynam File Layout.....	208
Product File Layout.....	209
Product Price File Layout	211
Searching for a Dictionary Item using TCL	211
Appendix C	212
File Layouts for Release 8	212
Entity File Layout Release 8.....	213
Product File Layout Release 8	220
AR File Layout Release 8	224
PSUB File Layout Release 8.....	227
Appendix D	228
Subroutines	228
Dictionary Subroutines.....	229

The UniVerse Database

What is a Database?

A database is a collection of related files. A file is a collection of related records. A record is a collection of fields (attributes). A unique identifier (ID) or Key identifies each record within the file. In multi-dimensional databases such as Universe, attributes can also consist of multi-values and values can consist of sub-values.

Files

Files are collections of logically related items or records. For example, a file cabinet contains folders, which in turn contain similar types of information. We all remember in grade school that everything we did (good and bad) was recorded in our record. Well, I'm sure they had a big filing cabinet filled with all of our records. At Eclipse, we have files about our customers, the products our customer sell, and our vendors just to name a few. Each of these files contains records made up of similar types of information. For example, at Eclipse we have about 400 customers so as you might guess there are about 400 records in our customer file, each containing similar types of information. Examples of that information would be the customer name, address, and phone number.

Records

A record is a collection of logically related attributes or fields. Therefore all of these 400 records in our customer file will contain a name of the customer, address, contacts, billing information, and other information for that ONE customer. Each record in the customer file is formatted in the exact same way. A record would be like a file within a section of the file cabinet. Going back to the grade school example, each "file" (one per student) would contain our grades, names, mom's name, and so on.

Attributes

An attribute is simply a dividing mark between each data element. When you access the data, you can quickly get to the field of data (attribute) you need. So the first attribute or field in the customer file might be name, the second; address, the third; contacts, etc.

Record ID / Key

Each record in a file must have its own unique identifier. This is called the key to the record. This key may be any combination of alphabetic, numeric, and most punctuation characters. At Eclipse we mainly use '.' and '~' as punctuation characters. No spaces are allowed in the key. Because of the uniqueness of the key, a programmer can pull data very easily from a database to be used in any program they write. The ID for our grade school records may have been our name or our social security number.

Frequently Asked Questions

Question:

Why doesn't Eclipse just use the old PICK systems approach and use one operating system and database combined?

Answer:

There are many drawbacks to this old model. The main one being that very few hardware platforms support the PICK operating system. The nice part about having a database like UniVerse is that most of the customization you would do to port onto different platform models is done by them. All databases and products that run on an OS (or hardware platform) must get "certified" by that manufacturer, which can take a lot of time and money. For Eclipse we just install the correct version of UniVerse on that OS and we're ready to go.

Question:

With all of the more popular databases (like Oracle) in the software industry, why did Eclipse choose a seemingly smaller and unknown database like UniVerse as its database provider?

Answer:

Oracle is a very high-end database platform with a very high price point. Oracle (and almost every other relational database including Access) is also limited to the 2-dimensional approach within a file and its record. The file contains records and the records contain data across the columns.

The UniVerse database is very powerful and can easily handle up to 5 and 6 dimensions within each record. Finally, the PICK database (and UniVerse in particular) is very easy to maintain and optimize. We can easily add fields to files (tables) without changing the whole file structure and our clients do not need a full-time Data Base Administrator to make sure the database is "tuned" and running properly.

Lesson 1

Accounts in Eclipse

Objectives

After you complete this lesson, you will understand:

- The Eclipse database accounts
- The Eclipse test/training account
- How to use the Eclipse University TrainS and TrainM accounts

Eclipse Database Accounts

The Eclipse database is called an account when it is installed on your system. In Eclipse you can set up multiple accounts. These are datasets used for different purposes.

Your main account is the “live” account. You may also have a play account set up for training purposes. Accounts are separate database occurrences with their own datasets. Transactions you enter in one account have no effect on any other Eclipse account.

At Eclipse University, we have set up two Eclipse accounts. As you log in to the “enterprise box” or “host,” you are prompted to pick one of the following accounts:

Account	is set up for a...
TrainS	single-branch company.
TrainM	multi-branch company.

When selecting an account, remember:

- The sales sources and terminal are unique for that account.
- Transactions, such as a sales order, will appear only in the account you are working in.
- A customer set up on one account, will not be added to any other account.

Exercise 1.1

1. Log into Eterm using the user ID assigned to you.

Your password at the Eclipse banner is the same as your Unix login ID.

How many account options are there when you log in?

2. Log into the TrainS account and access a customer record.
3. Log into the TrainM account and try accessing the same customer.

Lesson 2

Eclipse Files

Objectives

After you complete this lesson, you will understand:

- Eclipse static, dynamic and branch specific files
- The Eclipse files most commonly used for running reports

Valid Files in Eclipse

The files listed in the **Valid Files** Control Maintenance record are Eclipse files that can be used with the Report Writer and Mass Load programs. This list of files is maintained by Eclipse and should not be modified.

Static files contain data that does not frequently change. The static data files most commonly used in Report Writer and Mass Load include:

- ENTITY (contains Customer, Vendor, Branch, and Jobs/Ship-To information)
- PRODUCT (contains Stock, Non-stock, Special, and Catalog items)
- TAX.CODES
- BUY.LINE
- PRICE.LINE
- TERMS

Branch specific Files contain data that is branch and/or territory related and are linked to a static file. Report writer reports and Mass Loads are **not** created directly using these files. The parent file is used which will display dictionary items from these files.

- BUY.LINE.BR - Parent File is BUY.LINE
- ENTITY.BR – Parent File is ENTITY
- PRICE.LINE.BR – Parent File is PRICE.LINE
- PROCURE.GROUP.BR – Parent File is PROCURE.GROUP
- PROD.CALC.BR – Parent File is PRODUCT
- PROD.BR – Parent File is PRODUCT

Dynamic files contain transactional data and are continually changing. The dynamic data files most commonly used in Report Writer and Mass Load include:

- ORDER.QUEUE (contains *open* Ledger transactions)
- PRINT.QUEUE (contains Ledger transactions that are *queued* to print or need to be confirmed)
- AR
- PSUB
- CHECK.XREF

The PHYS File is dynamic and each time a Physical Control file is generated, a valid file is created in UniVerse. These files should be purged periodically to conserve disk space. The purge routine for this file is found on the Files Menu in the Merge/Purge option.

The four most utilized files for reporting purposes are:

- PRODUCT
- ENTITY
- AR
- PSUB

If you understand the file structure of these four files, you can utilize any other valid file in Eclipse with ease.

The following table lists many of the files used in the Eclipse system. We will discuss these files and the type of information that is stored in the file. This table is a good way to begin the process of getting to know your files.

File Name	Contains...
ENTITY	Name and address data for customer bill-tos, customer ship-tos, vendor pay-tos, vendor ship-froms, and branches.
PRODUCT	Product descriptions, groupings for the products, and pricing information. PROD.PRICE and PROD.DYNAM support this file.
AR	Summary information (header and totals, along with associated dates) for <i>closed</i> ledger transactions, such as sales invoices, received purchase orders, cash receipts postings, accounts payable invoices, and completed branch transfers. The ID to the file is the Ledger#.Invoice#. Example: S1012302.002. After transactions process, the summary information moves from the ORDER.QUEUE file to the AR file. Transaction details move to the PSUB file.
PSUB	Line item detail for <i>closed</i> ledger transactions, such as sales invoices, received purchase orders, transfers, and inventory adjustments. Each product on a transaction creates 1 record in this file. For example, a sales invoice with 5 products creates 5 PSUB records. When a transaction is processed, the detailed product information pertaining to the sales order moves from the ORDERS.QUEUE file to the PSUB file.
ENTITY.BR	This file contains branch specific data assigned to a customer/vendor. These overrides take place in Customer /Vendor file maintenance under the Additional Information, Branch Overrides option..
BUY.LINE	Purchasing-related information, such as Buyer, Target Factor, Target Value, Order Cycle Days, etc. The ID to the file is the Buy Line ID. Information from the Buy Line Maintenance Screen is stored in this file. Products are linked to this file when a buy line is assigned.
BUY.LINE.BR	This file contains the branch specific / Territory specific data for the Buy line.
CONTACT	Information entered on the Contact Maintenance screen.
COUNT.QUEUE	List of products to be counted for each branch. Products that have a negative on-hand, have been over-committed, or for which there was a manual backorder on a shipped ticket go to this queue.
EDICT	All Eclipse dictionary items for all files. The ID for records in the file is the Filename~DictionaryName. Example: PRODUCT~DESC.
ENTITY.LOG	Activity log information the System. This data is seen when accessing F2-S-V
ENTITY.PN.IDS	Customer / Vendor-Specific Part numbers. This file is populated only when customers or vendors are assigned a part number or if the Customer Product Demand Index from the AR/Utilities is run.
FAX.LOG	Activity information about faxes that are sent out using the Eclipse system.

File Name	Contains...
GENLED	G/L account and G/L report template records. This file does not contain G/L balances. The GENLED file has three indexes that can be used in selecting one type of record or the other: &INDEX& for G/L accounts only, &INDEX&.GRP for template groups only, and &INDEX&.ALL for all records.
GL.BUDGET	Information entered on the GL Budget Maintenance screen.
INITIALS	Information entered on the User Maintenance screen, such as user name, title, menu, message tune, keywords, etc.
LEDGER	All sales orders, cash receipts records, purchase orders, journal entries, accounts payable invoices, and branch transfers. This file contains both summary and detail line item information. The ID to the file is the sales order#, purchase order#, etc. This file is rarely used in Report Writer. Detailed information pertaining to shipped tickets should be retrieved from the PSUB file. The following ledger-index files are used in Report Writer: AR, ORDER.QUEUE, PRINT.QUEUE and PSUB.
LEDGER.LOG	All change logs associated with transactions.
MATRIX	All price matrix cells for the Sell Matrix and the Buy Matrix. It includes matrix cells for customer classes, as well as customer-specific prices. The ID to the file is BRANCH~Customer# or Class~Group or Product#~Effective Date. Examples: ~C2~GWIRE~10354 or 1~4993~1684~10354 Use this file to create a report that will provide a list of matrix cells that are about to expire.
MATRIX.DATES	This file works in conjunction with the MATRIX file and stores information such as expire dates, original and remaining quantities associated with the matrix record found in the matrix file.
MAINT.LOG	Audit change log information for any file for which maintenance logging has been turned on.
MENUS	All menus created in Eclipse.
MESSAGES	All messages sent and received, until the individual users delete them.
MISC.DATA	Items from various sources, including commission plans, product families and manifest information, quote maintenance, user's last run report, and customer-created product groups from Web Commerce. The ID is the Type Identifier~ID. Example: COMM~INSSLS.
ORDER.QUEUE	Open Ledger transactions, such as open (not printed) sales orders, purchase orders, and transfers. The ID to the file is the Ledger.ShipDate.GID. Example: S1012300.10865.1 (GID =Generation ID) Once the order is processed the information moves to the AR file and the detailed information moves to the PSUB file.
OVERRIDES.LOG	Manual overrides to sell and cost values on purchase orders, sales orders, or transfers, if the Log Sell Price, Purchase Price and Cost Overrides Control Maintenance record is set to yes. This file is used for reporting purposes only.
PRICE-GRP	Information entered on the Buy / Sell Group Maintenance screen.
PRICE.LINE	Data related to setting up price lines in Price Line Maintenance, including default units of measure, Basis Field names, and Basis assignments. Products are linked to this file when a price line is assigned to the product.

File Name	Contains...
PRICE.LINE.BR	Information that is branch specific/territory specific for the PRICE.LINE file are posted to this .br file.
PRINT.QUEUE	Ledger transactions that are <i>queued</i> to print. This file is updated from the Print Status prompt on the Status screen of SOE, POE, and TOE. A/P invoices (checks) are also in this file. The ID to this file is the Ledger#.GID. Example: S10112300.1 (GID = Generation ID).
PRINT.REVIEW	Transactions in the Open Order Status Review Queue.
PROCURE.GROUP	Information entered on the Procure Group Maintenance screen Products are linked to this file by the procure group entered on the product record or its buy line record.
PROCURE.GROUP.BR	Branch specific information for Procurement Groups is stored in this .BR file. Procurement group maintenance is found on the Maintenance menu in the purchasing menu.
PROD.BR	Branch specific / Territory specific information for the product file is stored in this .BR file
PROD.CALC.BR	Product branch specific/territory specific calculated data such as Average Cost, Product Demand etc are stored in this .BR file
PROD.DYNAM	Product on-hands, bin locations, average and last cost, and open orders for the product in each branch. Links to the Product file.
PROD.LIFO	All Product Lifo information, which is updated each time the Update routine for the Lifo is run. This file exists only if your company does "Lifo".
PROD.PRICE	Prices and costs, such as List Price and Replacement cost, for each product. The pricing information found on the Product Price Sheet Maintenance screen is stored in this file. The product's internal ID number is part of the key to this file.
PRODUCT.NOTES	Product notes entered using the Notes hot key in Product Maintenance. This file has the same key as the product file therefore they are linked together.
PRODUCT.RENTAL	Information entered on the Rental Product Maintenance screen.
QUAL.LOG	Information generated through the Unquality Event Tracking entries.
RECURRING.JE	Setup information for recurring journal entries.
REMINDER	Reminder notes created in Customer or Vendor Maintenance. These notes can also be created from the System Files Menu.
REPORTS	List of reports generated within Eclipse and sent to the Hold file. The detailed information regarding the reports is stored in the &HOLD& file. The date and time stamp is included for the purging routine.
SCHEDULES	Information entered in the Daily Scheduler.
SYSTEM.QUEUE	All scheduled phantom processes. This file is dynamic in that the key to each record changes each time the phantom processes an activity in the Phantom Status screen.
TAX.CODES	Information related to tax jurisdictions, such as tax rates and G/L posting information.

File Name	Contains...
TERMS	Customer and vendor-related terms entered on the Terms Maintenance screen. It includes information on discount percents, discount days, due dates, service charge percents, and service charge due dates. Customer maintenance, vendor maintenance and sales transactions link to the Terms file.
TERRITORY	Information entered on the Territory Maintenance screen.
TRACKING.LOG	Call tracking information for all entities and users in Eclipse. The key to this file is the Tracking ID number generated each time a tracker is created.
UD.WARRANTY	Warranty information entered on a user-defined screen when creating returns from your customers. This is a user-defined file, which contains limited information. You can enhance this file, based on how you handle warranties.
WHSE.OP.QUEUE	In-process RF transactions, until they are closed out.
WORK.MISC	Data from different sources, including saved Report Writer layouts, saved Mass Load layouts, Print Price Sheets, Manifest Information, Product Ranking, E-mail Standard Forms, Order Entry Clipboard (Ctrl-F5), Detailed Daily Schedules, Physical Generations, Cycle Count Generations (RDC), and F10 Quick Access information.
ZIP	Zip code information entered on the Zip Code Maintenance screen. Each Eclipse system contains all US 5-digit postal codes, including city and state information. This file also contains the tax jurisdiction codes.

File Definition Maintenance

Files are created and maintained in the File Definition Maintenance screen located on your System Files Menu. The system uses these parameters for indexing the file, sorting the file for display lists, and updating the file.

Only the system administrator should edit file definition parameters.

Note: This class does not discuss the creation of User Defined Files in Eclipse.

File Definition Maintenance				
File Name	:	PRODUCT		
Description	:	Product Master File		
Physical File	:	PRODUCT		
Parent File	:	PRODUCT		
Dictionary File:	PRODUCT	Type :		
Maint Logging	:	4 - Save Deleted Items		
Log Change Rsn	:	Y Min Days Before Purge :	730 Min # Logs to Save :	100
Select Index Dict ID	:	&INDEX&	Hot Sync (Y/N) :	
Select SortBy Dict ID:	LINE&SEQ	Prevent Mass Load (Y/N) :	N	
Disp Conv Expr/Attb	:	TPRODUCT;1;1;X		
Input Validation Subr:	VERF.PROD.ID			
Pre-Index Conv Subr	:	DICT.SOUNDA		
Update Validation	:			
Update Subr	:	UD.UPDATE.PRODUCT		
Select Filter Subr	:			
New ID Verification	:	No New		
Keep File in Sync With Parent (Y/N)	:	N		
Dict Maint	Delete	Branch Specific		F12-Absort

Maintenance Logging

Maintenance logging settings are used to collect changes that are made to files in the system. If a user changed the outside sales person in a customer's record you may want to know when the change was made, who made the change and what the value was prior to the change.

If a user is not prompted for their reason, chances are the maintenance logging is not turned on or requiring a reason for the change.

Below is a table that describes the different prompts for maintenance logging:

Field	Description
Maintenance Logging	<p>Indicate whether the system should log changes made to records in this file, and what type of information to log. Press F10 and select one of the following:</p> <ul style="list-style-type: none"> • Disabled – No maintenance logging occurs. This is the default value. • 1-Update Only – The log records the date and time of the update. • 2-Attr Only – The log records the date and time of the update and the attribute that changed. • 3-Attr W/Old Values – The log records the date and time of the update, the attribute that changed, and the attribute's old value. • 4-Save Deleted Items – In addition to recording the information described in option 3, the system also stores deleted records in the log.
Log Change Reason	<p>Indicate whether the system should prompt you to enter the reason for change when you press Esc after making the change:</p> <ul style="list-style-type: none"> • Y – Displays the Reason for Change screen, which prompts you to enter a reason. • N – Does not prompt you to enter a reason for the change. This is the default value. <p>If maintenance logging for this screen is disabled, skip this field.</p>
Min Days Before Purge	Enter the number of days to keep a log message before the system can delete it.
Min # Logs to Save	Enter the minimum number of messages the system should keep, regardless of the number of days they have been in the log.

File Definition Parameters

The following table describes the different parameters that can be set in the file definition maintenance screen that deal with sorting, displaying and updating records within the file.

Field	Description
Select Index Dict ID	<p>Defines the dictionary that indexes this file.</p> <p>Enter the dictionary item to index the file. This is usually set to the name of the key field, which Eclipse calls &INDEX&.</p> <p>If the Parent File is ENTITY or PRODUCT, you get the indexing capabilities of that file. Otherwise, you get the indexing capabilities defined in the File Definition Maintenance screen. Leave this field blank to select on record IDs.</p>
Select SortBy Dict ID	<p>Defines how a menu table, which lists the results of a search, sorts before it displays.</p> <p>Leave this field blank to sort on record IDs. Otherwise, enter the dictionary item by which to sort the file.</p>
Disp Conv Expr/Attb	<p>Displays the conversion expression/attribute, which defines the text describing the items displayed in a menu table. Menu tables list the results of a search.</p> <p>Leave this field blank to use the record ID. Otherwise, enter the name of the file attribute to display or enter an expression using PICK correlative commands.</p>
Input Validation Subr	<p>Identifies the subroutine the system uses in validating input data. If defined, the subroutine in this field overrides the subroutine in the Index Dict ID field.</p>
Pre-Index Conv Subr	<p>Identifies a conversion subroutine, which the system uses to remove unnecessary characters in attributes before indexing them.</p> <p>The only pre-index conversion subroutine the system uses is DICT.SOUNDA.</p>
Update Validation	<p>Identifies the subroutine to call for validating data in a record within the file.</p>
Update Subr	<p>Identifies the subroutine to call for running another program whenever a user updates a record in this file.</p> <p>For example, you can have a subroutine that sends a message to the credit control manager whenever someone updates the credit controls file.</p> <p>Note: The subroutine must adhere to Eclipse standards to work properly.</p>
Select Filter	<p>Identifies the subroutine to use to filter the displayed items.</p>

Subr	
New ID Verification	<p>Indicates whether you can enter a new record using a user-defined screen. Select one of the following options:</p> <ul style="list-style-type: none"> • No New – You cannot enter a new record from a user-defined screen. This is the default value and the designation used for all standard system files. • Sequential – The system assigns a unique ID to a new record. • Free Form – The user can assign a unique ID to a new record.
Keep File in Sync with Parent (Y/N)	<p>Indicates whether to keep the Physical file and Parent file in sync.</p> <p>For example, when set to Y, when you delete a record from the Parent file, the system also deletes the record from the Physical file.</p> <p>The only time one would need to set this flag is if a new file is being created that is linked to a parent file and you want the two files to always be in sync with each other.</p>
Hot Sync (Y/N)	Indicates whether this file should sync with the Hot Swap Server.
Prevent Mass Load (Y/N)	<p>Indicates whether the system prevents users from mass loading information to this file. The default is N.</p> <p>Note: Before you can change the setting of this field, the file must be listed in the Valid Files control maintenance record.</p>

File Definition Maintenance for a Branch Specific File

Earlier in this lesson we discussed the fact that some files have “parent” files. The branch specific files are files that use the parent file model. The parent file is the file used for the reporting and mass loading of data because the dictionary file is pointing directly to the parent file. However the actual data is stored in the physical file described by the file name.

Note in the example below that the Prevent Mass Load is set to a Yes. This does not limit you from mass updating data to the file, all this means is that the mass load will be conducted through the parent file of PRODUCT.

File Definition Maintenance			
File Name	: PROD.BR		
Description	: Branch specific product information		
Physical File	: PROD.BR		
Parent File	: PRODUCT		
Dictionary File:	PRODUCT	Type :	AppData
Maint Logging	: 3 - Attr W/Old Value		
Log Change Rsn	: Y	Min Days Before Purge :	Min # Logs to Save :
Select Index Dict ID :		Hot Sync (Y/N) :	
Select SortBy Dict ID:		Prevent Mass Load (Y/N) :	Y
Disp Conv Expr/Attb :			
Input Validation Subr:			
Pre-Index Conv Subr :			
Update Validation :			
Update Subr :	UD.UPDATE.PROD.BR		
Select Filter Subr :			
New ID Verification :	No New		
Keep File in Sync With Parent (Y/N) :	Y		
Dict Maint	Delete	Branch Specific	F12-Absort

A very important element of the branch specific files is the Branch Specific option we see on this screen. Notice in the above figure it is highlighted. This is an indication to you that an attribute within the file is storing the branches and territories that contain data.



By accessing this option we see that the branches containing data in this file are stored on attribute 40. Later when we discuss attributes and dig deeper into the branch specific file set up this will make more sense. What is important now is that you know you have a place to go to find where the branch listing is stored within a branch specific file.

File Definition Maintenance			
File Name	: PROD.BR		
Description	: Branch specific product information		
Physical File	: PROD.BR		
Parent File	: PRODUCT		
Dictionary File:	PRODUCT	Type :	AppData
Maint Logging	: 3 - Attr W/Old Value		
Log Change Rsn	: Y	Min Days Before Purge :	Min # Logs to Save :
Select Index Dict I	Enter the branch attribute: 40	ot Sync (Y/N) :	
Select SortBy Dict		ss Load (Y/N) :	Y
Disp Conv Expr/Attb			
Input Validation Su			
Pre-Index Conv Subr			
Update Validation			
Update Subr	: UD.UPDATE.PROD.BR		
Select Filter Subr	:		
New ID Verification	: No New		
Keep File in Sync With Parent (Y/N)	: Y		
Dict Maint	Delete	Branch Specific	F12-Absort

This attribute is never changed by the end user (that's you). Any file that is a branch file will have the appropriate branch setting assigned here.

Lesson 3

Eclipse File Structure

Objectives

After you complete this lesson, you will be able to:

- Understand Eclipse files, records, and attributes
- Recognize multi-values and sub-values
- Use TCL to view the raw data in a file
- View the dictionary items defined for a file



Eclipse File Structure

The Eclipse file structure can be compared to a file cabinet.

Eclipse Account – A drawer in the file cabinet.

Files – Folders in the drawer. For example, there can be separate folders for customers, products, and invoices.

Records – Paper in the folders. For example, a customer folder contains a separate sheet of paper for each customer. Each record has a unique identifier, referred to as an @ID or KEY. It can be almost any combination of alphanumeric characters, with no spaces.

Attributes/Data Elements – The lines or fields of information on the paper. For example, in a customer record, the name, address, phone number and billing terms are attributes.

The record key is always stored in attribute 0.

Attributes can have multiple values and values can have multiple sub-values.

File Cabinet	Database File Structure	Eclipse Example
Drawer	Account	Eclipse software package
Folder	File	ENTITY
Page of information	Record	ABC Company's ID
Lines (fields) of information on each page	Attributes/Data Elements Multi-valued Sub-Valued	Address for ABC Company

Attributes, Values, and Sub-Values

Eclipse records are variable-length. In the raw data file, markers separate attributes, values, and sub-values. The following example lists some attributes in a customer record. In Eclipse, customer records are stored in the “ENTITY” file.

200	Attribute 0
0001 Goldberg Boyle Brogan & Shusman PC	Attribute 1
0002 Cherry Hill Corporate Center ² 535 Route 38, Suite 300	Attribute 2
0003 CHERRY HILL	Attribute 3
0004 NJ	Attribute 4
0005 08002	Attribute 5

Attribute 0 – The number 200 represents the @ID or key of the record. The record key is always stored in attribute 0.

Single-Value Attributes – Attribute 1 contains the customer name, which is a single string of information stored as one value. The information stored in an attribute number is consistent for every record in a file. In this example, the customer name is stored in attribute 1 for every customer in the Eclipse database.

Multi-Value Attributes – Attribute 2, the street address, is multi-valued. The ² between “Center” and “535” is a value marker that represents the separation between the first and second value.

Sub-Values – Values can contain sub-values. Sub-value markers (ⁿ) separate multiple sub-values. The following is an example of the credit information for a customer.

This data is stored in attribute/data element 0022 in the ENTITY file. This attribute shows the credit limit, past due limit, job limit, days past due, deposit on stock items and non-stock items.

Credit Limit :	20000.00	Deposit%, Stock Items: 10
PastDue Limit:	15000.00	Days: 10 Deposit%, Nonstocks :100
Job Limit :	999.00	Terms Code: 9S Override Inv Terms: N

The raw data would display the sub-value markers and value markers as follows.

0022 2000000ⁿ1500000ⁿ99900^z10ⁿ10ⁿ100

Sub-value

Value

There are commands we will learn to view the raw data as we see above.

It is possible for the system to have values that are sub-sub-valued (a sub-value of a sub-value). The system displays this information using the square root symbol.

We will discuss this type of data structure later in this class.

```

150 2696z z z-1z-1z zS~z7038000000z7038000000z2240000000z2240000000z z z0921904257z+
    2√1z z z z z z z z1z z z5~441√5~263√5~62z-704√224√-224z z z z z z z zSLS√COGS√INVTV
151 2664z z z-2z-2z zS~z3519000000z3519000000z1378000000z1378000000z z z0921904257z+
    2√1z z z z z z z z1z z z5~441√5~263√5~62z-704√276√-276z z z z z z z zSLS√COGS√INVTV
152 2638z z z-1z-1z zS~z3001000000z3001000000z1172000000z1172000000z z z0921904257z+
    2√1z z z z z z z z1z z z5~441√5~263√5~62z-300√117√-117z z z z z z z zSLS√COGS√INVTV
  
```

Viewing Raw Data in a File using TCL

Eclipse Terminal Control Language (TCL) commands enable you to view the raw data stored in files.

Report Writer/Mass Load enables you to take raw data, manipulate it, and display it for analysis and interpretation. To understand how you can use Report Writer, you must first understand the TCL commands and be able to view raw data.

Use the following TCL commands to display raw data:

Command	Function
LIST.ITEM	Lists all the records in the file showing the attribute numbers and raw data for the attributes that contain data.
CT	Copies a record to the terminal, showing all of the attributes, even those containing no data.
DISP	Displays a record, showing the dictionary IDs and raw data for the attributes that contain data.
SORT.ITEM	Sorts the records by their Key or @ID and then displays a record, showing the attribute numbers and raw data for the attributes that contain data.
LIST	By default, displays only the Key or @ID of the records within the file. You can also use this command to display record data on the screen or send data to the printer. This function is discussed in Lesson 4.

Rules for using these commands:

- The name of the file to which the record belongs must appear directly after the command.
- The internal ID of the record must appear after the file name when using the CT command.

For example, to view the raw data for the customer who has an internal ID or key of 100, enter the commands as follows:

LIST.ITEM ENTITY 100
or
CT ENTITY 100
or
DISP ENTITY 100

Exercise 3.1

In Product Maintenance, display a product.

Note the Product ID and record the following information as you see it on the screen:

- **Price Line**
- **Buy Line**
- **GL Account / Product Type**

Use a TCL command to view the raw data for this product.

Record the information as you see it on the screen and note the corresponding attribute numbers.

Dictionary Maintenance Summary Screen

The Eclipse dictionary is a tool you can use to identify and access the data stored in the Eclipse data files. User-defined dictionary items associate names and report formatting characteristics with each attribute/data element stored in the Eclipse data files. All dictionary items reside in the EDICT file.

The Report Writer/Mass Load program uses the Eclipse dictionary to display raw data in a format you can understand.

Use the Dictionary Maintenance Summary screen to display the dictionary items defined for any validated file in Eclipse.

Dictionary Maintenance Summary									
File Name :ENTITY									
Dict ID	Description	Type	Attr	Val	Subv	Just	Width	Conv	
@ID	@ID	D	0			L	35		
CUST_ID	Customer ID	D	0			R	10	MR	
VEND_ID	Vendor ID	D	0			R	10	MR	
\$NAME\$	Entity Name	D	1			L	35		
NAME	Entity Name	D	1			L	35		
ADDRESS	Entity Address	D	2			L	35		
ADDRESS2	Address Second Line	D	2	2		L	35		
CITY	City	D	3			L	20		
STATE	State	D	4			L	2		
\$ZIP_CODE\$	Zip Code	D	5			R	10		
ZIP_CODE	Zip Code	D	5			R	10		
COUNTRY	Country	D	6			L	15		
ENTITY_TYPE	Entity Type MV by position	D	7			L	1	MR0	
IS_BILL_TO	Is Bill To Customer	D	7	1		R	1	YN	
IS_SHIP_TO	Is Ship To Customer	D	7	2		R	1	YN	
IS_BR_ENTITY	Is Branch Entity	D	7	4		R	1	YN	
\$IS_PAY_TO\$	Is Pay To Vendor	D	7	5		R	1		
							4 of 396		
<input type="button" value="Edit Dict"/>	<input type="button" value="I-Desc"/>	<input type="button" value="EXpand Desc"/>	<input type="button" value="Sortby"/>	<input type="button" value="Print Listing"/>	<input type="button" value="Select"/>	<input type="button" value="View"/>			

In the example shown above, there are 396 dictionary items defined for the ENTITY file. The first 17 are displayed. The **Attr** column shows the attribute/data element number to which a dictionary item is assigned.

It is possible to create multiple dictionary items in Eclipse for the same attribute or field of information. Any dictionary item that is flagged as an Eclipse Dictionary will appear in view-only format.

To locate a dictionary item:

- You can use the **Sortby** hot key to sort the list by attribute number or Dict ID, and then scroll through the list.
- You can also use the **Select** hot key to filter the list. Enter selection criteria based on the dictionary ID, description, dictionary type and Archived Dictionaries.

Dictionary Selection Criteria	
Dict ID	:
Description Pattern Search	:
Dictionary Type	:
Show Archived Dictionaries	: N

Dictionary ID is used if you know the dictionary you want to select out of the list. This will clear the entire Dictionary Summary screen with the exception of the dictionary item you enter on this prompt.

Description Pattern Search is used if you do not know the dictionary id, but you have an idea of what the description is. This will search on any word within the description of the dictionary. Remember, the less you key in the more options you will get. The more you key in the less number of options will appear.

Dictionary type is used to show either D type dictionaries (Data Elements) or I type dictionaries (I-Descriptors).

Show Archived Dictionaries are only relevant if you were using the Eclipse software application during release 7. Release 8 and above are using new and improved data elements for reporting and mass loading. However, Eclipse never purged out the old dictionaries. Instead we archived them so that users could see what dictionary items they had created in release 7.

Most dictionaries (not pointing to a branch specific value from release 7) that are archived are good dictionaries and work fine. They are not however using the new standards we have in place in release 8.

Field Descriptions

The fields on the Dictionary Maintenance Summary screen provide the following information:

Field	Description
Dict ID	Dictionary ID assigned to the attribute. An attribute can have multiple dictionary IDs.
Description	Functional description of the dictionary item.
Prompt	The default column heading for the selected data when printed on a report and the prompt for this information if used in a Report Writer Select statement with a variable value. This field displays in place of the Description field when you press the View hot key.
Typ	Dictionary item type: <ul style="list-style-type: none"> • D (Field Definition) – Information stored within a record, also known as an attribute or field. • I (Interpretive Descriptor) – A symbolic field derived from manipulation of data or a formula.
Attr	Numerical position of the attribute (field) within a record.
Val	Numerical position of a value within a multi-valued field.
Subv	Numerical position of a sub-value within a multi-value.
Just	Indication whether the data in this dictionary item should be left justified or right justified on a display/report column. Typically, text is left justified and numbers are right justified. Dates should be right justified. Some numbers such as UPC and Zip codes are left justified so that a preceding zero does not get removed.
Width	Character width of the display/report column for the data.
Conv	Pick output conversion code that determines the display/report format of the data. Conversion codes can be found in this workbook in a later lesson.

Hot Keys

The hot keys on the Dictionary Maintenance Summary screen provide the following functionality:

Hot key	Lets you...
Edit Dict	Edit the dictionary item on the Eclipse Dictionary Maintenance screen.
I-Desc	Edit the I-Descriptor formula, if this item is an I-Descriptor.
Expand Desc	Displays an expanded view of the description field. The expanded description contains 3 lines that are 60 characters wide.
Sortby	Sorts the displayed dictionary items numerically by attribute number or alphabetically by Dict ID
Print Listing	Sends a copy of the currently displayed dictionary file summary to your Hold file or default printer.
Select	Displays only the dictionary items that match selection criteria based on the dictionary ID, description, and type.
View	Toggles between displaying the Description field or the Prompt field for each dictionary item.

Example of Multi-Values in Dictionary Maintenance Summary

To recreate the following example, display the Dictionary Maintenance Summary screen for the ENTITY file. Then use the **Select** hot key and enter **freight** as the description selection criteria.

The numbers displayed in the **Val** column for attribute 7 indicate the multi-valued position the data is being stored for the data element represented on the screen. Each value represents something different for attribute 7.

Dictionary Maintenance Summary View Only									
File Name : ENTITY	Dict ID	Description	Type	Attr	Val	Subv	Just	Width	Conv
ZIP_CODE		Zip Code	D	5			R	10	
COUNTRY		Country	D	6			L	15	
ENTITY_TYPE		Entity Type MV by position	D	7			L	1	MR0
IS_BILL_TO		Is Bill To Customer	D	7	1		R	1	YN
IS_SHIP_TO		Is Ship To Customer	D	7	2		R	1	YN
IS_BR_ENTITY		Is Branch Entity	D	7	4		R	1	YN
\$IS_PAY_TO\$		Is Pay To Vendor	D	7	5		R	1	
IS_PAY_TO		Is Pay To Vendor	D	7	5		R	1	
IS_SHIP_FROM		Is Ship From Vendor	D	7	6		R	1	YN
IS_BR_ACCOUNT		Is Branch Cash Account Cus	D	7	7		R	1	YN
IS_FREIGHT_VEND		Is Freight Vendor	D	7	8		R	1	YN
IS_PROSPECT		Is Prospect Customer	D	7	9		R	1	YN
IS_MFR		Is Manufacturer Vendor	D	7	10		R	1	
\$SORT_BY\$		Sorts ENTITY file	D	8			L	12	
SORT_BY		Sorts ENTITY file	D	8			L	12	
INDEX		Index (used for searching)	D	9			L	35	
\$BILL_TO\$		Bill To ID (for Ship Tos)	D	10			R	10	MR0
								27 of 396	
Edit Dict Desc Expand Desc Sortby Print Listing Select View									

When viewing a customer or vendor you may have noticed bullet points that appear to the right of the customer or vendor's name and address.

Customer Maintenance		11790
Customer/New : 11790		
Name : HARRY EKBLUM Address : 204 SURPLUS STREET City : DUXBURY Zip : 02332 ST:MA Country: Sort By: EKBHAR Bill : Index : HARRY EKBLUM		(*) Bill To (*) Job or Ship To () Branch () Branch Cash Acct () Prospect (N) PO/Release# Required () Auto-Delete
1. Contacts 2. 3. 4. 5. 6.	Phones 617-741-4576	Bal Fwd/Open Item: 0 B/O Status: X Dflt Out Salesperson: WALKEV Dflt In Salesperson : WALKEV Dflt ShipVia: Frt In Exempt (Y/N) : N Frt Out Exempt (Y/N): N

The raw data for the entity @ID 11790 appears as follows:

LIST.ITEM ENTITY 11790 12:18:04pm 03 Jun 2004 PAGE 1

```

11790
001 HARRY EKBLUM
002 204 SURPLUS STREET2
003 DUXBURY
004 MA
005 02332
007 121 ←
008 EKBHAR
009 HARRY EKBLUM
013 1
014 11790
016 z z
017 617-741-4576
022 n z n n
024 1n2n3n4n5n6n8z1n1n1n1n1n1n
038 1z
039
041 WALKEV
043 X
044 WALKEV
045 N

```

Note the value marker that separates the bill-to flag from the ship-to flag.

Attribute 7 contains two values separated by a value marker (²). For this dictionary item, a value of 1 indicates that the field is set to “*.” In this case, both the first and second values are set to “1.” This corresponds to the “*” settings on the Customer Maintenance screen.

Example of Sub-Values in Dictionary Maintenance Summary

The as discussed earlier, credit parameters of a customer are entered using the Credit option from the customer file maintenance screen. This information is stored not only as multi-valued, but sub-valued as well. This means that each value can have multiple values within.

For example the credit parameters for entity @ID 832 looks like the following:

Credit Control Parameters	
() Use Default	() Use Bill-To
() Always COD () COD when credit limit exceeded () Print approval required message on all shipping tickets () Print approval required message when credit limit exceeded () No Order Entry () No Order Entry when credit limit exceeded, unless authorized () No Order Entry, regardless of credit limit, unless authorized (*) No printing of shipping ticket when credit limit exceeded, unless auth () No printing of shipping ticket, regardless of credit limit, unless auth () Only customer's authorized personnel may place orders (*) Company checks accepted (*) Personal checks accepted () Apply credits to oldest buckets when aging account	
Credit Limit :	10000.00 Deposit%, Stock Items: 0
PastDue Limit:	1000.00 Days: 5 Deposit%, Nonstocks : 50
Job Limit :	500.00 Terms Code: N30 Override Inv Terms: N
Authorized Personnel	Default Credit Card Info Add'l Credit Data EFT

If we open the dictionary maintenance summary screen we can find that the credit limit we see on the screen is located on attribute number 22 in the first value and in the first sub value of value 1.

Dictionary Maintenance Summary View Only									
File Name :ENTITY	Dict ID	Description	Typ	Attr	Val	Subv	Just	Width	Conv
	WIP_UNIT_COSTS	Work In Process Unit Costs	D	21		4	R	15	MR2
	CREDIT_LIMIT	Customer credit limit	D	22	1	1	R	11	MR2
	JOB_LIMIT	Job Limit	D	22	1	3	R	11	MR2
	PAST_DUE_LIMIT	Customer Past Due Limit	D	22	1	2	R	11	MR2
	NS_DEPOSIT	Deposit amount required fo	D	22	2	3	R	3	MR0
	PAST_DUE_DAYS	Customer Past Due Days	D	22	2	1	R	3	MR0
	STOCK_DEPOSIT	Deposit amount required fo	D	22	2	2	R	3	MR0
	CREDIT	Credit Specifications	D	23			R	9	
	IS_COD	Is Cod Customer	D	23	1		R	1	YN
	IS_COD_CREDIT	Is Cod Customer when credi	D	23	2		R	1	YN
	REQ_APPROVAL	Print approval required me	D	23	3		R	1	YN
	REQ_APPROVAL_CR	Print approval required me	D	23	4		R	1	YN
	NO_OE	No Order Entry	D	23	5		R	1	
	NO_OE_CREDIT	No Order Entry when credit	D	23	6		R	1	
	NO_OE_AUTH	No Order Entry regardless	D	23	7		R	1	
	NO_PRINT_CREDIT	No printing of shipped tic	D	23	8		R	1	YN
	NO_PRINT_AUTH	No printing of shipped tic	D	23	9		R	1	YN
63 of 396									
Edit Dict	I-Desc	Expand Desc	Sortby	Print Listing	Select	View			

The raw data for entity @ID 832 appears as follows:

```

LIST.ITEM ENTITY 832 12:21:43pm 03 Jun 2004 PA
      832
001 MIKE LAMB, INC
002 HUMMOCK POND RD2P 0 BOX 299
003 NANTUCKET
004 MA
005 02554
006 USA
007 121
008 LAMMIK
009 MIKE LAMB, INC 7521 LAMMIK
013 6
014 832
016 2FAX2
017 508-228-06352
022 1000000n100000n5000025n0n50 ←
023 12222222212
024 1n2n3n4n5n6n821n1n1n1n1n1n
028 N30
031 121
033 MA
034 04-2440843

```

The separation of each value is represented by the superscript “2” and the separation of the values within each value is represented by the superscript letter “n”.

Exercise 3.2

1. From the Files / Eclipse Dictionary... menu, select **Dictionary Maintenance Summary** to display the Dictionary Maintenance Summary screen.
2. Display the attributes of the ENTITY file.
3. Find attributes that contain a:
 - Single string of data
 - _____
 - Multi-valued string of data
 - _____
 - Multi-valued and sub-valued string of data
 - _____
4. Using TCL, view an ENTITY record in its raw data format.

Branch Specific Data File Structure

Sometimes you may here the terminology “product master” or “customer master”. The “master” is used when we are referring to the main parent file. The branch specific file is linked to the parent or master file for you.

The PRODUCT file (parent/master) has two linked files:

- PROD.BR
- PROD.CALC.BR

The data elements for the branch specific files are actually used FROM the parent file. Therefore you would never create a report writer or mass load using the PROD.BR or the PROD.CALC.BR file. Instead you would use the PRODUCT master file.

PROD.BR File

The Dictionary Summary Screen for the PROD.BR file will display the name of the data elements for the data being stored in this file. This information is mostly static information, meaning that it is not calculated by the system.

Dictionary Maintenance Summary									
File Name :PROD.BR									
Dict ID	Description	Typ	Attr	Val	Subv	Just	Width	Conv	
MANUAL_MIN	Manual Minimum Inventory L	D	2			R	6		
MANUAL_MAX	Manual Maximum Inventory L	D	3			R	6		
BUY_PKG	Purchase Package Quantity	D	4			R	6		
TREND	Trend Percentage	D	5			R	3	MR0	
SERVICE_STOCK	Service Stock Quantity	D	6			R	5	MR0	
\$LOST_SLS_CTRL\$	Lost Sales Control%	D	7			R	4	MR0	
\$LOST_SLS_CTRL\$	Lost Sales Control%	D	7			R	4	MR0	
LOST_SALE	Lost Sales Percentage	D	7			R	4		
LOST_SLS %	Lost Sales Percentage	D	7			R	4	MR0	
BCKRD_TLRNC_QTY	Backorder Tolerance Quanti	D	8			R	6		
EXCP_SLS %	Exceptional Sales %	D	8	1		L	4		
FCAST.METH	Forecast Method	D	8	2		L	10		
FORECAST_METHOD	Forecast Method S = Standa	D	8	2		L	1		
BTQ	Backorder Tolerance Quanti	D	8	3		R	3	MR0	
EOQ %	EOQ %	D	9			R	5	MR0	
EOQ_PERCENT	Economic Order Quantity Ca	D	9			R	7	MD2	
EOQ_\$	EOQ \$	D	10			R	5	MR2	
								19 of 65	
Edit Dict	I-Desc	Expand Desc	Sortby	Print Listing	Select	View			

As you can see, data such as the product's Lost Sales Percentage, Exception Sales Percentage and EOQ information are some examples of the type of data stored in this file. These dictionary items are found in the PRODUCT file for use in Report Writer and Mass Load.

The system will have a main record for an item as well as branch specific records (if cached). Below is the main record for @ID 9111 in the PROD.BR file. In lesson 2 we looked at the file definition maintenance screen for the PROD.BR file and we found that the branches attribute is listed as attribute 40.

```

    9111
004 10
005 z
007 20z
008 50z"
009 28z
010 100z
015 z
017 z
020 z
023 KOH
024 KOH
025 0
027 1zz0
033 z
040 ALLzEPz1 ←
051 15z
056 13449z
058 0

```

This information only appears on the main record within the file. However by looking at the main record one can surmise that there will be 1 other record in this file for this item that contains the branch specific data if the records have been cached. This record will have an internal id or @ID of 9111*1.

What is a cached record?

In the branch specific files, the system will create a new record for branch specific data. This creation occurs when any branch data for the record is viewed. This can include looking up the inventory inquiry screen that will display branch specific data or it can include looking at branch specific data for an item in product file maintenance.

This happens automatically by the system and is done to increase system performance when viewing the branch data for items.

Below is the branch specific data for the record described and shown above. One thing to note is that the attribute 40 on the branch specific record is nonexistent. This record is a cached record and is created when a user views branch specific data for a product. This includes the inventory inquiry screen.

```

      9111*1
004 10
007 20
008 50
009 28
010 100
023 KOH
024 KOH
025 0
027 0
051 15
056 13449
058 0

```

This is the Branch 1 record for @ID 9111. Notice in the @ID there is an asterisk separating the product @ID from the Branch number.

Exercise 3.3

1. Find a product in Product File Maintenance.
2. Set the Lost Sales Percentage and Exceptional Sales Percentage for Branch 1.
3. Display the item on your screen in TCL using one of the commands you have learned.
4. Find the attribute/data elements your data is stored in.
5. Verify your findings in the Dictionary Summary Screen.

PROD.CALC.BR File

The system calculated data such as hits, demand etc for an item are stored in the PROD.CALC.BR file. Just as we saw in the PROD.BR file, this file will have a main record which shows all of the branches the calculated data has been updated to. Each branch will then have its own record if it is cached. The @ID will display with the product's @ID"" BRANCH.

The dictionary summary screen for this file shows:

Dictionary Maintenance Summary View Only									
File Name :PROD.CALC.BR									
Dict ID	Description	Typ	Attr	Val	Subv	Just	Width	Conv	
PROD_CALC_BR_ID	Internal Id for the Produc	D	0			L	13		
RANKS	Multiple value listing of	D	1			L	1		
LAST_CALC_DATE	Date the product was last	D	2			R	10	D4/	
BR_HITS	Number of Hits for the dem	D	3			R	5		
DEMAND_PER_DAY	Calculated item demand per	D	4			R	10	MR3	
LEAD_TIME_DAY	Purchasing lead time in da	D	5			R	3		
LOW_SALE_QTY	Lowest sales quantity duri	D	6			R	5		
EOQ_QTY	Economic Order Quantity fo	D	7			R	6		
DEMAND_PERIOD	Number of days in the dema	D	8			R	3		
RAW_HITS	Raw hits for the demand pe	D	9			R	6		
RAW_DEMAND	Raw demand for the demand	D	10			R	6		
DAYS_OUT	Number of days that the pr	D	11			R	8		
AVERAGE_COST	Average Cost	D	12			R	10	MR3	
LAST_COST	Last Cost	D	13			R	10	MR3	
LANDED_AVERAGE_	Landed Average Cost	D	14			R	10	MR3	
LANDED_COST	Last Landed Cost	D	15			R	10	MR3	
FRZN_AVERAGE_CO	LIFO frozen average cost	D	16			R	10	MR3	
								1 of 24	
Edit Dict	I-Desc	EXpand Desc	Sortby	Print Listing	Select	View			

Note: under no circumstances will a mass load occur against this file directly.

Below is an example of the calculated data in Branch 1 for an item. This record is a cached which is why the *1 appears as a suffix to the internal id of the product. If the branch cache records were to be deleted, this record would be removed from the system. When branch specific data is viewed for the item, the system will automatically rebuild the cache record.

```

      9111*1
001  A²A²A²E²E
002  8735
003  18
004  47
006  1²1
007  1
008  365
009  18²18
010  17
011  0
012  176487
013  180089
014  176487
015  180089
016  176487
017  180089
018  176487
019  180089
020  12376
021  2

```

Exercise 3.4

6. Find a product in Product File Maintenance.
7. View the ranks and the Demand Audit screen for Branch 1.
 - Ranks are found off of the pricing hotkey.
 - Demand Audits are found off of the Inventory hotkey.
8. Display the item on your screen in TCL using one of the commands you learned.
9. Find the attribute/data elements your data is stored in.
10. Verify your findings in the Dictionary Summary Screen.

Any file that is designated as a Branch Specific will have a branch attribute. As we discussed in Lesson 2 this attribute number can be located using the File Definition Maintenance screen.

Lesson 4

Selecting and Listing Data in TCL

Objectives

After you complete this lesson, you will be able to:

- Understand TCL retrieve verbs
- Create a saved list in TCL
- Create a basic report using a saved-list

Using the Select Retrieve Verbs

In the previous lesson you learned how to use the CT, LIST, LIST.ITEM, and DISP retrieve verbs to list the contents of a single or multiple records.

The following TCL verbs enable you to select groups of records within a file:

- **SELECT** – selected records are listed in random order.
- **SSELECT** – selected records are sorted by the @ID or key.

The same technique we learn here is also applied in Report Writer and Mass load.

Retrieve Sentence Structure

The following rules apply when using the SELECT and SSELECT retrieve verbs:

- The retrieve verb must be the first word, indicating the action you want to take.
- The file name must follow the retrieve verb, indicating the file you want to select.
- The selection criteria must follow the file name and use the following syntax:

WITH Attribute name Operator Reference

Reference can be another attribute name or a literal value within quotes.

The following table lists the available operators.

Note: If you don't put the **AND/OR** in between dictionaries, then you have an implied **OR** in the dictionary before the values.

Retrieve Sentence Operators

Operators are important when narrowing the number of records to the select list you want to obtain.

Use this Operator...	To select a record if the attribute value...
= EQ EQUAL	is equal to the reference value.
# NE NOT NO <> ><	is not equal to the reference value.
> GT AFTER GREATER	is greater than the reference value.
< LT BEFORE LESS	is less than the reference value.
>= GE	is greater than or equal to the reference value.
<= LE	is less than or equal to the reference value.

Examples

To list all the customers and vendors in the state of NJ:

```
;SELECT ENTITY WITH STATE = "NJ"
```

```
;SSELECT ENTITY WITH STATE = "NJ"
```

To list customers and vendors who do not exist in the state of NJ:

```
;SELECT ENTITY WITH STATE # "NJ"
```

Exercise 4.1

1. Using TCL, write a command to select a list of customers and vendors that have a ship via of OT OUR TRUCK.

Hint: Display the ENTITY file in the Dictionary Summary Maintenance screen and then use the **Select** hot key to find the ship via attribute.

Multiple Selects

There are two ways to select records that meet multiple conditions. For example, list all the customers and vendors in the state of NJ whose address is within a range of zip codes. You can write:

- One select statement that uses multiple selection criteria, as shown in the following example.

```
SELECT ENTITY WITH STATE = "NJ" AND WITH ZIP > "07635" AND WITH ZIP < "08055"  
47 record(s) selected to SELECT list #0.
```

- Successive select statements in which each uses one selection criteria. Each successive select statement only applies to the records selected by the previous statement.

```
;SELECT ENTITY WITH STATE = "NJ"  
213 record(s) selected to SELECT list #0.  
;SELECT ENTITY WITH ZIP > "07635" AND WITH ZIP < "08055"  
47 record(s) selected to SELECT list #0.
```


Exercise 4.2

1. Using TCL, select products that are stock items and belong to buy line COPFIT.

Wild Cards

Use wild cards to select information that begins with a value, ends with a value, or contains a value. Following are examples using different wild card formats. If you do a select on a file that contains “Bob,” “Rob,” “Ben”, and “Abe,” the results you get will differ according to the wild card format you use.

Beginning with a value.

“B]” This option selects “Bob.”

Ending with a value.

“[B” This option selects “Rob” and “Bob.”

Containing a value.

“[B]” This option selects “Bob,” “Ben,” “Rob” and “Abe.”

Place-holding for the value. Use place-holding to determine where the value is stored in the string. Use the following wild card to find every value that contains an E as the second letter.

“^E]” This option selects “Ben.”

Exercise 4.3

1. Using TCL, select all entities that reside in the state of NJ and have a ship via containing the word "TRUCK."

Saved Lists

Once you have selected records, you can save the list. You can utilize a saved list in the Report Writer / Mass Load Program, Print Price Sheet Program, or TCL program, or simply list the data directly to the TCL screen.

To **save** a selected list, use the SAVE-LIST command. For Example:

SAVE-LIST SHIPVIA

There are two rules that apply to this command.

- The list must be assigned a name for retrieval purposes and
- The name cannot contain spaces.

To **retrieve** a saved list, use the GET-LIST command. For Example:

GET-LIST SHIPVIA

Exercise 4.4

1. Save the list that you created. Use your initials as a prefix and SHIPVIA for the name.

Listing Saved Data

Listing data in TCL is limited, because you see the data in its raw form. Listing data using the Report Writer/Mass Load program is much better, because the raw data is converted to an understandable format. If you understand the file structure and TCL commands, then you will have a better understanding of what the Report Writer/Mass Load program can do for you.

To list data in TCL the command structure is as follows:

```
LIST filename attribute-name attribute-name attribute-name
```

The **LIST command** displays the record key and the raw data in each designated attribute.

For example, the following command lists the record key and the raw data stored in the attribute called STATUS for every record in the PRODUCT file.

```
LIST PRODUCT STATUS :
@ID..... Status
33523
35526          1
19502          2
39532          5
15496          1
25511          1
27514          1
11490          2
45541          5
47544          5
49547          5
55556          5
61565          5
63568          5
65571          5
41535          5
```

The product status in the raw form is different from what is displayed on the Product Maintenance screen. In the raw form it is a numeric value. On the Product Maintenance screen it is a word that describes the status. The following list shows the word that corresponds to each numeric value.

1 = Stock	4 = Delete	7 = Purge
2 = Non-Stock	5 = Review	8 = Temp
3 = MiscChrg	6 = Comment	9 = LotItem

This concept is as important for Mass Load as it is for Report Writer. Mass loading the word “Stock” as a product status will cause program errors. Data must be stored in its raw format. In this example, use the number 1, which represents “Stock” as the input value when mass loading.

The most common question asked is how do you know the definition of the raw data? Unless you know it, the only way to learn it is to test it. For example, change the status of a product, then view it in the raw form in TCL or test the dictionary item. We will discuss testing dictionary items in future lessons.

In the following example, a dictionary item has been created in the product file that is an I-Descriptor called STAT.DESC. This dictionary item displays the alpha description for the status.

The following command lists records in the PRODUCT file. For each record it shows the record key, the raw data stored in the attribute called STATUS, and the alpha representation of that status.

```
LIST PRODUCT STATUS STAT.DESC
@ID..... Status Stat.Desc.
33523
35526      1 STOCK
19502      2 NONSTOCK
39532      5 REVIEW
15496      1 STOCK
25511      1 STOCK
27514      1 STOCK
11490      2 NONSTOCK
```

There is no limit to how many dictionary items you can list using the LIST command. When the information won't fit horizontally across the screen, it displays vertically.

The following example shows a vertical listing of the customer's name, first contact, first contact's phone number, and customer type from the ENTITY file:

```
LIST ENTITY NAME CONTACT1 PHONE1 CUST.TYPE
Entity ID... 5080
Entity Name. FRED MAZZACCO
CONTACT1.... TELEPHONE
PHONE1..... 401-861-2026

Entity ID... 25396
Entity Name. Gayl Skorenki
CONTACT1.... Gayl Skorenk
              i
PHONE1..... 217-309-9847
Type..... PH
```

Exercise 4.5

1. Using TCL, retrieve your saved list.
2. List the following information:
 - Name
 - State
 - Ship Via

Sorting

There are two verbs used to sort a list of records:

- **BY** – sorts records in ascending order
- **BY-DSND** – sorts records in descending order

The dictionary item on which you want to base the sort must follow the sort verb.

Example 1

List the records in the PRODUCT file. Display the product description and price line. Sort the products by price line and then sort the products within each price line by their description.

```
;LIST PRODUCT DESC LINE BY LINE BY DESC
```

Example 2

List the records in the PRODUCT file. Display the product description, price line, and list price. Sort the list by price, with the most expensive items at the top and least expensive items at the bottom.

```
;LIST PRODUCT DESC LINE LIST BY-DSND LIST
```

Exercise 4.6

1. Using TCL, retrieve your saved list.
2. List your saved list, sorted in ascending order.
3. List your saved list, sorted in descending order.

Setting Common Data

Some dictionary items prompt you to enter data, such as a branch number or date, for the item to evaluate properly. In TCL, use the SET.COMMON command to enter data required by dictionary items that have a prompt.

For example, in a multi-branch environment a product can have different on-hand quantities in each branch. Each branch can also store its on-hand quantities in multiple bin locations. Before you write a TCL command using the ON.HAND dictionary item, you need to specify which branch and bin location the command applies to.

SET.COMMON

Enter the SET.COMMON command at the TCL prompt to display the following screen:

Common Data	
Enter Br/Tr/All	:
Enter Start Date	:
Enter End/As of Date	:
Discount Class	:
Product Location	:
Price Basis Name	:
Location Number	:
Multi Value Pos	:
Enter Br	:
Generic Prompt	:
Ignore Branch Hierarchy	: N

Enter data for the fields related to the dictionary items you plan to use in following TCL commands and press **Esc**. Any information you enter on this screen is utilized by future TCL commands, until you exit from the TCL program or change the settings.

Exercise 4.7

1. Using TCL, list products and display their on-hands without setting any common data.
2. Using TCL, set common data by entering a branch and as-of-date.
3. List the products and their on-hands again.

Activating Upper/Lower Case Sensitivity

Eclipse TCL is case-sensitive. The system is set up to convert the TCL statements you enter to upper case, because TCL commands must be upper case and Eclipse file names and dictionary items are stored in upper case.

When selecting records based on attribute data, however, you may want your TCL statement to be case-sensitive. For example, a select statement that gathers all entities with a name beginning with “Joe]” produces different results, depending on whether Upper/Lower Case sensitivity is active.

UL Command

Use the UL command to activate Upper/Lower Case sensitivity. This remains active until you exit the TCL screen.

Following are two examples. The first selection was performed without activating Upper/Lower Case sensitivity. The command selected 14 records. The second selection was executed after activating Upper/Lower Case sensitivity. This time the command selected just 2 records.

```
;SELECT ENTITY WITH NAME = "Joe]"
```

```
14 record(s) selected to SELECT list #0.
```

```
.  
Active List Cleared
```

```
;ul
```

```
** Upper/Lower Case Active **
```

```
;SELECT ENTITY WITH NAME = "Joe]"
```

```
2 record(s) selected to SELECT list #0.
```

In the first example, only those entities whose name began with an upper case “JOE” were selected.

In the second example, only those entities whose name began with an upper case “J” and lower case ‘oe’ were selected.

Note: The select function in Report Writer / Mass Load respects Upper/Lower Case sensitivity. This function will be discussed further in Lesson 6.

Exercise 4.8

1. Using TCL without Upper/Lower Case sensitivity activated, select products with the word “White” in the description.
2. Using TCL with Upper/Lower Case sensitivity activated, select products with the word “White” in the description.

Lesson 5

Creating a Basic Report in Report Writer

Objectives

After you complete this lesson, you will be able to:

- Use the basic Report Writer/Mass Load screens
- Create a basic report using existing I-Descriptors and attributes

Report Writer

Report Writer enables you to extract information from the Eclipse database and create customized reports. A report can be printed directly from Eclipse or downloaded to your PC, where another application can access and use the data. The Report Writer program can calculate subtotals and or perform other mathematical calculations on numeric data. You can also use Report Writer to print mailing or bin labels.

There are three major steps in creating a report:

- **Designing the Report Layout** – First you design the report layout, including the title, columns, headings and totals.
- **Selecting the Report Data** – After you have designed the report layout, you need to specify the criteria to be used for selecting data for the report.
- **Running the Report** – After you have designed the report layout and selected the data for the report, you need to process the data and create the report.

Designing Your Report

What do you want to use this report for? Use the answer to this question to determine the information to include in the report and how to display it.

Then you need to determine which file the information will come from. For example, if the report is a sales report, but you only want monthly and yearly sales, this information can be obtained from the ENTITY file. If you need product information pertaining to the sales, then the PSUB file is more appropriate.

Use the report writer view of the Report Writer/Mass Load Design screen to design a report layout. From the Reports menu, select **Report Writer** to display this view.

In the header portion of the screen you assign an ID to the design, designate the file from which the data for the report will be obtained, and enter the title to be printed at the top of each report page.

The Design ID is something that will be repurposed. This means that you can re-run the report again and again. The ID should be something that references what the report is about and should have a prefix of your initials so that you can always find the reports you created quickly and easily.

File Name: The hardest part of creating a report writer report is making sure that you are starting in the correct file. Use the spreadsheet provided to you in Lesson 2 to help you pick the right file.

The Title can be more detailed description. Remember that this will appear on the report itself when it is complete. Make the title user friendly so that when the end user looks at their report, they understand what the report is all about.

Helpful Hint: It is best to clearly write down or obtain a specification sheet from the requester of what they want to see on the report. Important questions to have answered are how to select, sort and subtotal the report and of course, what do they need to see. This will help you to determine what file you need to turn to and whether all of the information is available from that file.

Painting Your Columns

Once you have determined the file and the fields of information you want to display on your report, you need to paint your columns. The Report Writer/Mass Load program enables you to paint your columns with more flexibility than TCL's LIST command. You have the option of changing your column headings as well as your width and output.

Report Writer/Mass Load Design						
Design ID : SHIPVIA		Created : 08/20/04		By : Nicole Dursi		
File Name : ENTITY				Total Width : 106		
Title : NJ Customer's Ship Via Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER_ID	8	Cus ID	N	*	
2	NAME	35	Customer Name	N	*	
3	ZIP_CODE	10	Zip Code	N	*	
4*	CONTACTS	20	Contact	N	*	
5*	PHONE_NBRS	20	Phone#	N	*	
6	OUTSIDE_SALES	8	Out Slsp	N	*	
-						

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	7 of 7	Analyzer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG Files Path

Col:

The **Col** number represents each column on the report. In some case scenarios this column will show an asterisk next to a number. This is a clear indication to you that the data element chosen requires more information to be entered using the

Dict Item/Formula:

In the **Dict Item/Formula** field, enter the dictionary item for the field you want to display.

Use the **F10** function here to view a list of the valid dictionary items available within the file. You can also use the **Dict Sum** hot key to display the Dictionary Summary Maintenance screen for the file.

It is possible to create a formula from the results of two columns on the report. Let's say you had a report on the product file that displayed the item's on-hand quantity in column 3 and the item's replacement cost in column 4. You can have column 5 be a calculation of 3*4. This means multiply the results in column 3 by the results in column 4 and display it in column 5.

Report Writer/Mass Load Design						
Design ID : ND.PROD.VALUE			Created : 07/28/04 By : Nicole Dursi			
File Name : PRODUCT			Total Width : 80			
Title : Special Product Evaluation Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER.ID	10	Part No	N	*	
2*	DESC	35	Description	N	*	
3*	PREV.ONHAND	10	On-Hand	N	Y	MR0
4	REPCOST	9	Rep-Cost	N	Y	MR3
5	3*4	12	Ext Value	N	Y	MR2
						6 of 6
Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load
Set VAL	LoG	Files	Path			

Formulas can also be a bit more complicated than the simple math we did in the above example. You can divide (/), add (+), subtract (-) and multiply (*) as well as a combination.

In both previous examples, notice that the first column's dictionary item is called VIEWER_ID. Utilize the VIEWER_ID in such files as ENTITY and PRODUCT. This dictionary item converts the internal ID or "key" of the record, so that you can drill down into the record from your Hold File or Spooler.

When you enter a dictionary ID, the system populates the remaining columns with information such as width, column heading, and format from Dictionary Maintenance, and standard defaults. You can change this information to reflect your desired output, as discussed below.

Width

The **Width** is filled with the default defined for the dictionary item. You can change the width larger or smaller. However you want the column to be wide enough to view all of the information in the field.

Column Heading

The **Column Heading** comes from the **Prompt** defined for the dictionary item in Dictionary Maintenance. This will appear as the column heading at the top of the report. If necessary, change the heading to mixed case, so it has a better look and feel.

Using the Files Option on Report Writer

The files option will provide you a list of other files you can link to from the file your report writer began with.

An important rule to this hotkey is that you need to know the data that you want to pull back and the “key” that is going to grant you this access for the data you are looking to obtain.

The same rules that apply when creating an I-descriptor using the TRANS command exists here. (We will be discussing the creation of TRANS command dictionaries in a later lesson.)

If I want to pull back the Outside Salespersons full name from the INTITALS file when running a report from the ENTITY file, it would be silly for me to use the Inside Salesperson as the “key” to the INTITALS file. I would want to use the Outside sales person.

In the following example we have a report writer report that is being written off of the PRODUCT file.

Col	Dict	Item/Formula	Width	Column Heading	Brk	Tot	Format
1		VIEWER_ID	7	ID	N	*	
2*		DESC	35	Product Description	N	*	
3*		PREV.ONHAND	10	Onhand	N	Y	MR0
4		REPCOST	14	Rep-Cost	N	Y	MR3
5		3*4	12	Extended Cost	N	Y	MR2
		-					

Report Writer/Mass Load Design							
Design ID : ND.PROD.VALUE				Created : 09/22/04 By : Nicole Dursi			
File Name : PRODUCT				Total Width : 82			
Title : Special Product Evaluation Report							

6 of 6	Anal	zer
--------	------	-----

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	Files	Path
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	Log

The Files option when accessed provides a list of files we can obtain information from.

Report Writer/Mass Load Design							
Design ID : ND.PROD.VALUE		Created : 07/28/04		By : Nicole Dursi			
File Name : PRODUCT				Total Width : 80			
Title : Special Product Evaluation Report							
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format	
1	VIEWER.ID	10	Part No	N	*		
2*	DESC		Files links				
3*	PREV	SALES.BUDGET.GRP	- Sales Budget groups				
4	REPC	BUY.LINE	- Buy Line				
5	3*4	PROCURE.GROUP	- Procure Group				
		PRICE.LINE	- Price Line				
		GENLED	- G/L Account Data				
		PRODUCT.NOTES	- Product Notes File				
		PROD.DYNAM	- Product Dynamic Data				
				Pg 1 of 4			
6 of 6							
Rel Build	Adv Selection	Rel Dct	Rel Data	Cop	Rel	Rel	Rel
Run Rpt	Dict Sim	Label	Opt	Notes	View	Begin Load	Set Val
						Lo	Files
							Path

Once a file is selected the system will display the different dictionaries that will link you to the file you chose. This is the crucial part of the entire process.



In the figure below we can see that both BUY_LINE and SECONDARY_BLINE are options to choose from. If we are interested in seeing information pertaining to the buy line attached to the product we would choose buy-line. If however, we need to obtain information relative to a secondary buy line that may or may not be attached to the product, we would pick secondary buy line.

Report Writer/Mass Load Design							
Design ID : ND.PROD.VALUE		Created : 07/28/04		By : Nicole Dursi			
File Name : PRODUCT				Total Width : 80			
Title : Special Product Evaluation Report							
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format	
1	VIEWER.ID	10	Part No	N	*		
2*	DESC	Select a dictionary to link PRODUCT to BUY.LINE					
3*	PREV \$BUY_LINE\$		- Buy Line				
4	REPC BUY_LINE		- Buy Line				
5	3*4 SECONDARY_BLINES		- VM list of secondary buy-lines				
				Pg	1 of 1		
6 of 6							
Del Build	Adv Select	Ion	Del Dct	Del Data	Cop	Del	Idg
Run Rpt	Dict Sum	Label	Opt	Notes	View	Begin Load	Set Val
						Lo	Files
							Path

Once the link has been selected we can search for dictionary items from the file we just linked to.

Report Writer/Mass Load Design						
Design ID : ND.PROD.VALUE		Created : 07/28/04 By : Nicole Dursi				
File Name : PRODUCT		Total Width : 80				
Title : Special Product Evaluation Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER.ID	10	Part No	N	*	
2*	DESC	35	Description	N	*	
3*	PREV.ONHAND	10	On-Hand	N	Y	MR0
4	REPC					
5	3*4					

Dict Item to Search for in BUY.LINE: _

6 of 6
Anal

Run Rpt
Dict Sim
Label
Opt
Notes
View
Begin Load
Set Val
Lo
Files
Math

Any data element that exists in the file will become options. In this example here we will select the procurement group that is assigned to buy line for the items appearing on the report.

Report Writer/Mass Load Design						
Design ID : ND.PROD.VALUE		Created : 07/28/04 By : Nicole Dursi				
File Name : PRODUCT		Total Width : 80				
Title : Special Product Evaluation Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER.ID	10	Part No	N	*	
2*	DESC	35	Description	N	*	
3*	PREV.ONHAND	10	On-Hand	N	Y	MR0
4	REPC					
5	3*4					

BUY.LINE Archived Dictionary Items

- NET.HITS - Network Hits
- NONSTOCK - NONSTOCK
- ORDER.CYCLE - ORDER.CYCLE
- OVR.LT - OVR.LT
- PROCURE.DAYS - PROCURE.DAYS
- PROCURE.GRP - PROCURE GROU**
- PROMISE.DAYS - PROMISE.DAYS
- SAFETY.FACTOR - SAFETY.FACTOR
- SUGGEST.ALL - Suggest on all
- TAR.VALUE - TAR.VALUE

3 of 4

Run Rpt

After the dictionary is selected the system will display the dictionary differently from others being used on the report. The system will insert an exclamation point (!) as a prefix with the file name attached to it.

A tilde (~) will appear after the file name which separates the file you linked to, to the dictionary you are displaying on the report.

Report Writer/Mass Load Design						
Design ID : ND.PROD.VALUE		Created : 07/28/04 By : Nicole Dursi				
File Name : PRODUCT		Total Width : 91				
Title : Special Product Evaluation Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER.ID	10	Part No	N	*	
2*	DESC	35	Description	N	*	
3*	PREV.ONHAND	10	On-Hand	N	Y	MR0
4	REPCOST	9	Rep-Cost	N	Y	MR3
5	3*4	12	Ext Value	N	Y	MR2
6	!BUY.LINE~PROCURE.GRP	10	PROCURE GROU	N	*	
						6 of 6
Sel Build		Adv Selection		Ecl Dct	Col Data	CopY
Del		Hdg		AnalyZer		
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load
Set VAL	LoE	Files	Path			

Advanced Search for Dictionaries in Other Files

The files option above is great if you know the file that you want to go to but you are unclear as to the data elements from the file you want to display on your report. What if you knew the dictionary item from another file? Is there a short cut to get to the dictionary without using the Files option?

Using the forward slash (/) as a prefix to a dictionary item search, the system will be smart enough to display all of the dictionary items you can utilize from the file you are creating your report from.

In the following figure a report is created from the product file. Using the forward slash we will search for all data elements that can be used in the product file regardless of the file the dictionary item actually belongs. After we key in our search criteria we use the <enter> key to begin the process.

Report Writer/Mass Load Design						
Design ID : ND.PROD.VALUE		Created : 08/04/04 By : Nicole Dursi				
File Name : PRODUCT		Total Width : 98				
Title : Sepcial Product Evaluation Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER.ID	10	Part No	N	*	
2*	DESC	35	Description	N	*	
3*	PREV.ONHAND	10	On-Hand	N	Y	MR0
4	REPCOST	14	Rep-Cost	N	Y	MR3
5	3*4	12	Ext Value	N	Y	MR2
6	!BUY.LINE~PROCURE_GROU /desc	12	Procure Group	N	*	

Sel Build	Adv SelectIon	Ecl Dct	Col Data	CopY	Del	Hdg	7 of 7	AnalyZer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG Files Path

There are pages of options to choose from. The system will display the name of the dictionary item it encountered as well as the file for which it belongs.

Dictionary Items	
DESC	- PROD.DYNAM
ACCOUNT_DESC	- GENLED
DESC_GL_ID	- GENLED
INTERNAL_ACCT_DESC	- GENLED
SHORT_ACCOUNT_DESC	- GENLED
SUB_LEDGER_DESC	- GENLED
DESCRIPTION	- PRICE.LINE

Pg 4 of 6

Find the element you want to use and select it by positioning your cursor on the option and hit your <enter> key.

Next the system will ask you for the appropriate path you want to use to get to the dictionary item.

Select a file path
PRODUCT>PRICE.LINE
PRODUCT>ENTITY>PRICE.LINE
PRODUCT>TERRITORY>ENTITY>PRICE.LINE
PRODUCT>LEDGER>ENTITY>PRICE.LINE
PRODUCT>LEDGER>TERRITORY>ENTITY>PRICE.LINE
PRODUCT>LEDGER>PRICE-GRP>PRICE.LINE
PRODUCT>LEDGER>PRICE-GRP>MATRIX>ENTITY>PRICE.LINE

Pg 1 of 1

As stipulated earlier, picking the path is the most important part of this process. Pay careful attention to how you want to get to the file and what will be returned in the column with the path you chose.

Lastly you will be prompted to choose the key that will give you access to the file. This process is the same as using the files option from the report writer/mass load screen.

Select a dictionary to link PRODUCT to PRICE.LINE

\$PRICE_LINE\$	- Price Line
LINE	- LINE
PriceLine	-
ND.LINE	-
PRICE_LINE	- Price Line

Pg 1 of 1

How does the System Know what Files you have Access to?

In the dictionary maintenance screen there is a hotkey called “Key”. This will provide you the name of the file(s) the dictionary item will give you access to.

Eclipse Dictionary Maintenance View Only

File Name : PRODUCT
Dict ID : PRICE_LINE Version # :

Dict Type : D Attr# : 9 Val# : Subval# :
Description : Price Line
Prompt : PR*View Only*

Key Files

PRICE.LINE

Justify (L/-
Maximum Width
Multi-Valued (Y/
Sub-Valued (Y/
Required (Y/
Allow Update (Y/
Case Mapping
SFA Category
Format (Output)
Update Subroutine :

2 of 1

Max Lines :
dexed (Y/N) : N

Copy Delete Valid Desc Expand Prompt Log Test Access **Key**

Undo Close Program

As we can see in the above figure the data element in the product file called price line is the Key that will give you access to the Price Line file itself. It is possible that a single data element can be the “key” to multiple files.

This one field in the dictionary maintenance screen is what drives the ability to use the files option or the forward slash option in the report writer module.

[illegible]

A new screen will appear on top of your report writer design displaying the end file and end dictionary item you retrieved and below the file you started in and how you linked into the 'end file'.

Just because a file is listed it doesn't mean that the desired data from the file is achieved. Be careful by clearly outlining what it is you need to extract from another file.

To be clear lets take a look at an example of a report writer report from the Buy Line file.

Report Writer/Mass Load Design									
Design ID : BUYLINE			Created : 04/01/04 By : Nicole Dursi						
File Name : BUY.LINE			Total Width : 81						
Title : Buyline Info									
Col	Dict Item/Formula		Width	Column Heading		Brk	Tot	Format	
1	@ID		10	B Line		N	*		
2	DESCRIPTION		30	Description		N	*		
3	!INITIALS~NAME		30	Buyer Name		N	*		
4	!PRODUCT~GL_PRODUCT_TY		8	GL_PRODUCT_TYPE		N	*	L8	
							4 of 4		
Sel Build		Adv Selection		Ecl Dct		Col Data		Copy Del Hdg	
Run Rpt		Dict SUM Label Opt		Notes View		Begin Load Set Val Log Files		Path	

In column 4 of the above figure, we are linking out to the product file to extract the GL_PRODUCT_TYP. The question that we need to ask ourselves is what are we using to get to the Product file from buy line file? The buy line is easy to get to from the product file because every product must have a buy line assigned to it. So, how are we going in the reverse direction?

Buy lines can have a Non-stock Default Template product assigned. It is not a requirement that a buy line has this, therefore some buy lines may have one and others may not.

When the path option is shown for column 4, we can see that the system is using the @ID of the non-stock template as our key into the product file.

Report Writer/Mass Load Design									
Design ID : BUYLINE		Created : 04/01/04 By : Nicole Dursi							
File Name : BUY.LINE		Total Width : 81							
Title : Buyline Info									
Col	Dict Item	File Path						Tot	Format
1	@ID	End File : PRODUCT						*	
2	DESCRIPTION	End Dict : GL_PRODUCT_TYPE						*	
3	!INITIALS~N	File			Foreign Key			*	
4	!PRODUCT~GL	BUY.LINE			NS_TEMPLATE_PROD			*	L8
		PRODUCT							

1 of 2
4 of 4

Rel Build
Adv Selection
Rel Dct
Rel Data
Cop
Rel
Rel
Rel
Rel
Rel

Run Rpt
Dict Sim
Label
Opt
Notes
View
Begin Load
Set Val
Lo
Files
Path

Does **one** single product provide you the data that you are looking for and keep in mind that it is a non-stock? If you were looking to display information for the item attached to the buy line that is a different story.

Exercise 5.1

1. From the Reports menu, select **Report Writer**.
2. Design a basic report using the Entity file. Set up your columns and your column headings to display:
 - Customer's Name
 - Customer's Contact
 - Contact's Phone number
 - Price Class the Customer belongs to
 - Customer's Credit Limit
 - Customer's Inside Salesman's Full Name
 - i. Hint: use the files option to retrieve this data for your report.
 - Customer's Start Date (Created Date)

Make sure your columns are wide enough to view all of the information in the fields.

3. Once you've painted your columns, stop.

Column Data

When your cursor is positioned on a dictionary item chosen for your report, the **Column Data** hot key may be highlighted. This indicates that more specific input is required in order for that dictionary item to produce accurate information on the report.

For example, the **PHONES** dictionary item forces the **Column Data** hot key to highlight. Because this dictionary item is multi-valued, you are prompted to designate which value to select for the report. From the Report Writer/Mass Load Design screen, you can use the **Eclipse Dict** hot key to display the Eclipse Dictionary Maintenance screen and confirm that the **Multi-Valued** field for this dictionary item is set to **Y**.

Eclipse Dictionary Maintenance				View Only	
File Name	: ENTITY	Version #	:		
Dict ID	: PHONE_NBRS				
Dict Type	: D	Attr#	: _17	Val#	: Subval# :
Description	: Phone Numbers				
Prompt	: PHONE_NBRS				
Justify (L/R)	: L	Data Format			
Maximum Width	: 20	() Date	Year :		
Multi-Valued (Y/N)	: Y	() Numeric	Decim :	Negs:	
Sub-Valued (Y/N)	: N	() Y/N Only			
Required (Y/N)	: N	() '*' Only			
Allow Update (Y/N)	: Y	() Time			
Case Mapping	:	() Word Wrap Disp Lines:	Max Lines :		
SFA Category	:	(*) Eclipse Dictionary	Indexed (Y/N) :	N	
Format (Output)	:	() Archived			
Update Subroutine	:				
Copy	Delete	Valid	T-Desc	EXpand	Prompt
Log	Test	Access	Key		
Undo	C10se	ProGram			

Use the **Multi-Valued (Y/N/B)** field to specify whether the information stored in this attribute has multiple values or is it branch-specific.

- **Y** – Column data will prompt for a multi-valued position.
- **N** – Column data will not prompt for anything.
- **B** – Column data will prompt for a branch.

For example, a vendor or customer's list of contacts and phones is multi-valued. When you use the PHONES dictionary item in Report Writer/Mass Load, the column data prompt will appear as "Multi Valued Pos."

In Lesson 3, you learned that information could be stored as a sub-value of a multi-value. The customer's credit parameter is a perfect example as we discussed earlier.

In other cases you may be prompted to enter a branch, territory or ALL as well as Ignore Branch Hierarchy shown below:

Col	Column Heading	Column Data	Prompt	Data
5*	BUYER	Enter Br/Tr/All Ignore Branch Hierarchy	N	

1 of 1

Expand Delete Data

Ignore Branch Hierarchy

This setting is a yes or no prompt. In Eclipse you can prioritize your territories so that one territory value will override another territory value.

A territory is a group of branches used for authorization, vendor, product and customer settings as well as reporting purposes. For example, a nationwide company can designate geographical territories, such as east, mid-west, and west. A branch can belong to more than one territory.

Territory Maintenance	
ID : GLOBAL	
Desc : Distribution Branches	
Entity Priority : 65	
Product Priority : 65	
Br	Branch Name
1	Branch 1
2	Branch 3
WIND	Windmere

4 of 4

Recall Delete Find

The territory maintenance screen is located on the Files menu under the Branches option.

The Entity and Product priority levels are used to determine which territory will take precedence over another territory.

So the question becomes, when running a report do you want to see the value that is posted on the product or entity for the branch or do you want to see the value that is being USED for the branch?

The hierarchy for priority is:

- Branch specific setting will override a territory setting.
- Territories look to the priority level defined here for hierarchy.
- The territory of “All” which is predefined in the system is used when a territory or branch setting is not found.
- Lastly, the system will use any default control parameter setting.

Responding to the common data prompts serves the same function as using the SET.COMMON command in TCL.

Exercise 5.2

1. Using the report you designed in Exercise 5.1, set the Column Data for the dictionary items that require more specific information.
2. Stop.

Break and Total

How you break and total the report should line up with the way that you plan on sorting the report.

When you flag a column to break, the system will provide to you a warning if the column is not already defined in your sorting criteria.

Report Writer/Mass Load Design

Design ID : SHIPVIA Created : 08/20/04 By : Nicole Dursi
 File Name : ENTITY Total Width : 106
 Title : NJ Customer's Ship Via Report

Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER_ID	8	Cus ID	N	*	
2	NAME	35	Customer Name	N	*	
3	ZIP_CODE	10	Zip Code	Y	*	
4*	CON					
5*	PHO					
6	OUT					

Dictionary is not in the Sort Criteria. Continue (Y/N) : _

3 of 6

☐ Build ☐ Adv Select ☐ Cl Dct ☐ Col Data ☐ Cop ☐ Del ☐ Idg ☐ Analyser

☐ Run Rpt ☐ Dict Sum ☐ Label ☐ Opt ☐ Notes ☐ View ☐ Begin Load ☐ Set Val ☐ Lo ☐ Files ☐ Path

In the above example we are sorting by Zip Code. However the zip code is not in our sort criteria yet as we have not begun the process of selecting out our records. You can still enter a Y in this column and continue. Simply remember that your first sort should be using the dictionary item you are breaking your report on.

What would happen if the break was designated for column 2 (Name), but the report was sorting on Zip Code?

Total Column

If this report had a dictionary item that was returning a dollar value or some sort of numeric data, we could use the total column to total up the values and provide a total at each break on the report.

Earlier in this lesson we saw an example of a report that was written from the product file. In the report we discussed the ability to do math between two columns on the report.

Report Writer/Mass Load Design											
Design ID : ND.PROD.VALUE				Created : 07/28/04 By : Nicole Dursi							
File Name : PRODUCT				Total Width : 91							
Title : Special Product Evaluation Report											
Col	Dict Item/Formula			Width	Column Heading		Brk	Tot	Format		
1	VIEWER.ID			10	Part No		N	*			
2*	DESC			35	Description		N	*			
3*	PREV.ONHAND			10	On-Hand		N	Y	MR0		
4	REPCOST			9	Rep-Cost		N	Y	MR3		
5	3*4			12	Ext Value		N	Y	MR2		
6	!BUY.LINE~PROCURE.GRP			10	PROCURE GROU		N	*			
									1 of 6		
Sel Build		Adv SelectIon		Ecl Dct		Col Data		CopV	Del	Hdg	AnalyZer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set Val	Log	Files	Path	

Note the total column is displaying a “Y” for the on-hand column, the replacement cost column as well as the column that will multiply the results of both.

If your goal was to obtain a calculated value such as a gross profit percentage or margin, the total column would use a “C” to calculate as opposed to the “Y” for totaling.

Formatting Data

Use the **Format** column on the Report Writer/Mass Load Design screen to change the output format of some dictionary items. You can format the following types of data:

- Date
- Time
- Case characters
- Numeric values

The following tables list the format conversion codes used by Report Writer/Mass Load.

Date Output

Code	Data	Output
D	10851	15 SEP 1997
D2-	10851	09-15-97
D4-	10851	09-15-1997
D2/	10851	09/15/97
D4/	10851	09/15/1997
DM	10851	9
DMA	10851	SEPTEMBER
DW	10851	1
DWA	10851	MONDAY
DY	10851	1997
DI	09/15/97	10851

Time Output

Code	Data	Output
MT	32400	9:00
MT	54000	15:00
MTH	3600	01:00AM
MTH	46800	01:00PM
MTS	54000	15:00:00
MTHS	54000	3:00:00PM

Character Output

Code	Data	Output
MCA	#123ABC*	ABC
MC/A	#123ABC*	#123*
MCN	#123ABC*	123
MCT	JOHN DOE	John Doe
MCU	John Doe	JOHN DOE
MCL	John Doe	john doe

Numeric Output

Code	Data	Output
MR0	123456	123456
MR2	123456	1234.56
MR3	123456	123.456
MR23	123456	123.46
MR2,	123456	1,234.56
MR2,\$	123456	\$1,234.56
MR2,\$*12	123456	\$***1,234.56
MR2E	-123456	<1234.56>
MR2D	123456	1234.56DB
MR2C	-123456	1234.56CR
MR2N	-123456	1234.56
MR2M	-123456	1234.56-
MR29	1234560000	1.23
MR29	1237891234	1.24

Exercise 5.3

1. On the report you are building, fill in the format columns as follows:

- Force the customer's name to appear in all Caps.
- Force the credit limit for the customer to appear with a dollar sign and comma if above 1000.00.
- Force a format of your choice for the date when the customer was created.

2. Stop.

Lesson 6

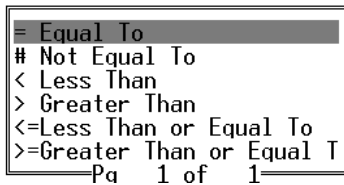
Selecting and Sorting Your Report

Objectives

After you complete this lesson, you will be able to:

- Select records to be included in your report
- Use the standard selection screen in Report Writer
- Use the advanced selection screen in Report Writer
- Sort your report data

Each condition is expressed using one of the following operators. Press **F10** in the **Op** column to display the options:



The TCL command that corresponds to the selection shown on the previous page is as follows:

SELECT ENTITY WITH V/C = "C" AND WITH STATE = "NJ" AND WITH SHIP_VIA = "[TRUCK]"

This command selects customers that have a ship via containing the word "truck" and reside in the state of "N.J." Both criteria need to be met for a customer to appear on the report.

Exercise 6.1

1. For the report you are creating, select those customers who belong to Price Class "3."
2. Stop.

Compare To Column on the Select Screen

The **Compare To** column is used in conjunction with the dictionary item and operator to narrow the number of records to appear on the report.

The information entered in the **Compare To** column can be:

- A dictionary item from the file used to create the report writer.
- A text string enclosed in quotation marks.
- A null (or blank) value.
- A user-defined prompt.

Comparing to a Dictionary Item

Use dictionary items to compare the data in one dictionary item to another. The dictionary item represents field/attribute data or an I-Descriptor that provides data.

For example, to generate a list of customers whose Inside Salesperson and Outside Salesperson are the same, enter the following information:

Report Writer/Mass Load Selection			
File Name : ENTITY	Sample :	Count :	0
Select Number : 1		Reselect (Y/N) :	Y
Conj	Dictionary Name	Op	Compare To
***	OUTSIDE_SALES	=	INSIDE_SALES

To produce a list of customers whose Inside Salesperson and Outside Salesperson are not the same, use the # operator.

Comparing to a Text String

A text string is specific information designated by you to compare to the dictionary name. You can also use wild cards, as discussed earlier, in this format.

For example, to select customers assigned to a designated Inside Salesperson, enter the following information:

Report Writer/Mass Load Selection			
File Name : ENTITY	Sample :	Count :	0
Select Number : 1		Reselect (Y/N) :	Y
Conj	Dictionary Name	Op	Compare To
***	INSIDE_SALES	=	"BERGERD"

This conditional statement selects records whose INSIDE_SALES attribute contains "BERGERD."

To list those customers whose Inside Salesperson is not "BERGERD," use the # operator.

Comparing to a Null Value

Use the null value in the **Compare To** column to select records based on whether the designated dictionary item contains data or does not contain data. A null value is represented by two quotation marks.

For example, to select customers who have not been assigned an Inside Salesperson, enter the following information:

Report Writer/Mass Load Selection			
File Name : ENTITY	Sample :	Count :	0
Select Number : 1		Reselect (Y/N) :	Y
Conj	Dictionary Name	Op	Compare To
***	INSIDE_SALES	=	" "

This statement selects records whose INSIDE_SALES attribute is empty.

For example, to select all customer records that were created within a range of dates, enter the following information:

Report Writer/Mass Load Selection			
File Name : ENTITY		Sample :	Count : 0
Select Number : 1		Reselect (Y/N) : Y	
Conj	Dictionary Name	Op	Compare To
***	START_DATE	>=	\$SDATE\$
AND	START_DATE	<=	\$EDATE\$

\$SDATE\$ and \$EDATE\$ are the user-defined fields that will contain the values entered at the prompt. If you used \$DATE\$ for both prompts, the system would only prompt for one date instead of a date range. When requiring different information for the same dictionary item, it is important that the prompt be unique for each one.

Use the **Selection Data** hot key on the Report Writer/Mass Load Selection screen to enter the data requested by the prompts.

Selection Data		
Selection	Prompt	Data
\$SDATE\$	Start Date >=	/ /
\$EDATE\$	Start Date <=	
		1 of 2
Multi	Delete Data	

The text displayed in the **Prompt** column is comprised of the **Prompt** associated with the dictionary item you are prompting for and the operator from the conditional statement.



Eclipse Dictionary Maintenance				View Only	
File Name	: ENTITY	Dict ID	: START_DATE	Version #	:
Dict Type	: D	Attr#	: _48	Val#	:
Description	: Start Date (Create Date)	Subval#	:		
Prompt	: Start Date				
Justify (L/R)	: R	Data Format	(*) Date	Year	: 4
Maximum Width	: 5		() Numeric	Decim	: Negs:
Multi-Valued (Y/N)	: N		() Y/N Only		
Sub-Valued (Y/N)	: N		() '*' Only		
Required (Y/N)	: N		() Time		
Allow Update (Y/N)	: Y		() Word Wrap	Disp Lines:	Max Lines :
Case Mapping	:		(*) Eclipse Dictionary	Indexed (Y/N)	: N
SFA Category	:		() Archived		
Format (Output)	:				
Update Subroutine	:				
Copy	Delete	Valid	I-Desc	Expand	Prompt
Log	Test	Access	Key		
Undo	Close	Program			

To change these default prompts, use the **Edit Prompts** hot key on the Report Writer/Mass Load Selection screen. When first displayed, the screen shows the default prompts. You can replace the defaults with your own text.

Report Writer/Mass Load Selection			
File Name	: ENTITY	Sample	:
Select Number	: 1	Count	: 0
Conj	Dictionary Name	Compare To	Reselect (Y/N) : Y
***	START_DATE		
AND	START_DATE		
<div>2 of 2</div>			
Begin Sel	Edit Prompts	Selection Data	Delete Sel
Sort	Addl Slct	Path	

Exercise 6.2

1. For the report you are creating, select those customers that have a ship via assigned and an outside salesperson assigned.
2. Narrow your select further by selecting only those customers who have a credit limit greater than one hundred dollars.
3. Stop.

***Note:** The ENTITY file contains both vendor and customer records. Use the V/C dictionary ID to select records that are customers or vendors.*

- V/C = “C” for customer records.
- V/C = “V” for vendor records.

Sorting the Report

After records have been selected, you may want to sort them before running the report. You should sort your records to work in conjunction with the break points and totals defined on the Report Writer/Mass Load Design screen.

Use the **Sort** hot key on the Report Writer/Mass Load Selection screen to designate the sorting sequence.

In the following example the selected customer records will sort first by the outside salesperson in ascending (alphabetical) order. Then the records in each salesperson's group will sort by the customer name in ascending (alphabetical) order.

Sort Sequence	
Design ID :	
OUTSIDE_SALES	Order A
NAME	A
-	
Path	F12-Abort

Enter the dictionary items to be used as sort criteria and designate the sort **Order** for each as ascending or descending. The **A** represents the TCL “BY” option discussed earlier. The **D** represents the TCL “BY-DSND” option discussed earlier.

Use the Path option to use dictionaries from other files while sorting.

Exercise 6.3

1. For the report you are creating, sort the selected records by Zip code and then alphabetize the customer names within each Zip code.

Report Writer Advanced Selection Screen

The Report Writer Selection screen, displayed using the **Adv Selection** hot key, serves the same function as the SSELECT command in TCL. You can use this screen, just as you used the Report Writer/Mass Load Selection screen, to specify the criteria for selecting and sorting the records to be included in the report.

Advanced Report Writer/Mass Load Selection Screen		
File Name : ENTITY	Sample :	Reselect (Y/N) : Y
1. WITH START_DATE >= \$SDATE\$ AND WITH START_DATE <= \$EDATE\$		Count
2. BY OUTSIDE_SALES BY NAME		
3.		
4.		
5.		

Begin Select	Edit Prompts	Selection Data	Delete Selections	Path	
--------------	--------------	----------------	-------------------	------	--

Note: This screen is less structured than the Report Writer/Mass Load Selection screen. You should have a good understanding of the TCL commands before using this screen.

Exercise 6.4

1. Create your report again, this time instead of using the Select screen to select the report, use the Advance Select Screen.

Report Driver: Running the Report

The Run option from the report writer/mass load screen will run send the report to either your hold file or the printer depending on what you choose.

Report Driver					
Design ID : PRODUCT.LINE		Created : 04/01/04 By : Nicole Dursi			
File Name : PRODUCT		Total Width : 90			
Title : Names of the Price Lines assigned					
Sample :					
Multi Value Pos :					
Enter Br/Tr/All :					
Enter End/As of Date :					
Print	Hold	Opts	Column Data	Selection Data	Notes
					1 of 3

The options that appear on this screen will be determined by the select criteria/column data from the report writer itself.

If a dictionary item on the column of your report requires additional information such as the Branch/Territory or All and the information is NOT entered on the column data option from the report writer screen itself, the system will prompt you to enter the information here.

If more than one dictionary prompts for the same information, the system will only prompt you on this screen ONCE. This is important to understand if you want columns on the report to have different information. The detail for each column should be entered under the column data on the report writer.

If you forget, this screen comes with the column data option. This will bring you into the column data screen displaying the columns that are requiring more information.

Column Data			
Col	Column Heading	Prompt	Data
2*	Product Description	Multi Value Pos	-
4*	On-Hand	Enter Br/Tr/All	
		Enter End/As of Date	
			1 of 2
Expand		Delete Data	

The most important aspect to recognize is that the prompts on the Report Driver are not limited to the column data prompts from the report writer itself. Any dictionary from your select screen will also be included on the report driver.

The same rules will apply meaning that if more than one dictionary item requires the same information, you will only be prompted once... An exception to this would be data ranges for which you are prompting for a beginning and ending date. Please refer back to your selecting data section of this workbook if you need more information.

Other options from the Report Driver screen such as the **Sample** prompt may be useful.

Report Driver	
Design ID : PRODUCT.TEST	Created : 09/02/04 By : Nicole Dursi
File Name : PRODUCT	Total Width : 73
Title : testing	
Sample :	
Multi Value Pos :	
Enter Br/Tr/All :	
Enter End/As of Date :	
Ignore Branch Hierarchy :	
LINE = :	
1 of 5	
Print	Hold
Opts	Column Data
Selection Data	Notes

You can access this prompt simply by using your arrow key to move up. Here you can enter a numeric value of the number of records you want to sample. This allows you to determine if the report is what you want it to be before running it for all of the records.

Another key component of this screen is the Selection Data which provides the ability to enter multiple listings for a prompt from the select screen.

For example: The above report driver has a prompt for a Price Line. This prompt was added to the report selection screen. Using the Selection Data option the system will open a new screen shown below where multiple price lines can be added using the “Multi” option.

Selection Data		
Selection	Prompt	Data
\$LINE\$	LINE =	-
		1 of 1
<input type="button" value="Multi"/>	<input type="button" value="Delete Data"/>	

The notes option is used to provide notes for the end user that may help them to better understand the report or the use for the report. These notes are entered by the writer of the report. Instructions on how to run the report are also useful notes.

Lesson 7

Report Writer Options

Objectives

After you complete this lesson, you will understand:

- Understanding other options available from Report Writer
- Creating Mailing Labels using Report Writer

Report Writer Options Screen

The report writer options screen provides the ability to show the number of records that appear on the report, create a column delimited file format, double space your report and print the prompts entered on the Report Writer/Mass Load Selection screen that were used to select the data for the report.

Report Writer/Mass Load Design						
Design ID : SHIPVIA_ADVANCED		Created : 08/20/04		By : Nicole Dursi		
File Name : ENTITY		Total Width : 112				
Title : NJ Customer's Ship Via Report						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER_ID	8	Cus ID	N	*	

Report Writer Options Screen	
Spaces Between Lines :	0
Print Prompts? (Y/N) :	Y
Print Line Numbers? :	
-Download Options-	
Column Delimiter :	
Record Delimiter :	
Trim Blanks? (Y/N) :	N

						1 of 7
Del Build	Adv Selection	Del Dct	Del Data	Cop	Del	Delg Analyser
Run Rpt	Dict Sim	Label	Opt	Notes	View	Begin Load
Set Val	Lo	Files	Path			

For those of us that wear glasses and at times feel that they cannot see the world, reading a report can become tedious. For that reason the options screen allows you to provide **spaces between lines** on your report. When the report runs to your hold file the rows will appear double spaced if you enter a 1.

The **print prompts** option is a Yes or No response with the default set to a Yes. Any data specified on the select criteria screen using the Selection option will appear in the header of the report. This will help to see how the report was selecting at the time it ran.

The **print line numbers** prompt looks like it could be a yes or no question. However, this is really asking for a numeric value. The number you enter at this prompt will dictate how wide the column will be. Each record that appears on the report will be counted sequentially to the end of the report. The column will automatically become the very first column on your report.

If you are planning to download the report so that you can open it up in Excel or some other third party software you can specify what the delimiters will be and whether you wish to trim out any blank (null) spaces that appear at the end of each string on the report.

Trimming the spaces makes the size of the report smaller and therefore quicker to download to your pc. The columns can then be extracted in Excel by using the text to columns option.

The column delimiter can be what ever character you specify. The pipe sign (|) works well here because typically one would not see a pipe sign in the data.

An example of a report using the trim and delimiter function is shown below:

```
ID..... Product Description..... Onhand.... Buyer.....
                               Value.....

^000005932|EG-RWH-10-40 10YR PARTS & LABOR|7|IGNORE
^000000146|1-1/4" X 10' SCH 40 PVC PIPE|27|DAVIDB
^000000147|1-1/4" X 20' SCH 40 PVC PIPE|23|DAVIDB
^000000141|1/2" X 20' SCH 40 PVC PIPE|82|DAVIDB
^000000142|3/4" X 10' SCH 40 PVC PIPE|2|DAVIDB
^000000143|3/4" X 20' SCH 40 PVC PIPE|16|DAVIDB
^000000145|1" X 20' SCH 40 PVC PIPE|86|DAVIDB
```

Exercise 7.1

1. Pull up the report you created and access the options screen.
2. Make your report double spaced and show the number of lines on the report.
3. Re-run your report and verify the results in your hold file.

Exercise 7.2

2. Create a simple report from the product file that will select out only stock keeping items for the price line assigned to you by your instructor.
3. Use the options to trim the blanks with a column delimiter.
4. Run the report to your hold file and verify the results.

Creating Mailing Labels

Mailing labels are accomplished in Eclipse by using a special dictionary item and the Label option on your report writer.

The dictionary item is simply NA which stands for Name and Address. This dictionary item will position the name and the address in such a way you would see on a mailing label or envelope.

We will discuss this dictionary in more detail later in this workbook.

The report writer report should be set up similar to the following:

Report Writer/Mass Load Design							
Design ID : CUST.LABELS		Created : 07/30/04 By : Nicole Dursi					
File Name : ENTITY		Total Width : 39					
Title : Customer Mailing Labels							
Col	Dict	Item/Formula	Width	Column Heading	Brk	Tot	Format
1	SPACE		1	SPACE	N	*	
2	NA		35	NAME&ADDR	N	*	
3	SPACE		1	SPACE	N	*	
4 of 4							
Sel	Build	Adv Select	Ion	Ecl Dct	Col Data	CopY	Del
Hdg	Analy	Zer					
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL
LoG	Files	Path					

Depending on the type of labels used and how many lines will be available for the label will determine the number of added line spaces you will add.

The above example is assuming a label that will hold 6 lines of information. Because the addresses in our training data will only be a maximum of 4 lines, we are padding the label with two extra lines using the SPACE dictionary.

Once we have our columns painted, now we need to define the label layout.

Use the label option on the report writer screen to access.

Report Writer/Mass Load Design							
Design ID : CUST.LABELS		Created : 07/30/04 By : Nicole Dursi					
File Name : ENTITY		Total Width : 39					
Title : Customer Mailing Labels							
Col	Dict	Item/Formula	Width	Column Heading	Brk	Tot	Format
1	SPACE		1	SPACE	N	*	
2	NA		35	NAME&ADDR	N	*	
3	SPACE		1	SPACE	N	*	
							4 of 4
Sel Build	Adv Selection	Ecl Dct	Col Data	Copy	Del	Hdg	Analyzer
Run Rpt	Dict Sum	Label Out	Notes	View	Begin Load	Set Val	Log Files
Path							

Once activated the Report Writer Label Specifications screen will open.

Report Writer/Mass Load Design																																								
Design ID : CUST.LABELS		Created : 07/30/04 By : Nicole Dursi																																						
File Name : ENTITY		Total Width : 39																																						
Title																																								
<table border="1"> <thead> <tr> <th>Col</th> <th>Dict</th> <th>Report Writer Label Specifications</th> <th>Format</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>SPACE</td> <td>Print Labels (Y/N).....: N</td> <td rowspan="10"></td> </tr> <tr> <td>2</td> <td>NA</td> <td># of Labels Across The Page.....: 1</td> </tr> <tr> <td>3</td> <td>SPACE</td> <td># of Lines Per Label (Height).....: 5</td> </tr> <tr> <td></td> <td></td> <td># of Lines to Skip Between Labels...: 1</td> </tr> <tr> <td></td> <td></td> <td># of Spaces to Indent (Left Margin)..: 0</td> </tr> <tr> <td></td> <td></td> <td># of Spaces Per Label (Width).....: 35</td> </tr> <tr> <td></td> <td></td> <td># of Spaces to Skip Between Labels...: 0</td> </tr> <tr> <td></td> <td></td> <td>Print Null Fields (Y/N).....: N</td> </tr> <tr> <td></td> <td></td> <td># of Line Up Labels to Print.....: 0</td> </tr> <tr> <td></td> <td></td> <td># of Copies to Print For Each Label.: 1</td> </tr> <tr> <td></td> <td></td> <td>Print Label Headings (Y/N).....: N</td> </tr> </tbody> </table>			Col	Dict	Report Writer Label Specifications	Format	1	SPACE	Print Labels (Y/N).....: N		2	NA	# of Labels Across The Page.....: 1	3	SPACE	# of Lines Per Label (Height).....: 5			# of Lines to Skip Between Labels...: 1			# of Spaces to Indent (Left Margin)..: 0			# of Spaces Per Label (Width).....: 35			# of Spaces to Skip Between Labels...: 0			Print Null Fields (Y/N).....: N			# of Line Up Labels to Print.....: 0			# of Copies to Print For Each Label.: 1			Print Label Headings (Y/N).....: N
Col	Dict	Report Writer Label Specifications	Format																																					
1	SPACE	Print Labels (Y/N).....: N																																						
2	NA	# of Labels Across The Page.....: 1																																						
3	SPACE	# of Lines Per Label (Height).....: 5																																						
		# of Lines to Skip Between Labels...: 1																																						
		# of Spaces to Indent (Left Margin)..: 0																																						
		# of Spaces Per Label (Width).....: 35																																						
		# of Spaces to Skip Between Labels...: 0																																						
		Print Null Fields (Y/N).....: N																																						
		# of Line Up Labels to Print.....: 0																																						
		# of Copies to Print For Each Label.: 1																																						
		Print Label Headings (Y/N).....: N																																						
F12-Absort		4 of 4 Analyser																																						
<table border="1"> <tr> <td>Run Rpt</td> <td>Dict Sim</td> <td>Label</td> <td>Opt</td> <td>Notes</td> <td>View</td> <td>Begin Load</td> <td>Set Val</td> <td>Lo</td> <td>Files</td> <td>Path</td> </tr> </table>			Run Rpt	Dict Sim	Label	Opt	Notes	View	Begin Load	Set Val	Lo	Files	Path																											
Run Rpt	Dict Sim	Label	Opt	Notes	View	Begin Load	Set Val	Lo	Files	Path																														

There is no rhyme or reason for the default information you see on this screen other than the Print Labels yes or no. Eclipse assumes that when you are creating a report you are creating it as a normal report and not labels, therefore the default is a No.

Take a look at the labels that you plan on printing the customer addresses on. It is important to know the design of the label when filling in the specifications here.

Most often customer mailing labels are the avery labels that are 3 labels across the page and 10 labels down the page.

Of Labels Across the Page - The number of labels that can be printed across the page.

Of Lines per Label (Height) - The maximum number of lines that can be used for printing on a label.

Of Lines to Skip between Labels - The number of lines to skip between rows of labels.

Of Spaces to Indent (Left Margin) - The number of character spaces to indent from the left edge of the label before printing the text on the label.

Of Spaces per Label (Width) - The maximum number of character spaces that can be used for printing a line of text on a label.

Of Spaces to Skip between Labels - The number of character spaces to skip between labels across the page.

Report Writer/Mass Load Design		
Design ID : CUST.LABELS		Created : 07/30/04 By : Nicole Dursi
File Name : ENTITY		Total Width : 39
Title		
Col	Dict	Report Writer Label Specifications
1	SPACE	Print Labels (Y/N).....: N
2	NA	# of Labels Across The Page.....: 3
3	SPACE	# of Lines Per Label (Height).....: 6
		# of Lines to Skip Between Labels...: 1
		# of Spaces to Indent (Left Margin)..: 3
		# of Spaces Per Label (Width).....: 32
		# of Spaces to Skip Between Labels...: 1
		Print Null Fields (Y/N).....: Y
		# of Line Up Labels to Print.....: 0
		# of Copies to Print For Each Label..: 1
		Print Label Headings (Y/N).....: N
F12-Abort		4 of 4
Sel Build		Analyzer
Run Rpt	Dict	Sim
Label	Opt	Notes
View	Begin Load	Set Val
Lo	Files	Path

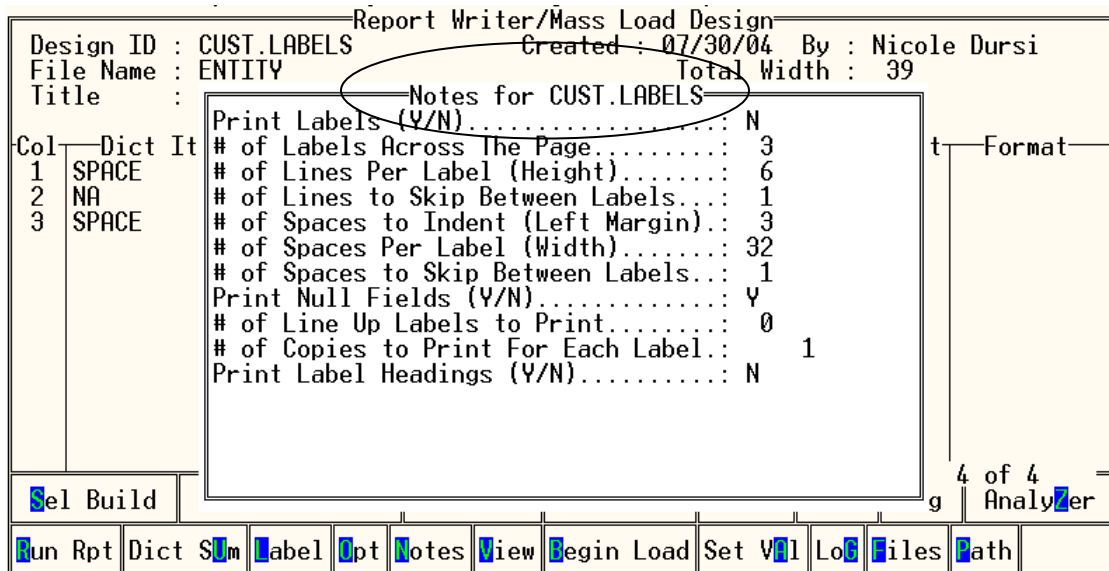
If you need to test a sample of records you can either use the sample option when running the report or you can indicate the number of labels to print in your testing. You are looking to make sure that the labels will not creep up when the second and third page of labels are printed out.

If you need more than one copy of your labels to print, you can change the # of copies option from a 1 to the desired number.

The very last option of printing the label headings is not currently doing anything.

Once you have the Label Specifications filled in and have tested them you may want to copy this information into the **Notes** option of the report writer. This way if someone changes a value, you will know what your settings were.

This is not a requirement, just a useful tip.



Report Writer/Mass Load Design

Design ID : CUST.LABELS Created : 07/30/04 By : Nicole Dursi
 File Name : ENTITY Total Width : 39
 Title : Notes for CUST.LABELS

Col	Dict	It	
1	SPACE		# of Labels Across The Page..... 3
2	NA		# of Lines Per Label (Height)..... 6
3	SPACE		# of Lines to Skip Between Labels... 1
			# of Spaces to Indent (Left Margin).. 3
			# of Spaces Per Label (Width)..... 32
			# of Spaces to Skip Between Labels... 1
			Print Null Fields (Y/N)..... Y
			# of Line Up Labels to Print..... 0
			# of Copies to Print For Each Label.. 1
			Print Label Headings (Y/N)..... N

4 of 4
g Analyser

Buttons: Sel Build, Run Rpt, Dict Sim, Label Opt, Notes View, Begin Load, Set Val, Log, Files, Path

Exercise 7.3

1. Create a report writer for custom labels. Use your initials as the prefix to your design id.
2. Select out customers only that have a customer type of NE.NORM.
3. Fill in your desired label specifications and test by sending the report to your hold file.

Note: Use the sample option on the Run Driver Screen by positioning your cursor on the prompt and entering the number of records you wish to use in your test, or specify your test input on the label specification screen as discussed.

Lesson 8

Creating a Mass Load

Objectives

After you complete this lesson, you will understand:

- What can be mass loaded in Eclipse
- How to use the Default/Set Value column

Mass Load

Mass Load enables you to update information in the Eclipse database. It is commonly used to enter information, such as credit parameters for customers or inventory control parameters for products.

The Mass Load program lists the records and fields to be updated on the Mass Load Update screen. You can manually update these fields one record at a time or, if each record is being updated with the same information, you can let the system update all the records at once.

There are three major steps in creating a mass load:

- **Designing the Mass Load** – First you design the Mass Load Update screen layout, including the file and fields to be updated.
- **Selecting the Mass Load Data** – After you have designed the mass load screen layout, you need to specify the criteria for selecting the records to be updated.
- **Running the Mass Load** – After you have designed the mass load screen layout and selected the records to be updated, you need to perform the updates.

Rules for Mass Loading

The following two rules apply to mass loading:

- You cannot mass load information into an I-Descriptor. You can only mass load into D-type dictionary items, which represent actual fields in a record.
- You cannot mass load information into a dynamic file.

Examples of dynamic files that cannot be mass loaded are:

- | | |
|---------------|--------|
| • PROD.DYNAM | • PSUB |
| • LEDGER | • PHYS |
| • ORDER.QUEUE | • AR |

Examples of static files that can be mass loaded are:

- | | |
|--------------|------------|
| • ENTITY | • BUY.LINE |
| • PRODUCT | • INITIALS |
| • PRICE.LINE | • ZIP |

The most common files where mass loading occurs are the ENTITY and PRODUCT files.

Designing the Mass Load

Use the mass load view of the Report Writer/Mass Load Design screen for designing the layout of the Mass Load Update screen. From the Files menu, select **Mass Load/Update** to display this view.

Report Writer/Mass Load Design				
Design ID : SHIPVIA		Created : 08/20/04 By : Nicole Dursi		
File Name : ENTITY		Total Width : 116		
Title : NJ Customer's Ship Via Report				
Col	Dict Item/Formula	Width	Type	Default/Set Value
1	VIEWER_ID	8	D*	
2	NAME	35	D	
3	ZIP_CODE	10	D	
4*	CONTACTS	20	D	
5*	PHONE_NBRS	20	D	
6	OUTSIDE_SALES	8	D	
7	START_DATE	5	D	
8	STATE	2	D	

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	8 of 8	Analyzer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG Files Path

In the header portion of the screen, assign an ID to the design and designate the file that contains the records to be updated.

In the body of the screen, you must describe each column you want to display on the Mass Load Update screen. Some columns just display information that identifies the record and other columns identify the fields to be updated. Sequential numbers identify the columns on the screen, with 1 being the leftmost column, 2 being the next column to the right, and so forth.

For each column of the display you must:

- Identify the dictionary item (or formula) to be displayed in the column.
- Specify the width of the column.

The system populates this field with the width defined for the dictionary item. You cannot change the width of a dictionary item you plan to update. If you change the width, the field becomes display-only.

- Indicate whether the data in the column is for display only or to be updated.
- If the data is to be updated, specify how and with what value.

Identifying the Data Type

The value you enter in the **Typ** field indicates whether the data in that column is display-only or can be updated.

Data Type	Data in this mass load column...
D (Display)	is display-only and cannot be changed. The system displays an asterisk (*) following a D when the field is a dictionary item that does not allow updates. Note: Any time the width of a column on the mass load is changed an asterisk will appear and will deem the data element display only.
U (Update)	can be manually updated. <ul style="list-style-type: none"> • If the Default/Set Value field on this screen is left blank, you can manually enter the data for each record on the Mass Load Update screen. • If the Default/Set Values field on this screen contains a value, you can enter this same value for each record, one at a time, on the Mass Load Update screen as you scroll through the records using the Enter key.
S (Set)	will be updated by the system according to the data entered in the corresponding Default/Set Value column. <p>Caution: Do not use this Typ code until you are sure that you are updating the correct field with the correct data.</p>

Setting a Replacement Value

When the **Typ** code for a column of data is **S** or **U**, the data entered in the **Default/Set Value** field determines what will be placed in that field.

Value	Function
“”	A set of quotes with nothing between them replaces the current value in the designated field with a null value.
“ ”	A space enclosed in quotes replaces the current value in the designated field with a space.
“text”	Text enclosed in quotes replaces the current value in the designated field with the actual text.
Dict ID	A dictionary ID replaces the current value in the designated field with the value currently stored in that dictionary ID.
3+2	A number or numerical expression replaces a value in a designated numerical field with the value of that number or expression.
:	A colon can be used to concatenate two values. This means you are combining data together to create new data. For example, Dict ID:“TEST”
	A separator indicates the data will be added and should be word wrapped to the length specified for that field in the mass load.
“value”=“new-value”	This is a Find and Replace method of mass loading. The mass load routine will find the string of characters entered in the first text string and replace it with the value entered in the second text string.
%AM%, %VM%, %SVM%	Allow you to specify new line characters or record delimiters in your data.
blank	If the Typ code is “U” and you leave this field blank, you can manually update this field for each record on the Mass Load Update screen. If the Typ code is “S”, do not leave this field blank.

Leaving the Default/Set Value Field Blank

If you enter **U** in the **Typ** field and leave the **Default/Set Value** field blank, you can manually update this field for each record on the Mass Load Update screen.

Report Writer/Mass Load Design				
Design ID : SHIPVIA		Created : 08/20/04 By : Nicole Dursi		
File Name : ENTITY		Total Width : 73		
Title : NJ Customer's Ship Via Report				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	8	D*	
2	NAME	35	D	
3	ZIP_CODE	10	D	
4	OUTSIDE_SALES	8	U	-
5	START_DATE	5	D	
6	STATE	2	D	

Sel Build	Adv Selection	Ecl Dct	Col Data	Copy	Del	Hdg	4 of 6	Analyzer
Run Rpt	Dict Sum	Label	Opt	Notes	View	Begin Load	Set Val	Log Files Path

Exercise 8.1

1. Create a mass load, using the select criteria assigned by the instructor, to manually update the first contact name assigned to each customer record.

Replacing a Value with a Text String

If you enter **U** in the **Typ** field and text enclosed in quotation marks in the **Default/Set Value** field, you can replace the current value with that text.

For example, when one salesperson replaces another, select all the records assigned to the original salesperson and display the SALEMAN dictionary item for each record in update mode. Enter the new salesperson's ID, enclosed in quotation marks, in the **Default/Set Value** field. When you process each record, the system will overwrite the original value with the new value.

Report Writer/Mass Load Design				
Design ID : AO.SMITH		Created : 08/20/04 By : Nicole Dursi		
File Name : PRODUCT		Total Width : 43		
Title : Description Fix				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	7	D*	
2*	DESC	35	U	"AOSMITH"="AO SMITH"

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	3 of 3	AnalyZer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG Files Path

You can also search for and replace part of a field with a text string.

In the following example the vendor name 'AO SMITH' was loaded as 'AOSMITH' in the **Description** field and a space is needed to separate the two words.

Report Writer/Mass Load Design				
Design ID : AO.SMITH		Created : 08/20/04 By : Nicole Dursi		
File Name : PRODUCT		Total Width : 43		
Title : Description Fix				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	7	D*	
2*	DESC	35	U	"AOSMITH"="AO SMITH"

3 of 3

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	AnalyZer			
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG	Files	Path

The first value in quotes represents the string of data you want to search for. The equal sign is a replacement operator, indicating that the first value should be replaced by the text within the following set of quotation marks.

As you step through the records listed on the Mass Load Update screen, the space is inserted between the two words, as shown below:

Mass Load Update		Recall	Expand
File Name:PRODUCT	Design ID:AO.SMITH		
IEWER-DESCRIPTION	Description		
022945 PGD-50 AO SMITH 50-GAL RG 15T/10P W	PGD-50 AO SMITH 50-GAL RG 15T/10P W		
022953 PGD-75 AO SMITH 75-GAL RG 15T/10P W	PGD-75 AO SMITH 75-GAL RG 15T/10P W		
022956 PGC-30 AO SMITH 30-GAL RG 10T/10P ² W	PGC-30 AO SMITH 30-GAL RG 10T/10P		
022957 PGC-40 AO SMITH 40-GAL RG 10T/10P W	PGC-40 AO SMITH 40-GAL RG 10T/10P W		
022958 PGC-50 AOSMITH 50-GAL RG 10T/10P WA	PGC-50 AOSMITH 50-GAL RG 10T/10P WA		
022960 FGC-30 AOSMITH 30-GAL RG 8T/8P ² WAT	FGC-30 AOSMITH 30-GAL RG 8T/8P		
022969 FGC-40 AOSMITH 40-GAL RG 8T/8P WAT	FGC-40 AOSMITH 40-GAL RG 8T/8P WAT		

Exercise 8.2

1. Using select criteria assigned by your instructor, create a mass load that will assign a designated ship via to a group of customers.

Setting a Value to Null

A set of quotes with nothing between them replaces the current value in the designated field with a null value. This erases any information that previously existed in the field. Be very careful when using this option.

For example, your purchasing department has assigned new procurement groups to all the buy lines and now you want to remove all the procurement groups assigned at the product level.

Report Writer/Mass Load Design				
Design ID : PROC_GROUP		Created : 08/20/04 By : Nicole Dursi		
File Name : PRODUCT		Total Width : 30		
Title : Procure Group Fix				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	7	D*	
2	BUY_LINE	8	D	
3	PROCURE_GROUP	13	U	...
-				
<div> <div>Sel Build</div> <div>Adv Selection</div> <div>Ecl Dct</div> <div>Col Data</div> <div>CopY</div> <div>Del</div> <div>Hdg</div> <div>4 of 4 AnalyZer</div> </div>				
<div> <div>Run Rpt</div> <div>Dict SUM</div> <div>Label</div> <div>Opt</div> <div>Notes</div> <div>View</div> <div>Begin Load</div> <div>Set VAL</div> <div>LoG</div> <div>Files</div> <div>Path</div> </div>				

Replacing a Value with another Dictionary ID Value

To replace the value in one dictionary item with the value from another dictionary item, enter a dictionary ID in the **Default/Set Value** field.

For example, if your company adds a new branch, you need to update the PRODUCT file to include sell group assignments for the new branch. The following mass load copies the branch 1 sell group assignment (SELL_GRP,1) to the new branch (SELL_GRP).

Note that SELL_GRP is a multi-valued dictionary item. It contains the sell group assignments for all the branches. The asterisk next to the column number indicates that you must enter column data to specify which multi-value (branch) you are updating.

SELL_GRP,1 : using the comma and branch number after the dictionary item will pinpoint the data in the branch specified to be copied into the new branch loaded in the column data.

Report Writer/Mass Load Design				
Design ID : PRODUCT		Created : 08/20/98 By : Steven A. Grundt		
File Name : PRODUCT		Total Width : 56		
Title : MAINTAIN SELL GROUPS AND DESC'S				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	7	D*	
2*	DESC	35	D	
3*	SELL_GROUP	12	U	SELL_GROUP,1

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	4 of 4	AnalyZer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG Files Path

Replacing a Value with a Text String

You can replace a text string with new information.

In the following example, you can use the mass load program to replace the abbreviation “Ave” with the word “Avenue” in the ADDRESS dictionary item.

Report Writer/Mass Load Design				
Design ID : MASSLOAD.EXAMPLE		Created : 08/20/04 By : Nicole Dursi		
File Name : ENTITY		Total Width : 92		
Title : Example Mass Load				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	8	D*	
2	NAME	35	D	
3*	ADDRESS	35	U	"AVE"="AVENUE"
4	CREDIT_LIMIT	11	D	

Sel Build	Adv SelectIon	Ecl Dct	Col Data	Copy	Del	Hdg	5 of 5	AnalyZer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	LoG Files Path

Warning: Be careful when running a mass load of this nature. Think clearly what would occur if this load was run 2 times or 3 times?

Exercise 8.3

- Using the selection criteria assigned by the instructor, create a mass load that will change the address to display “Street” instead of the abbreviation “St”.

Replacing a Value with a Concatenated Value

Use a colon to concatenate two values in the **Default/Set Value** field.

Consider the following problem: You are trying to update your prices using the auto price update program, but your products do not have a UPC number for you to match to a diskette. So, you are forced to use the model number of the product to update your prices. Because another price line might use the same model number, you need to make the model number unique to the line you want to update. You can do this by concatenating a unique prefix to the model number and then storing this new ID in a user-defined field.

In the following example, the mass load concatenates the text string “AOS” to an I-Descriptor dictionary item that extracts the first word of the product description. This information is placed in a position of the multi-valued PU.IDS (Price Updating ID Maintenance) dictionary item, based on what you enter for the column data prompt.

Report Writer/Mass Load Design				
Design ID : PRODUCT		Created : 08/20/98 By : Steven A. Grundt		
File Name : PRODUCT		Total Width : 77		
Title : MAINTAIN SELL GROUPS AND DESC'S				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	7	D*	
2*	DESC	35	D	
3*	SELL_GROUP	12	D	
4*	PU_IDS	20	U	"AOS-":DESC.1STWD
	-			

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	5 of 5	Analyzer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL	Log Files Path

Exercise 8.4

- Using the selection criteria designated by your instructor, create a mass load to update the **Commodity Code** field in the PRODUCT file with the sell group in branch 1 concatenated to your initials.

Replacing a Value with a Numerical Expression

You can replace a numerical value with another number or a value calculated by a numerical expression.

You can use the mass load in the following example to increase the credit limit of each selected record by 2 percent.

Report Writer/Mass Load Design				
Design ID : MASSLOAD.EXAMPLE		Created : 08/20/04 By : Nicole Dursi		
File Name : ENTITY		Total Width : 92		
Title : Example Mass Load				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	8	D*	
2	NAME	35	D	
3*	ADDRESS	35	D	
4	CREDIT_LIMIT	11	U	CREDIT_LIMIT*"1.02"
	-			

Sel Build	Adv Selection	Ecl Dct	Col Data	CopY	Del	Hdg	5 of 5 Analyzer
Run Rpt	Dict SUM	Label	Opt	Notes	View	Begin Load	Set VAL
LoG	Files	Path					

Exercise 8.5

- Using the selection criteria assigned by your instructor, increase the credit limit for each customer by 4%.

Word Wrapping

The pipe sign (|) at the end of the Default / Set Value causes the data to word wrap if the updated information is longer than the character width of the dictionary item or field being updated.

For example, the description of products in the PRODUCT file may not contain information that is required. The mass load in the following example appends the word “HONEYWELL” to the DESC dictionary item. The column data for DESC, the dictionary item being updated, is set to the first multi-valued position, because that is the line of the description that is being updated.

Report Writer/Mass Load Design				
Design ID : PROD.DESC		Created : 08/20/04 By : Nicole Dursi		
File Name : PRODUCT		Total Width : 43		
Title : Product Description Fix				
Col	Dict Item/Formula	Width	Typ	Default/Set Value
1	VIEWER_ID	7	D*	
2*	DESC	35	U	"HONEYWELL ":DESC
	-			

S el Build	Adv Select I on	E cl Dct	C ol Data	Cop V	D el	H dg	3 of 3	Analy Z er
R un Rpt	Dict S um	L abel	O pt	N otes	V iew	B egin Load	Set V al	L o E Files P ath

The pipe sign at the end of the string ensures that you can insert information as the first word of the description without losing any of the description if it exceeds 35 characters. It tells the program to word wrap to the next line if there is not enough space.

Exercise 8.6

1. Using the select criteria assigned by your instructor, append your First Name to the description of your products. Make sure that the description will word wrap

Lesson 9

Dictionary Maintenance and I-Descriptors

Objectives

After you complete this lesson, you will be able to:

- Understand the Dictionary Maintenance Screen
- Understand how to create I-Descriptors
- Use the elements in an I-Descriptor formula
- Test the I-Descriptors you create

Dictionary Maintenance

Before we can create Interpretive Descriptor elements first we must understand all of the elements that make up the Dictionary Maintenance Screen.

Eclipse Dictionary Maintenance									
File Name :									
Dict ID :		Version # :							
Dict Type :		Attr# :		Val# :		Subval# :			
Description :									
Prompt :									
				Data Format					
Justify (L/R) :		() Date		Year :					
Maximum Width :		() Numeric		Decim :		Negs:			
Multi-Valued (Y/N) :		() Y/N Only							
Sub-Valued (Y/N) :		() '*' Only							
Required (Y/N) :		() Time							
Allow Update (Y/N) :		() Word Wrap		Disp Lines:		Max Lines :			
Case Mapping :		() Eclipse Dictionary		Indexed (Y/N) :					
SFA Category :		() Archived							
Format (Output) :									
Update Subroutine :									
Copy	Delete	Valid	I-Desc	EXpand	Prompt	Log	Test	Access	Key
Undo	C10se	ProGrama							

The dictionaries are stored in the file called EDICT which stands for Eclipse Dictionary.

The following tables outline all of the different options available in the dictionary maintenance screen.

Dictionary Field Definitions

The following table describes the fields in the middle portion of the dictionary maintenance screen.

Field	Description
Dict Type	<p>Character representing the dictionary item type.</p> <ul style="list-style-type: none"> • D (Data Element) – Identifies a physical field of data, such as a name, user ID, number, address, description, or quantity. • I (I-Descriptor) – Identifies a symbolic field the system can use for calculating values.
Attr#	<p>Attribute number, which indicates the numerical position of the field within the record for the dictionary ID.</p> <p>This field applies to Data element type dictionaries only.</p> <p>Use the Dictionary Maintenance Summary screen to view the defined dictionary items and corresponding attribute numbers for a file.</p>
Val#	<p>Numerical position of a value in a multi-valued field. For example, in a customer file with six contacts, the third contact would be identified as value 3.</p>
Subval#	<p>Numerical position of a sub-value in a multi-valued field. For example, in a customer file with six contacts, each of which contains a contact name and a phone number, the phone number of the third contact person would be sub-value 2 of value 3.</p>
Description	<p>Description of the dictionary item.</p>
Prompt	<p>Default column heading for the selected data when printed on a report, and the prompt if used in a Report Writer/Mass Load select statement with a variable value.</p> <p>The system populates this field with the Dict ID, but you can enter your own heading or prompt in this field.</p>

Field	Description
Justify (L/R)	<p>Indicate whether this data item should align on the right or left side of the report column.</p> <p>Typically, text is left justified and numbers are right justified. Dates should be right justified.</p>
Maximum Width	Enter the default maximum character width for the value.
Multi-Valued (Y/N)	<p>Indicate whether this dictionary item can accommodate more than one entry.</p> <p>For example, in the entity file the ADDRESS dictionary item is multi-valued, allowing two lines of street address.</p> <p>Note: If a dictionary item identifies a single value or sub-value of a multi-valued field, set this field to N.</p>
Sub-Valued (Y/N)	<p>Indicate whether this dictionary item contains sub-values within a multi-valued field.</p> <p>For example, a record in the customer file can have six contacts, which are multi-values. Each contact has a name and phone number, which are sub-values.</p>
Required (Y/N)	Indicate whether an entry for this dictionary item is mandatory when creating a record in the file.
Allow Update (Y/N)	Indicate whether this dictionary item can be updated using mass load or user-defined screens.
Case Mapping	<p>Indicate whether the data stored in this field is set to Upper Case, Lower Case, Title Case (initial uppercase), or Alpha Only (all alpha characters are uppercase). Press F10 to select an option.</p> <p>Changing the case mapping has a from-this-point-forward effect; the system does not change existing data.</p>
SFA Category	If the displayed dictionary item will be used in the Sales Force Automation application, press F10 and select an SFA category with which to associate this item.
Format (Output)	When needed, enter a standard Pick output conversion code that determines the report display format for the data stored in this field. The codes can be found in Lesson 5 of this document. They can be utilized in the format column when you create a report writer report without the need of entering it here on the dictionary.
Update Subroutine	Identify a subroutine to be used to process the data entered from a screen or mass load to update the field.
Date / Year	<p>If this is a date field, enter an asterisk (*). Define the format of the year by entering the number of digits to be displayed. The number of digits can be 0, 2, or 4.</p> <p>Ensure that the Maximum Width is set to allow for the separators, a two-digit month, a two-digit day, and the selected number of digits for the year. If the Maximum Width is not large enough, the date wraps to the next line.</p>
Numeric / Decim / Negs	If this field is numeric, enter an asterisk (*). You must define the number of decimal places, and indicate whether negative numbers are allowed.



Y/N Only	If this field requires a Y or N response, enter an asterisk (*).
** Only	If this field must contain an asterisk (*) to be activated, enter an asterisk (*.)
Word Wrap Disp Lines / Max Lines	<p>If the data in this field can wrap from one line to the next, enter an asterisk (*).</p> <ul style="list-style-type: none"> • Disp Lines is the number of lines to display on the screen. • Max Lines is the maximum number of lines you can enter for this field.
Eclipse Dictionary	If this dictionary item is part of the core Eclipse system, this option displays an asterisk (*) and you cannot modify it.
Archived	<p>If this dictionary item has been archived, this option displays an asterisk (*).</p> <p>An archived dictionary item is obsolete, but is still being used by various user-defined report writer reports and user-defined screens. If you access a report or screen that uses an archived dictionary item, the system displays a warning. We recommend that you replace it with a current dictionary item, but you can also ignore the warning. Customers who went 'live' on Eclipse on Release 8 will not see dictionary items set to Archived.</p> <p>An archived dictionary item does not display the first time you press F10 to add items to mass load, report writer, order entry prompts, and user-defined screens. Press F10 again to access archived items.</p>

[illegible]

Hotkeys in Dictionary Maintenance Screen

Hot Key	Function
Copy	Creates a dictionary item for the displayed file or another file, by copying the displayed dictionary item.
Delete	<p>Deletes the displayed dictionary item from the dictionary file. The system prompts you to confirm the deletion.</p> <p>Note: You cannot delete a dictionary item that is open for editing. The system deletes the latest revision, but does not delete the dictionary item.</p>
Valid	Makes the dictionary item a validated field. Validating data can be accomplished using a manual list, from a control record that contains the list, another file's @ID or a verification subroutine that will verify the data.
I-Desc	Displays the I-Descriptor Program Maintenance screen, where you can enter a formula that determines the value of this dictionary item.
Expand	<p>Displays an expanded view of the Prompt or Description field.</p> <ul style="list-style-type: none"> The expanded Prompt field contains 3 lines the same width as the value in the Maximum Width field. The expanded Description field contains 3 lines that are 60 characters wide. <p>You can erase the displayed text and type new text for this field. Use this screen to determine whether the text fits on one line or wraps to a new line.</p>
Prompt	<p>Lists the standard prompts that you can use with this dictionary item.</p> <p>When creating a dictionary item, press F10 to display the list of available prompts.</p>
Log	Displays the Maintenance Log Viewing screen for the displayed dictionary item.
Test	Selects a record from the file and displays the output defined by this dictionary item.
Access	Displays the Access Control List screen, where you can designate user IDs or group IDs that can access the data in this record from a laptop or Palm computing device and the level of access for each ID.
Key	<p>Use to identify the names of files for which this dictionary item is the key.</p> <p>For each entry, press F10 and select a file name.</p>
Undo	Eclipse Dictionaries are now being tracked with version numbers. If the undo option is used on a dictionary item that is a Version 1, the system will delete the dictionary as if you used the delete hotkey. If however, the dictionary item is a 2 nd or 3 rd version, the



	undo option would back out the version to the previous version number.
Close	If you edit a dictionary item in the system that is not marked as an Eclipse Dictionary (those will be view only) the system will keep the dictionary open to you until you have completed it. When you are done, you should close the dictionary. You will know if the dictionary is open or closed based upon the “Open To: “, message that will appear in the upper right hand corner.
Program	<p>The program hotkey will open a window to create a subroutine that will exist only at the dictionary item level. Any subroutine created here will not be a Basic Program but a Dictionary Program stored at the dictionary level.</p> <p><i>Note: Programming dictionaries will not be discussed in this class. This option is instructed in the “Basic Programming” class.</i></p>

[illegible]

Creating an I-Descriptor

An I-Descriptor (interpretive descriptor) is a symbolic field derived from real data or a formula applied to real data. You can use I-Descriptors to manipulate strings of information or calculate values for selecting, sorting, and displaying data on reports.

For example, in the ENTITY file the dictionary item called ADDR.LINE2 is an I-Descriptor that refers to the second line of the entity's address. The formula that defines this I-Descriptor selects the second value of the second attribute in the record.

Use the Eclipse Dictionary Maintenance screen to create I-Descriptors.

Eclipse Dictionary Maintenance			
File Name	: ENTITY		
Dict ID	: ADDR.LINE2		
Dict Type	: I	Attr#	: 0
Description	: ADDRESS LINE 2		
Prompt	:		
Val#	:	Subval#	:
Justify (L/R)	: L	() Date	Year :
Maximum Width	: 35	() Numeric	Decim : Negs:
Multi-Valued (Y/N)	: N	() Y/N Only	
Sub-Valued (Y/N)	: N	() '*' Only	
Required (Y/N)	: N	() Time	
Allow Update (Y/N)	: N	() Word Wrap	Disp Lines: Max Lines :
Update Subroutine	:	() Eclipse Dictionary	
Case Mapping	:		
Format (Output)	:		
SFA Category	:		
C opy	D elete	V alid	I -Desc
E xpand	P rompt	L og	T est
A ccess	K ey		

Fill in the appropriate **File Name** and **Dict ID** for the I-Descriptor. Enter the letter **I** in the **Dict Type** field. An I-Descriptor describes a symbolic field for a file. Fill in the data format to determine how the output of the I-Descriptor will be displayed when the I-Descriptor formula is processed.

Use the **I-Desc** hot key to open the I-Descriptor Program Maintenance screen, and then enter the I-Descriptor formula.

I-Descriptor Program Maintenance				
File Name : ENTITY				
Dict ID : ADDR.LINE2				
@RECORD<2,2>				
Test	Find Subr	Edit Subr	Set Common	

Elements in an I-Descriptor Formula

An I-Descriptor formula can contain any of the following elements:

- Field names
- Operators
 - Arithmetic
 - Relational
 - Logical
 - Conditional
- Constants
 - Numeric
 - String
- Internal variables
- Substring extraction expressions
- File transfer function (the TRANS function)
- Other BASIC functions

Note: Do not use symbols or operators in your I-Descriptor names.

Field Names

An I-Descriptor can refer to data in another field by specifying the dictionary item name in the formula. The dictionary item must refer to an element that is defined in the dictionary file where the I-Descriptor is compiled. When Eclipse processes the I-Descriptor formula, the current data value of the dictionary item is used.

For example, an I-Descriptor formula that uses the values from the **Surplus** and **Rep Cost** fields to determine the value of the surplus, uses the names of these dictionary items in the formula as follows: (SURPLUS * REPCOST).

Operators

I-Descriptor s can use arithmetic, relational, logical, and conditional operators in their formulas. To avoid unintentional conflicts with operators, Eclipse suggests that you do not use symbols or operators in your dictionary names. For example, use BAL.60.PLUS instead of BAL.60+ and use PROD.NUMBER instead of PROD#.

Operators are listed in the table below in order of precedence. Multiple symbols exist for operators of the same description and may be used interchangeably. Eclipse suggests always using parentheses in a formula to control operator precedence.

Operator	Description
** ^	Exponentiation
*	Multiplication
/	Division
+	Unary plus
-	Unary minus
+	Addition
-	Subtraction
: CAT	Concatenation
< LT	Less than
> GT	Greater than
= EQ	Equal to
NE <> # ><	Not equal to
<= =< LE	Less than or equal to
#>	Not greater than (same as LE)
>= => GE	Greater than or equal to
#<	Not less than (same as GE)
MATCHES MATCH	String matches pattern
AND &	Performs a logical AND function on two formulas to produce a true (1) or false (0) result. If <i>both</i> formulas are true, the formula evaluates to 1 (true).
OR !	Performs a logical OR function on two formulas to produce a true (1) or false (0) result. If <i>either</i> formula is true, the formula evaluates to 1 (true).

Testing I-Descriptors

Use the **Test** hot key on the Eclipse Dictionary Maintenance or I-Descriptor Program Maintenance screen to test an I-Descriptor. This displays the Dictionary Testing screen, where you can enter record IDs from the file for which the I-Descriptor is defined.

Note: When common data is required for an I-Descriptor, use the **Set Common** hot key to enter that data before using the **Test** hot key.

Dictionary Testing			
File IDs	:		
Show Blank Values	:	Y	
B egin	I Ds		F12-Abort

After entering the record IDs for the test and indicating whether to list fields whose values are blank, use the **Begin** hot key to do the test. The system displays the contents of the field described by this dictionary item for each test record.

*In this class, due to the volume of students we ask that you test the dictionary items you create by using the **LIST** command in **TCL**.*

Exercise 9.1

Find an I-Descriptor in the file assigned by your instructor and test the dictionary item.



Lesson 10

Creating Mathematical I-Descriptors

Objectives

After you complete this lesson, you will be able to:

- Formulate mathematical expressions using Dictionary Maintenance
- Test your dictionary items

I-Descriptors that use a Mathematical Formula

The I-Descriptor formulas in the following examples use mathematical operators. When dealing with a mathematical formula, be aware of the decimal positions. If dictionary items in a formula do not have the same number of decimal positions, you may not get the desired result.

Example 1

In this example the mathematical formula is derived from two existing dictionary items. Between the dictionary items is the mathematical function that takes place. The I-Descriptor called OVER_CRLIMIT is calculated by subtracting the output of the dictionary CR_LIMIT from the output of the dictionary AR_BAL.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	OVER_CRLIMIT
AR_BAL - CREDIT_LIMIT	

Example 2

This example also creates a formula using two dictionary items. The COST.DIFF is calculated by subtracting the output of FIRST.COST from the output of LAST.COST.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	COST.DIFF
LAST.COST - FIRST.COST	

This dictionary item would be an archived dictionary if you were upgraded from a Release 7 to a Release 8 account.

Example 3

The formula in this example utilizes three dictionary items. It calculates the sum of the three values.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	BAL.OVER60DAYS
BAL.60DAYS+BAL.90DAYS+BAL.120DAYS	

Example 4

The asterisk is used for multiplication. The formula in this example multiplies the values of two existing dictionary items.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SURPLUS\$
SURPLUS * REPCOST	

Example 5

The following formula adds the values of two dictionary items. Note that there is a minus sign before the first dictionary item.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	TOTAL.INV.AMT
-INV.AMT+TAX.AMT	

If you view the raw data in the PSUB file, the sale amount of outgoing inventory displays as a negative value. The minus sign in this formula changes the value of the dictionary item to a positive number. Therefore, the output of the INV.AMT becomes a positive value, which is then added to the TAX.AMT

```
@ID..... QTY.....
2076~1~08690~$0265054~1~2~$~CTR          -4
4047~1~08675~$0164968~1~1~$~CTR          -1
5412~1~08511~$0669605~1~14~$~CTR        -20
.....
```

Example 6

The following formula uses multiple operators. You are not forced to create formulas based only on dictionary items that already exist. In this example a hard-coded text string of information is used within the formula. The formula calculates the Gross Profit Percentage by multiplying the value of the GP\$.CMP dictionary item by 1000, and then dividing that amount by the value of the SUB.AMT dictionary item.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	GP%(CMP)
GP\$.CMP * "1000" / SUB.AMT	

Example 7

The final example shows two subroutines used in a subtraction formula.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	GP\$.CMP
SUBR('DICT.GET.LEDGER.VALUE',15) - SUBR('DICT.CMP.COST.CALC')	

Exercise 10.1

1. Create a dictionary item in the PRODUCT file that multiplies the on-hand quantity by the replacement cost of the product.
2. Test your dictionary item using the **Test** hot key or using the LIST command in TCL.

Lesson 11

Creating Internal Variable I-Descriptors

Objectives

After you complete this lesson, you will be able to:

- Create dictionary items that return values or sub-values of an attribute
- Use the @ID variable in an I-Descriptor
- Create I-Descriptors using internal variables
- Use the @RECORD, @ID, and @VM internal variables

Internal Variables

An I-Descriptor can use a number of pre-defined variables within the formula. Some of these variables are “short cuts” for other functions.

Internal Variable	Returns...
@n	the value of the previous expression, where <i>n</i> is a sequential number. That is, the result of the first expression is assigned to internal variable @1, the result of the second expression is assigned to internal variable @2, etc.
@AM	an attribute mark (the same as using the function CHAR(254)).
@ID	the record identifier (also known as the Item ID or the Record ID), which is the “key” to the record. For example, in the Customer file, @ID returns the customer number; in the Product file, @ID returns the product number.
@RECORD	the entire record, including attributes, multi-values and sub-values. @RECORD is a dynamic array (a table of data) containing these elements.
@SVM	a sub-value mark (the same as using the function CHAR(252)).
@VM	a value mark (the same as using the function CHAR(253)).

Using the @RECORD Variable

The @RECORD variable can return an entire record, an entire attribute in a record, or a value within an attribute.

Example 1

This dictionary item returns the first attribute of a record in the PRODUCT file. This returns all of the information in the attribute, regardless of whether it is multi-valued

I-Descriptor Program Maintenance				
File Name : PRODUCT Dict ID : DESCR				
@RECORD<1>				
Test	Find Subr	Edit Subr	Set Common	

Example 2

This dictionary item returns a product's UPC number, which is stored in the first value of attribute 63 of each PRODUCT file record.

Attribute 63 stores the UPC in the first value, User Defined #1 in the second value, User Defined #2 in the third value, User Defined #3 in the fourth value, and User Defined #4 in the fifth value. This I-descriptor is saying, go to attribute 63 in the PRODUCT file and return only the value that is stored before the first value marker.

I-Descriptor Program Maintenance View Only				
File Name : PRODUCT				
Dict ID : UPC				
@RECORD<63,1>				
Test	Find Subr	Edit Subr	Set Common	

Example 3

This dictionary item returns the 3rd multi-valued position of attribute 26 of each PRODUCT file record.

I-Descriptor Program Maintenance				
File Name : PRODUCT Dict ID : CREATED.BY				
RECORD<26,3>				
Test	Find Subr	Edit Subr	Set Common	

The following TCL display of raw data shows a PRODUCT file record. Attribute 26 displays 3 value markers, which indicate that there are 3 sets of information in this attribute. The 3rd value of this string shows us that ARTHURW created this product. The 1st and 2nd values show the date and time the product was created.

```

101316
001 3/4-10 X 6 1/2 HEX CAP SCREW PKG*GR-5 ZINC PLTD
002 IND
003 2
004 46347*FASTENERS
007 1
008 C
009 FAST
012 FAST
026 11432*51078*ARTHURW*
063 *46347
073 FASTNS*FASTNS

```

Example 4

Not only is the following dictionary item returning the specific information from the ENTITY file, but it is performing a BASIC function as well.

If @RECORD<7,1> contains a value of 1, the record is a bill-to. If @RECORD<7,2> contains a value of 1, the record is a ship-to. A customer can be a bill-to, ship-to, or both. The following dictionary item displays a value of 1 for ENTITY records that are flagged as being a ship-to-only customer; otherwise, it displays a value of 0.

I-Descriptor Program Maintenance				
File Name : ENTITY				
Dict ID : SHIPTO.ONLY				
@RECORD<7,1> = "" AND @RECORD<7,2> = "1" OR @RECORD<7,1> = "0" AND@RECORD<7,2> = "1"				
Test	Find Subr	Edit Subr	Set Common	

The following screen shows a customer that is flagged as a ship-to customer.

Customer Maintenance	
Customer/New : 41477	41477
Name :ASC Address:13222 gROSSBECK City :ROSEVILLE Zip :48066 ST:MI Country: Sort By:ASC Bill :Argus Group Index :ASC	() Bill To (*) Job or Ship To () Branch (*) Branch Cash Acct () Prospect (P) PO/Release# Required () Auto-Delete
Contacts 1. 2. 3. 4. 5. 6.	Phones Bal Fwd/Open Item: B/O Status :C Out Salesperson :HOUSE In Salesperson :HOUSE Ship Via:PN NORWALK Frt In Exempt (Y/N) : N Frt Out Exempt (Y/N): N

Following is a partial TCL listing of the raw data in this customer's record. The asterisk from the **Job or Ship To** field above is reflected as a **1** in the second value of attribute 7.

```

41477
0001 ASC
0002 13222 gROSSBECK*
0003 ROSEVILLE
0004 MI
0005 48066
0006
0007 1 1 1 1 1 1 1

```

Exercise 11.1

1. Using your initials and the word BILLTO, create a bill-to-only dictionary item for the ENTITY file, similar to the ship-to-only dictionary item described above.
2. Test the dictionary item using @ID 41725. This is a bill-to-only customer in the ENTITY file.

Using the @VM Variable

Using @VM inserts a carriage return in the dictionary item. In the following example, L#16 defines the number of left-justified character spaces to be allocated for displaying a piece of data.

The following dictionary item uses an IF THEN ELSE statement (discussed in more detail later) to display a customer's name and address. If a second address line exists, the dictionary item displays four lines of information; otherwise, it displays three lines of information. The city, state, and Zip code are concatenated and displayed on the same line. Because 16 character spaces are allocated for each city name, the state and Zip codes align. The @VM variable inserts the line breaks.

I-Descriptor Program Maintenance				
File Name : ENTITY				
Dict ID : NA				
<pre>IF @RECORD<2,2> THEN @RECORD<1>:@VM:@RECORD<2,1>:@VM:@RECORD<2,2>:@VM:@RECORD<3>"L#16":@RECORD<4>"L #3":@RECORD<5> ELSE @RECORD<1>:@VM:@RECORD<2,1>:@VM:@RECORD<3>"L#16":@RECORD<4>"L#3":@RECORD<5></pre>				
Test	Find Subr	Edit Subr	Set Common	

The following list shows the output for this dictionary item.

17059	JOE BENNETT	
	12345 MAIN STREET	
	MAHOPAC	NY 10541
19497	ECLIPSE PLAY ACCT	
	12345 MAIN STREET	
	CROTON	NY 10520
17060	ECLIPSE PLAY ACCT	
	12345 MAIN STREET	
	PELHAM	NY 10803
14623	ECLIPSE PLAY ACCT	
	12345 MAIN STREET	
	SCARSDALE	NY 10583

This dictionary was used when we created our mailing labels report.

Exercise 11.2

1. Using your initials and the word PROD as the dictionary ID, create a dictionary item that will display three lines of information.
 - On the first line, display the first line of the description.
 - Leave the second line blank.
 - The third line should display the price line of the product, the status of the product and first sell group of the product.

Make sure that the third line of information is spaced properly.

2. Test the dictionary item.

The test output should be similar to the following example:

```
11494      VOW-V-60-D OAK DRAWER BASE
          BER      2      BER
```



Lesson 12

Creating Field Command I-Descriptors

Objectives

After you complete this lesson, you will be able to:

- Create I-Descriptors using the FIELD function

FIELD Function

The FIELD function takes part of a delimited string and returns the specified occurrence of the contents within the string. The syntax is:

FIELD(String,Delimiter,Position)

- **Field** is the command being used.
- **String** is the element from which we are extracting a piece of information.
- **Delimiter** is the character or space that separates data within the string.
- **Position** is a number that represents where the data is stored within the string.

Example

The following I-Descriptor formula extracts the 2nd word of the description in the PRODUCT file. The FIELD function states the following three pieces of information:

- The string of data from which you will obtain the information is the first line of the first attribute in the record.
- The character that delimits words in the description is a space.
- The word you want is in the 2nd position.

I-Descriptor Program Maintenance			
File Name : PRODUCT			
Dict ID : DESC.WORD.2			
FIELD(@RECORD<1,1>,' ',2)			
Test	Find Subr	Edit Subr	Set Common

The following TCL listing shows the values that would be listed for this dictionary item:

@ID.....	DESCRIPTION.....	DESC.WORD
101467	1/2 X 1 1/2 CARR SCR BULK GR-5	X
	ZINC PLTD	
50810	CENTER MIRROR FOR BMSXSCTVL36	MIRROR
17038	BA-711-WH SOAP DIS. (RP-011)	SOAP
25481	P.BRASS 6" TUB SPOUT	6"
101468	1/2 X 1 1/2 CARR SCR PKG GR-5 ZINC	X
	PLTD	
50811	LEFT MIRROR FOR BMSXSCTVL48	MIRROR
101469	1/2 X 1 3/4 CARR SCR PKG GR-5 ZINC	X
	PLTD	
25482	CHROME 6" TUB SPOUT	6"
153	1/2X3/4 CXF ELL 707-3 4707-3R	CXF
50812	RIGHT MIRROR FOR BMSXSCTVL48	MIRROR
101470	1/2 X 2 CARR SCR BULK GR-5 ZINC	X
	PLTD	

Using the PSUB File

When an order is created, the order information is stored in the LEDGER and ORDER.QUEUE files. Once an order is processed, the summary information moves from the ORDER.QUEUE file to the AR (Accrual Register) file and the product detail information moves to the PSUB (Product Subsidiary) file.

The PSUB file contains line-item detail for closed ledger transactions, such as sales invoices, received purchase orders, transfers, and inventory adjustments. Each product on a transaction creates one record in this file. Each product creates two records if the shipping branch is different from the pricing branch.

```
LIST PSUB 10:49:24am 28 Nov 2001 PAGE 1
@ID.....
```

```
2076~1~08690~S0265054~1~2~S~CTR
4047~1~08675~S0164968~1~1~S~CTR
5412~1~08511~S0669605~1~14~S~CTR
1700~1~08452~S0665680~1~11~S~CTR
12127~1~11171~S0417511~1~25~S~WHSE
34354~1~11750~S1000131~1~4~S~WHSE
```

The key to the record is comprised of a concatenation of information unique to an inventory movement, as shown in the following example. A tilde (~) delimits items that comprise the record ID.

```
40929~1~11499~S1000000~1~2~S~WHSE
```

These items contain the following information each field separated by a tilde:

- **Field 1** – PRODUCT_ID = product Id
- **Field 2** – BR = branch on transaction
- **Field 3** - SHIP_DATE = date material shipped
- **Field 4** – ORD_ID = order number for the transaction
- **Field 5** – INVOICE_NBR = the order generation for the transaction
- **Field 6** – LED_DET_ID = the position of the product on the transaction
- **Field 7** – QTY_TYP = the type of material that shipped (Stock, Defective, Tagged etc). If the order is a direct shipment this field will be a “D”.
- **Field 8** – LOCATION = the bin location the material shipped from or was received into. If the order is a direct shipment this field will be the internal id for the vendor.
- **Field 9** – COMPONENT_POS = the component part numbers if the item was a kit.
- **Field 10** – DIFF_BR = the shipping branch if it is different from the BR (field 2)

Example 1

The following dictionary item returns the Internal Part number from the @ID of the PSUB record. This dictionary item can be used as the key to obtain access to the PRODUCT and other files where the @id of the file is the internal ID of the PRODUCT.

I-Descriptor Program Maintenance — View Only	
File Name :	PSUB
Dict ID :	PRODUCT_ID
FIELD(@ID, '~', 1)	

Example 2

The following dictionary item selects the order number from the @ID of the PSUB file. This dictionary can be used as the key to obtain access to the LEDGER file.

I-Descriptor Program Maintenance — View Only	
File Name :	PSUB
Dict ID :	ORD_ID
FIELD(@ID, '~', 4)	

Example 3

The DIFF_BR dictionary item in the next example can be used to isolate products that are shipping out of a different branch than the pricing branch.

I-Descriptor Program Maintenance — View Only	
File Name :	PSUB
Dict ID :	DIFF_BR
FIELD(@ID, '~', 10)	

The product serial number is also stored in the PSUB file; however, it is in a file attribute rather than the @ID of the file.

Exercise 12.1

1. From the ENTITY file create a dictionary item that returns the 2nd word of the first line from the Address field.

Exercise 12.2

1. From the PRODUCT file, create a dictionary item that returns the 3rd word of the second line of Keywords.



Lesson 13

Creating TRANS Command I-Descriptors

Objectives

After you complete this lesson, you will be able to:

- Create and use TRANS command I-Descriptors

TRANS Function

The transfer functionality in dictionary maintenance or the ability to access data from another file is very similar to the Files Option on your report writer / mass load screen.

The TRANS function accesses data in another file and returns a value for use in the I-Descriptor formula. In other words, it “transfers” the data.

You can do this if a field in the source file is a “key” for records in the other file. For example, you are running a report from the PSUB file and want to display the customer name, which is stored in the CUSTOMER file. A field in the PSUB file contains the customer number. You can use this number to access the corresponding record in the CUSTOMER file and then transfer the customer name from that record back to the PSUB file.

Example 1

When running a report from the PSUB file you want to list the customer name with each transaction, but the name is not stored in the PSUB file. So, you need to create a dictionary item in the PSUB file that uses the TRANS function to transfer the name from the CUSTOMER file. The syntax of the TRANS function is as follows:

TRANS(FILENAME,KEY,ATTRIBUTE #,“X”)

- **Trans** is the command.
- **Filename** is the name of the file that contains the information you require.
- **Key** is the string of data from the file you are currently in that matches the @ID or Key of the records in the file from which you want to obtain information.
- **Attribute** is the attribute number in the record from which you are retrieving the information where the required data is stored.

- The transfer code “X” indicates that if the attribute contains no data, TRANS returns a null value. Use the transfer code “C” to instruct the function to return the @ID for debugging purposes.

In the following screen, the TRANS function uses the data in the 4th field of the PSUB record to access the correct record in the ENTITY file and return the value of attribute number 1.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	NAME
TRANS('ENTITY',@RECORD<4>,1,'X')	
-	

Example 2

In this example, from the PRODUCT file you access the BUY.LINE file and retrieve the name of the buyer. The key to the BUY.LINE file is contained in attribute 12 of the PRODUCT file. The buyer's name is stored in attribute 17 of the BUY.LINE file.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	BUYER
TRANS('BUY.LINE' ,@RECORD<12>,17, 'X')	

If a dictionary item corresponds to the key to another file, then that dictionary item can be used instead of the @RECORD statement. The dictionary item BUY_LINE in the PRODUCT file, points to attribute 12.

Example 5

The following TRANS function accesses the ENTITY file using the 15th attribute in the AR file as the key. It returns the value of attribute number 17 from the ENTITY file. This is the multi-valued PHONES attribute. It then uses the <1,1,2> sub string extraction expression to select the second phone number.

<x,y,z> allows you to extract the “x” **file**, the “y” **value**, and the “z” **multi-value** from the record. You do not need to specify x, y, and z if you only need to retrieve x and y (<x,y>) or just x (<x>). Use the <x,y,z> with the @RECORD variable to return specific elements of a record.

When using the <x,y,z> on a trans routine, you must define all three segments to pinpoint the value marker you want to return as shown below.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	SHIPT0.PHONE
TRANS(ENTITY,@RECORD<15>,17,"X")<1,1,2>	

Exercise 13.1

1. From the AR file create a dictionary item that returns the outside salesperson from the ENTITY file.

Exercise 13.2

1. From the AR file, create a dictionary item that returns the writer from the LEDGER file.

Note:

The key to the LEDGER file consists of just the first 8 characters of the @ID of the AR file. You need to create a dictionary item that defines this key. Use the following FIELD statement to return the first 8 characters:

```
FIELD(@ID,'.',1)
```

The LEDGER file holds multiple generations. Information can be different in the header screen for each generation. Eclipse has created subroutines that work through the multiple generations to pull back the information that you are looking for.

Using the ORDER.QUEUE File

Use the ORDER.QUEUE file to run reports on all the open orders in the system. These orders have not yet gone to the AR or the PSUB file. This is a summary file that doesn't contain many details. Therefore, you need to create I-Descriptor dictionary items that transfer detail information from other files.

```

      $1013863.12058.1
001 B
002 1
003 1
004 -166000
005 -106245
006 -106245
007 41715
008 41715
009 TAMMYF
010 PETERA
011 TAMMYF

```

The following I-Descriptor uses attribute 7 (the bill-to customer ID) of the ORDER.QUEUE file as the key to access the ENTITY file and then transfer attribute 1 (the customer name) from that file.

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	BT.NAME
IRANS('ENTITY',@RECORD<7>,1,'X')	

Using the PSUB File

The PSUB file contains line-item detail for closed ledger transactions, such as sales invoices, received purchase orders, transfers, and inventory adjustments. Each product on a transaction creates one record in this file. The system creates two records for each product if the shipping branch is different from the pricing branch.

The following example shows a record from the PSUB file. The record key is comprised of a concatenation of information unique to an inventory movement. The customer ID associated with the transaction is stored in the 4th attribute.

```

2076~1~08690~S0265054~1~2~S~CTR
001 -4
002 3784000000
003 2247000000
004 31458
005 31458
006 1
007 1

```

The following I-Descriptor formula uses attribute 4 as the key to the ENTITY file and transfers attribute 1 (the customer name) of that file back to the PSUB file.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	NAME
TRANS('ENTITY',@RECORD<4>,1,'X')	
-	

The following I-Descriptor uses a portion of the PSUB record ID to access the LEDGER file and return the value in attribute 13 (the customer's P/O number).

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	PO.NUMBER
TRANS('LEDGER',FIELD(@ID,'~',4),13,'X')	

The next I-Descriptor uses a different portion of the PSUB record ID to access the PRODUCT file. The following example returns the product description from attribute 1.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	ITEM.DESC
TRANS('PRODUCT',FIELD(@ID,'~',1),1,'X')	

Using a Double TRANS Command

The following example shows how to use a double TRANS command.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	BANK
TRANS(GENLED,FIELD(TRANS(LEDGER,FIELD(@ID,'.',1),13,'X'),'~',1),3,'X')	

The value obtained from the inner TRANS command is used as the key to records in the file named in the outer TRANS command. The inner command uses the first value of the AR file's @ID as the key to records in the LEDGER file. The command obtains the value from attribute 13 in LEDGER file. The outer command then uses this value as the key to records in the GENLED file. Once there, the command retrieves the bank description, which is stored in attribute 3.

Using Notes Files

On the Vendor Maintenance and Customer Maintenance screens there is a hot key called **Notes** that you can use to enter internal notes. Information entered in this field is stored in the ENTITY.NOTES file. The key (record ID) for records in this file is the same as the key for the customer and vendor records in the ENTITY file.

The following example uses the @ID of the ENTITY record to retrieve attribute 3 from the ENTITY.NOTES file.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	1099.REQ
TRANS(ENTITY.NOTES,@ID,3,'X')	

You can also go the other way. The following example uses the key of the PRODUCT.NOTES file to access the PRODUCT file and transfer the value in the 9th attribute.

I-Descriptor Program Maintenance	
File Name :	PRODUCT.NOTES
Dict ID :	PRICE.LINE
TRANS("PRODUCT",@ID,9,'X')	

Exercise 13.3

1. From the PSUB file, create a dictionary item that returns the weight of the product.

Exercise 13.4

1. From the ORDER.QUEUE file create a dictionary item that returns the ship-to Fax number.

Exercise 13.5

1. From the PRODUCT file create a dictionary item that returns the buyer's name.



Lesson 14

Basic Functions

Objectives

After you complete this lesson, you will be able to:

- Use the basic functions LEN, TRIM, OCONV, and STR
- Write a conditional expression using IF, THEN, and ELSE

Basic Functions

Many BASIC functions can be used in I-Descriptor formulas. The same syntax applies whether using a BASIC function in an I-Descriptor or in a BASIC program.

The following topics describe a few commonly used functions.

LEN Function

The LEN function returns the length of a string. For example, if a product's description is "BLACKFORD 10661615 WATERHEATER" and an I-Descriptor formula uses the function LEN(DESC), the value returned is 30.

The I-Descriptor formula in the first example returns the character length of the product's buy line. You can use this function to identify records in which the buy line contains too many characters.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	JH.LEN.LINE
LEN(BUY.LINE)	

The I-Descriptor formula in the second example adds the length of the A/R record, the length of the record ID, and the number 9. This formula calculates the number of bytes in the record.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	SIZE
LEN(@RECORD)+LEN(@ID)+9	

The following listing shows the output of this dictionary item.

```

@ID..... SIZE.
S0671864.001 154
S0421802.001 138
S0418573.002 165
S0371864.001 138
S1008886.005 207
S1012115.004 234
S0675093.001 156
S1000000.001 125
S1008886.006 127
S0703229.001 152
S0128260.001 102
S1003229.001 129
S0678322.001 130
C0003229.001 4653

```

IF THEN ELSE Operators

Use the IF/THEN/ELSE operators in I-Descriptor formulas to create a conditional statement. The basic syntax is:

IF formula one THEN formula two ELSE formula three

If the first formula is true, then process the second formula. If the first formula is false, then process the third formula. Both the THEN and ELSE operators are required by the formula syntax.

Example 1

The conditional formula in the following example, displays the data from the city, state, and Zip code dictionary items on one line. The city is concatenated to a comma and space, the state, another comma and space, and then the Zip code. Use this I-Descriptor to display the information in one column of a report rather than three separate columns.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	CSZ
IF CITY NE "" THEN CITY:", ":STATE:" ":ZIP ELSE ""	

Example 2

The I-Descriptor in the following example enables you to sort your CUSTOMER file by bill-to, by ship-to, by customer Sortby order. This dictionary is used primarily for sorting data for your reports. You can also use it to sort your records when listing them on screens in Eclipse. After each bill-to customer, the ship-to customers are listed in order according to the value in the customer record **Sortby** field.

The formula for this I-Descriptor consists of three expressions. You can enter each on a separate line or enter them on one line with each expression separated by a semicolon, as shown in this example.

I-Descriptor Program Maintenance	
File Name :	CUSTOMER
Dict ID :	BT.ST.SORT
IF ENTITY.TYPE.1 = "1" THEN "1" ELSE "2"; @RECORD<8>; @1:@2	

In the first expression, the ENTITY.TYPE.1 dictionary item is the flag that indicates whether the customer is a bill-to or not. When the flag is set to "1" the record is a bill-to. When the flag is set to "0" the record is a ship-to. If the dictionary item contains a "1," then assign a value of "1" to this expression. Otherwise, assign a value of "2" to the expression.

The second expression @RECORD<8> identifies the field that contains the SORTBY information in the Customer file.

The third expression @1:@2 concatenates the value of the first expression ("1" or "2") with the value of the second expression (the data from the **Sortby** field).

**ACTIVANT™**

Entering this I-Descriptor on the File Definition Maintenance screen in the **Select SortBy Dict ID** field enables you to sort your index differently.

File Definition Maintenance			
File Name	: CUSTOMER		
Description	: Customer File		
Physical File	: ENTITY		
Parent File	: ENTITY		
Dictionary File:	CUSTOMER	Type :	
Maint Logging	: 4 - Save Deleted Items		
Log Change Rsn	: Y	Min Days Before Purge	: 730 Min # Logs to Save : 100
Select Index Dict ID :		Hot Sync (Y/N) :	
Select SortBy Dict ID:	BT.ST.SORT	Prevent Mass Load (Y/N) :	N
Disp Conv Expr/Attb	: TCUSTOMER;X;1;1		
Input Validation Subr:	VERF.CUS.ID		
Pre-Index Conv Subr	: DICT.SOUNDA		
Update Validation	:		
Update Subr	:		
Select Filter Subr	:		
New ID Verification	: No New		
Keep File in Sync With Parent (Y/N)	: N		
Dict Maint	Delete	Branch Specific	F12-Abort

Using IF THEN ELSE to Fix a Problem

What can we do to fix the following I-Descriptor formula?

I-Descriptor Program Maintenance	
File Name	: AR
Dict ID	: GP%(CMP)
GP\$.CMP * "1000" / SUB.AMT	

This dictionary returns errors if the value is zero.

I-Descriptor Program Maintenance	
File Name	: AR
Dict ID	: GP%(CMP)
IF SUB.AMT # '0' THEN GP\$.CMP * "1000" / SUB.AMT ELSE '0'	

Use the IF, THEN, ELSE statement to make sure no errors occur.

Examples of other IF THEN ELSE Dictionaries

From the Product File:

I-Descriptor Program Maintenance				
File Name : PRODUCT Dict ID : PURCH.UOM				
IF @RECORD<16,2> # "" THEN @RECORD<16,2> ELSE "1"; FIELD(@RECORD<7>,@VM,@1)				
Test	Find Subr	Edit Subr	Set Common	

The above dictionary will pass back the correct quantity value based upon the UOM flag on the front screen of PFM for Purchasing.

Then it will use this to determine what the actual UOM Qty is based upon that flag.
This dictionary * 1 will provide an accurate mass load to the buy package quantity.

From the Entity File:

I-Descriptor Program Maintenance				
File Name : ENTITY Dict ID : BRCH_SALES				
IF SUBR('DICT.CALC.SLS',1,1,1,'','') = SALES THEN "NO" ELSE IF SUBR('DICT.CALC.SLS',1,1,2,'','') = SALES THEN "NO" ELSE IF SUBR('DICT.CALC.SLS',1,1,4,'','') = SALES THEN "NO" ELSE IF SUBR('DICT.CALC.SLS',1,1,5,'','') = SALES THEN "NO" ELSE IF SUBR('DICT.CALC.SLS',1,1,6,'','') = SALES THEN "NO" ELSE IF SUBR('DICT.CALC.SLS',1,1,7,'','') = SALES THEN "NO" ELSE "YES"				
Test	Find Subr	Edit Subr	Set Common	

The above dictionary narrows down a selection of customers who purchase from more than one branch in the company.

The above dictionary item is sweeping through the branches to see if the sales that occurred in a branch match the total sales for all of the other branches. If the sales do not match then the customer is one who purchases from multiple branches.

Each line represents a branch. For this company they only had branches 1,2,4,5,6 and 7. They do not have a branch 3.

Character Strip

Example 3

The I-Descriptor formula in this example states if the record in the PSUB file is a sales order, then multiply the quantity by the unit cost; otherwise, multiply the quantity by the unit sell price. Because purchase orders do not have a unit cost, you can use this dictionary item for both purchase orders and sales orders.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	EXT.COST
IF OID(1,1)='S' THEN (QUANTITY*UCOST)/1000000 ELSE (QUANTITY*USELL)/1000000	

We divide each of these values by 1000000 to place the decimal point in the correct position. Eclipse stores the unit price and cost out to nine characters, as shown in the following listing of raw data.

```

2076~1~08690~S0265054~1~2~S~CTR
001 -4
002 3784000000
003 2247000000
004 31458
005 31458
006 1
007 1

```

Example 4

The I-Descriptor formula in this example states that if the A/R record is a sales order, use a subroutine that calculates the outgoing freight; otherwise, use the incoming freight. The conditional statement enables you to use this dictionary item for both purchase orders and sales orders.

I-Descriptor Program Maintenance	
File Name : AR	
Dict ID : FREIGHT	
IF @ID[1,1] = 'S' THEN -SUBR('DICT.ASUB.AMT', 'FGHT') ELSE FRT.IN	

Exercise 14.1

1. From the ENTITY file, create a dictionary item that will return a TOTAL.CURRENT balance. It should include the Future A/R bucket, Current A/R bucket, and Deposits.

Exercise 14.2

1. From the PRODUCT file, create a dictionary item that displays the price line if the product is a stock product and the buy line if the product is a nonstock product.

Exercise 13.3

1. Write a report from the PRODUCT file that shows the product's inventory value at Replacement Cost and also Average Cost. Show the difference between the two costs on the report.

STR Function

The STR function repeats a character string as many times as you want. You can use this function to create separators in reports. The data within the quotation marks is the string of characters that will print to the screen. This is followed by the number of times you want this string to appear.

The following example prints happy faces for everyone!

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	DASH
STR(":-)",70)	

Use the **Test** hot key to view the results produced by an I-Descriptor formula.

[illegible]

The following I-Descriptor is typically used to insert lines on a report for data entry.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ENTER
STR("___",5)	

TRIM Function

The TRIM function removes spaces between elements in a string so that only one logical space separates each string element. For example, TRIM(DESC) removes unnecessary spaces in a product's description.

This enables you to isolate second and third words from a string of characters in which there may have been a more than one space in between the words.

The following example retrieves the second word of the description from the PRODUCT file while reporting from the PSUB file. Using the TRIM function guarantees that the second word is returned.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	MODEL
FIELD(TRIM(TRANS('PRODUCT',FIELD(@ID,'~',1),1,'X'))," ",2)	

Concatenation

Concatenation occurs when you combine two elements of a string together. A colon “:” or “CAT” represents concatenation. As discussed during lesson 7, concatenation can also be applied to I-Descriptors as well as mass loading data. Sometimes creating a dictionary item that concatenates different values can create a “key” to a new file.

Example 1

The following I-Descriptor concatenates the values of two dictionary items and inserts a period between them.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	CATEGORY
GL.SOURCE:'.':GL.TYPE	

Example 2

In the next example, the formula concatenates the Entity ID from the AR file first with a space and then with the actual Customer’s name.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	BILL.#.NAME
@RECORD<10>:" ":TRANS('ENTITY',CUST.NO,1,'X')	

Example 3

The dictionary item in this example is commonly used to gain access to the AR file. The AR file's @ID or key to each record includes the order number and the generation. The PSUB file stores this information in the @ID; however, each element in the PSUB @ID is separated by a tilde (~) and the leading zeros have been stripped from the generation number.

The dictionary item concatenates the order number, a period, and the generation number to create a key to the AR file. This dictionary item also adds the leading zeros to the generation number.

The R%3 forces a number of 1 to read 001. The "R" in R%3 indicates right justification, the "3" indicates the number of positions, and the "%" fills in with zeros.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	FULL.OID
FIELD(@ID, '~', 4):'.' : FIELD(@ID, '~', 5)"R%3"	

Example 4

The following dictionary item retrieves the first line of the address, makes sure that the second field starts in the same position each time, concatenates an "F" and a space, and then concatenates the fax number.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	ADDR1&FAX
@RECORD<2,1>"L#30": "F ":@RECORD<17,2>	

The text within the quotation marks is a literal. Whatever is between the quotation marks is what displays. In this example, the literal is an "F" and a space. The "L#30" forces the address line to be left justified and the space allocated for it to be 30 characters long, even if the address contains less than 30 characters. This guarantees that the Fax number will always start at character position 31.

OCONV Function

The OCONV function converts an attribute according to a specified conversion code and returns the converted output. For example:

```
OCONV(@RECORD<16>,"MCU")
```

The value of @RECORD<16> is converted to uppercase (MCU is the UniVerse code for "Mask Character Uppercase"). Refer to Lesson 4 for different conversion codes. You can also refer to Jonathan E. Sisk's **Basic Programming Guide**, which can be obtained from <http://www.jes.com/pb/>.

The Eclipse search programs differentiate between upper and lower case characters. Use OCONV to change all characters to one case prior to using the search engine.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	PROPER.NAME
OCONV(@RECORD<1>,"MCT")	

DCOUNT Function

I bet that we can agree not all of the product descriptions have the same number of words for you to be able to obtain the last word for every product.

The following dictionary counts the number of words in the product description. It recognizes a space as the delimiter between words. Once it retrieves the count, we can then determine where the last word exists in the description.

The FIELD statement in the next expression uses the count from the first expression to extract the last word of the description. Even though each product description can be different, this dictionary always retrieves the last word.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	DESC.LAST.WORD
<code>DCOUNT(DESCR<1,1>,' '); FIELD(DESCR<1,1>,' ',@1)</code>	

This model can also be used if you wanted to extract the last value in a multi valued string of data. By converting the value markers to spaces you can then count the spaces to come up with the last value assigned to a data element.

Exercise 14.4

1. Create a dictionary item in the PRODUCT file that displays the following information in a blocked format:
 - Keywords
 - Description
 - Weight
 - Price Line
 - Buy Line

Hint: Use the NA dictionary item from the ENTITY file as a template.

Exercise 14.5

1. Create a dictionary item in the ENTITY file for the last word in the Name.

Lesson 15

Using Subroutines

Objectives

After you complete this lesson, you will be able to:

- Learn about the capabilities of Subroutines in Eclipse
- Create a dictionary using a subroutine

Subroutines

Subroutines have been created by Eclipse programmers to be used in dictionaries. There are times that the data needed can only be extracted using a subroutine because the system needs to do more work than a simple trans or field command.

You may have encountered dictionary items using a subroutine. The arguments needed to be passed into the subroutine may be different for each subroutine. This lesson will discuss what is needed to use a subroutine on a dictionary item and how to recognize the requirements.

To view a subroutine you will need to access the Program editor program located from your system programming menu.

File Name		Edit Program		Run Program	
BP		DICT.GET.LEDGER.VALUE		DICT.GET.LEDGER.VALUE	
E,ED,C,CF,R,S,DS,T,C*,P.V...: V					
Version#	Last Updated By	Open to	12 Open Pgms		
Log	Open List	Close	CoPy	CopY	Open
			Run	Undo Chg	Docs
PAtch Create	Bld Pgm Idx	Find ObJ	SHow Calls	PhaNtom Run	
Send/Receive Data		Ecomm	NoTify on Close		

The system will want to know the file name for which the program is stored. There are:

SP	System Programs
BP	Basic Programs (90 % of our code)
FBP	Field (or Fixed) Basic Programs
UBP	User Specific (Custom) Basic Programs
CBP	Client Modified Basic Programs

The filename for which these subroutines are stored is in the BP – Basic Program file.

The Edit Program prompt is where you want to key in the subroutine that is of interest to you. In the above figure we will be viewing the subroutine called DICT.GET.LEDGER.VALUE.

Once you hit <enter> after typing in your subroutine the system will automatically position your cursor in the “Option to Perform” box. The system will default to the letter “E” which represents Edit mode. We do not want to view our program in edit mode.

Option to Perform

In this field, select the option you want to run the edit for the selected program. The options are all listed and described below.

- **C** - Compile this program (edit program) using the Eclipse pre-compiler.

- **C*** - Compile this program without using the Eclipse pre-compiler. This option will put your code straight into the BASIC compiler and should not be used very often. You will never use this option in this manual, however we will discuss areas of our code that differ from true PICK code.
- **CD** – Compile this program with debug window inputs.
- **CF** – Compile a list of programs (your active list). This option can be done after you do a “Find” from the program editor to compile all of the programs you found in that list.
- **CT** – Compile test(s) on the edit program. This option will perform a test compilation before activating the new code. It is very important when you compile high profile routines in develop or routines on a site. When a compile blows up, that code cannot be accessed until you complete a successful compile on that routine.
- **D** – Delete the edit program from the file you specified in File Name. This option will save a copy of the routine in the DP file.
- **DS** – Send a grid layout of the screen to the printer.
- **DB** – Database allows you to enter a UniVerse BASIC command and get help from the on-line UniVerse help documents.
- **DU** – **DU** let’s you put in a Unix command and get help from the on-line AIX/UNIX manual on that command.
- **E** – Edit the “edit program” with the Eclipse program editor.
- **ED** – Edit the “edit program” with the UniVerse line editor (not recommended for use at Eclipse).
- **EF** – Use the Eclipse program editor to edit all of the programs in your active find list.
- **P** – Print the edit program to the active printer.
- **R** – Run the edit program from this editor.
- **S** – Invoke the Eclipse screen editor to create or edit a screen with the same name as the edit program.
- **T** – Drop down to Eclipse TCL (True Command Language).
- **V** – View the edit program in view only mode (no changes can be made).
- **EV** – Edit the “edit program” using the new GUI program editor.

Again the option we want to choose is V to view the program.

This class does not teach how to create a subroutine. Let’s dissect the following subroutine together.

DICT.GET.LEDGER.VALUE

```

SUBROUTINE (VAL,ATTB)

GN = DCOUNT(@ID,')
IF NUM(FIELD(@ID,',GN)) OR GN <= 1 THEN NULL ELSE VAL="";RETURN

BEGIN CASE
CASE ATTB[1,3] = "ID."
  POS = FIELD(ATTB,',2)
  VAL = FIELD(@ID,',POS+2)
  RETURN
CASE INDEX(@ID,'~',1)
  OID = FIELD(@ID,'~',4)
  IDX = FIELD(@ID,'~',5)
  READV GEN.IDX FROM LEDFILE,OID,8 ELSE GEN.IDX = "
CASE GN=4
  OID = FIELD(@ID,',3)
  IDX = FIELD(@ID,',4)+0
  READV GEN.IDX FROM LEDFILE,OID,12 ELSE GEN.IDX = "
CASE GN = 2
  OID = FIELD(@ID,',1)
  IDX = FIELD(@ID,',2)+0
  READV GEN.IDX FROM LEDFILE,OID,8 ELSE GEN.IDX = "
CASE OTHERWISE
  OID = FIELD(@ID,',1)
  IDX = FIELD(@ID,',3)+0
  READV GEN.IDX FROM LEDFILE,OID,12 ELSE GEN.IDX = "
END CASE

LOCATE IDX IN GEN.IDX<1> SETTING GEN ELSE VAL=""; RETURN

OE.GET.QSIGN QSIGN,OID
BEGIN CASE
CASE NUM(ATTB)
  LOCATE IDX IN GEN.IDX<1> SETTING GEN THEN
    READV VALS FROM LEDFILE,OID,ATTB ELSE VALS = "
    VAL = VALS<1,GEN>
  END ELSE
    VAL = "
  END
  IF ATTB>13 AND ATTB<19 THEN
    VAL = VAL*QSIGN
  END
CASE ATTB[1,3] = 'LI.'

```

```
TYPE  = ATTB[4,99]
VAL   = "
MATREAD LED FROM LEDFILE,OID ELSE MAT LED = "
LDIDS = LED(48)<1,GEN>
LD.CT = DCOUNT(LDIDS,SVM)
FOR LDN = 1 TO LD.CT
  LDID = LDIDS<1,1,LDN>
  LD.GET LDID
  PN = LD(1)
  BEGIN CASE
  CASE NUM(FIELD(TYPE,',',1))
    MATBUILD WRK FROM LD
    VAL<1,LDN> = WRK<TYPE>
  CASE TYPE = 'PN'
    IF NUM(PN) THEN
      VAL<1,LDN> = PN
    END ELSE VAL<1,LDN> = "
  CASE TYPE[1,4] = 'SQTY'
    BEGIN CASE
    CASE NUM(PN)
      SP.QTY = (SUM(LD(5)<1,GEN>) + SUM(LD(6)<1,GEN>))*QSIGN
      IF TYPE = 'SQTY.PER' THEN
        *** convert the UOM qty.
        MATREAD PRD FROM PRDFILE,PN ELSE MAT PRD = "
        IF PRD(15) # " THEN
          UMTBL = PRD(15)
        END ELSE
          READV UMTBL FROM PLNEFILE,PRD(9),3 ELSE UMTBL = "
        END
        IQ.TO.ALPHA UMTBL,PRD(7),LD(23),SP.QTY,,,,,ALPHA
        SP.QTY = TRIM(ALPHA)
      END
      VAL<1,LDN> = SP.QTY
    CASE OTHERWISE
      VAL<1,LDN> = '*'
    END CASE
  CASE TYPE[1,4] = 'OQTY'
    BEGIN CASE
    CASE NUM(PN)
      ORD.QTY = LD(4)*QSIGN
      IF TYPE = 'OQTY.PER' THEN
        *** convert the UOM qty.
        MATREAD PRD FROM PRDFILE,PN ELSE MAT PRD = "
        IF PRD(15) # " THEN
          UMTBL = PRD(15)
        END ELSE
          READV UMTBL FROM PLNEFILE,PRD(9),3 ELSE UMTBL = "
```

```
END
IQ.TO.ALPHA UMTBL,PRD(7),LD(23),ORD.QTY,,,,,ALPHA
ORD.QTY = TRIM(ALPHA)
END
VAL<1,LDN> = ORD.QTY
CASE OTHERWISE
  VAL<1,LDN> = '*'
END CASE
CASE TYPE = 'CT'
  BEGIN CASE
    CASE NUM(PN)
      VAL<1,LDN> = 1
    CASE OTHERWISE
      VAL<1,LDN> = 0
    END CASE
  CASE TYPE[1,4] = 'DESC'
    IF NUM(PN) THEN
      READV DESC FROM PRDFILE,PN,1 ELSE DESC = '* Not Found *'
    END ELSE DESC = "
    IF LD(3)#" THEN DESC<1,-1> = LD(3)
    SVCHK = TYPE[5,99]
    IF TRIM(SVCHK) = " OR NOT(NUM(SVCHK)) THEN
      VAL<1,LDN> = LOWER(DESC)
    END ELSE
      VAL<1,LDN> = DESC<1,SVCHK>
    END
  CASE TYPE[1,4] = 'ENT#'
    IF NUM(PN) THEN
      FOUND = NO
      CMT.CT = DCOUNT(LD(3)<1>,VM)
      FOR JT = 1 TO CMT.CT
        IF OCONV(LD(3)<1,JT>,'MCU')[1,6] = 'YOUR #' THEN
          VAL<1,LDN> = TRIM(FIELD(LD(3)<1,JT>,'#',2))
          FOUND = YES
          EXIT
        END
      NEXT JT
      IF NOT(FOUND) THEN
        OE.CUS.PN.CMT.GET LED(1)<1,GEN>,LED(5)<1,GEN>,,PN,CMT
        VAL<1,LDN> = CMT<1,1,1>
      END
    END
  CASE TYPE = 'PRCLN'
    IF NUM(PN) THEN
      READV PRCLN FROM PRDFILE,PN,9 ELSE PRCLN = '* Not Found *'
    END ELSE PRCLN = "
    VAL<1,LDN> = PRCLN
```

```

CASE TYPE = 'PRCEXT'
  SQTY    = (SUM(LD(5)<1,GEN>)+SUM(LD(6)<1,GEN>))*QSIGN
  VAL<1,LDN> = ICONV(OCONV(LD(8)<1,GEN>,'MR9')*SQTY,'MR2')
CASE TYPE = 'UNIT'
  VAL<1,LDN> = ICONV(OCONV(LD(8)<1,GEN>,'MR9'),'MR3')
CASE TYPE = 'CSTEXT'
  SQTY    = (SUM(LD(5)<1,GEN>)+SUM(LD(6)<1,GEN>))*QSIGN
  VAL<1,LDN> = ICONV(OCONV(LD(10)<1,GEN>,'MR9')*SQTY,'MR2')
CASE TYPE = 'COMEXT'
  SQTY    = (SUM(LD(5)<1,GEN>)+SUM(LD(6)<1,GEN>))*QSIGN
  VAL<1,LDN> = ICONV(OCONV(LD(27)<1,GEN>,'MR9')*SQTY,'MR2')
CASE TYPE = 'TYPE'
  VAL<1,LDN> = FIELD(LD(7)<1,GEN>,'~',1)
CASE TYPE = 'TAG' OR TYPE = 'TAGDT'
  LOCA = FIELD(LD(7)<1,GEN>,'~',2)
  TAG = FIELD(LOCA,'^',2)
  TAG = FIELD(TAG,".",1)
  LOCATE TAG IN VAL<1> BY "AL" SETTING TLOC ELSE
    VAL = INSERT(VAL,1,TLOC;TAG)
  END
CASE TYPE[1,6] = 'ONHAND'
  STK.BR = LED(2)<1,GEN,2>
  IF NUM(PN) THEN
    PRDD.BR.GET.REC STK.BR,PN,REC
    GET.ONHAND REC<1>,REC<8>,STK.OH
    IF TYPE = 'ONHAND.PER' THEN
      *** convert the UOM qty.
      MATREAD PRD FROM PRDFILE,PN ELSE MAT PRD = "
      MATREAD PLNE FROM PLNEFILE,PRD(9) ELSE MAT PLNE = "
      DFLT.PER.GET 'I',PER,ALPHA
      IF PRD(15) # " THEN
        UMTBL = PRD(15)
      END ELSE
        UMTBL = PLNE(3)
      END
      IQ.TO.ALPHA UMTBL,PRD(7),ALPHA,STK.OH,,,,,ALPHA
      STK.OH = TRIM(ALPHA)
    END
  END ELSE
    STK.OH = 0
  END
  VAL<1,LDN> = STK.OH
CASE TYPE = 'PN.STK.FLAG'      ;* stock flag
  *** for stock branch only
  STK.BR = LED(2)<1,GEN,2>
  STK.FLAG = "
  IF NUM(PN) THEN

```

```
GET.PRD.BR.VAL STK.BR,PN,12,STK.FLAG
IF STK.FLAG='1' THEN STK.FLAG='Y' ELSE
  IF STK.FLAG='0' THEN STK.FLAG='N' ELSE STK.FLAG='- '
END
END
VAL<1,LDN> = STK.FLAG
CASE TYPE = 'PN.BUY.ID'      ;* buyer's ID
  *** for stock branch only
  STK.BR = LED(2)<1,GEN,2>
  BUY.ID = "
  IF NUM(PN) THEN
    READV BLINE FROM PRDFILE,PN,12 THEN
      BUYLINE.BR.GET.VAL STK.BR,BLINE,17,BUY.ID
    END
  END
  VAL<1,LDN> = BUY.ID
CASE TYPE[1,4] = 'ONPO'
  IF NUM(PN) THEN
    *** for stock branch only
    STK.BR = LED(2)<1,GEN,2>
    IF TYPE = 'ONPO.DT' THEN      ;* ONPO
      *** for TYPE = 'ONPO.DT'
      ONPO.DT = 99999
    END ELSE
      *** for TYPE = 'ONPO' or 'ONPO.PER'
      MATREAD PRD FROM PRDFILE,PN ELSE MAT PRD = "
      MATREAD PLNE FROM PLNEFILE,PRD(9) ELSE MAT PLNE = "
      ONPO = 0
    END
  PRDD.BR.GET STK.BR,PN
  TRS = PRDD.BR(2)
  TRN = DCOUNT(TRS,VM)
  FOR TRX=1 TO TRN
    IF FIELD(TRS<1,TRX>,'~',3)[1,1]='P' THEN
      IF TYPE = 'ONPO.DT' THEN
        *** get the receiving date
        PO.DT = FIELD(TRS<1,TRX>,'~',2)
        *** set ONPO.DT to the latest date
        IF ONPO.DT > PO.DT THEN ONPO.DT = PO.DT
      END ELSE
        *** get the QTY
        ONPO += PRDD.BR(3)<1,TRX>
      END
    END
  NEXT TRX
  IF TYPE = 'ONPO.DT' THEN
    IF ONPO.DT = 99999 THEN ONPO.DT = "
```

```

      VAL<1,LDN> = ONPO.DT
    END ELSE
      IF TYPE = 'ONPO.PER' THEN
        *** convert the UOM qty.
        DFLT.PER.GET 'P',PER,ALPHA
        IF PRD(15) # " THEN
          UMTBL = PRD(15)
        END ELSE
          UMTBL = PLNE(3)
        END
        IQ.TO.ALPHA UMTBL,PRD(7),ALPHA,ONPO,,,,,ALPHA
        ONPO = TRIM(ALPHA)
      END
      VAL<1,LDN> = ONPO
    END
  END ELSE
    VAL<1,LDN> = "
  END
CASE TYPE[1,7] = 'ON.XFER'      ;* on transter
  *** for stock branch only
  STK.BR = LED(2)<1,GEN,2>
  ONPO = 0
  IF NUM(PN) THEN
    INV.GET.TOTALS
    PN,STK.BR,STK.OH,TAG.OH,STK.CMTD,TAG.CMTD,STK.PO,TAG.PO,STK.XFER,TAG.
    XFER,OTHER,ON.BID,STK.INPR,TAG.INPR
    ONPO += STK.XFER
  END
  IF TYPE = 'ON.XFER.PER' THEN
    *** convert the UOM qty.
    MATREAD PRD FROM PRDFILE,PN ELSE MAT PRD = "
    MATREAD PLNE FROM PLNEFILE,PRD(9) ELSE MAT PLNE = "
    DFLT.PER.GET 'T',PER,ALPHA
    IF PRD(15) # " THEN
      UMTBL = PRD(15)
    END ELSE
      UMTBL = PLNE(3)
    END
    IQ.TO.ALPHA UMTBL,PRD(7),ALPHA,ONPO,,,,,ALPHA
    ONPO = TRIM(ALPHA)
  END
  VAL<1,LDN> = ONPO
END CASE

NEXT LDN

IF TYPE = 'TAGDT' THEN

```

```

TAG.CT = DCOUNT(VAL,VM)
FOR TAG.ID = 1 TO TAG.CT
  TAG = VAL<1,TAG.ID>
  READV SHIP.DT FROM LEDFILE,TAG,9 ELSE SHIP.DT="
  SHIP.DT = SHIP.DT<1,1>
  VAL<1,TAG.ID> = SHIP.DT
NEXT TAG.ID
END
CASE ATTB[1,3]='TV.'
TYPE=ATTB[4,99]
BEGIN CASE
  CASE TYPE = 'PN'
    IF NUM(PN) THEN
      VAL<1,LDN> = PN
    END ELSE VAL<1,LDN> = "
  CASE TYPE = 'NAME'
    READV CN FROM LEDFILE,OID,5 ELSE CN="
    CN = CN<1,GEN>
    READV VAL FROM CUSFILE,CN,1 ELSE VAL="
  CASE TYPE = 'BFLW.DT'
    VAL="
    READ REC FROM LEDLFILE,OID ELSE REC="
    DTS=DCOUNT(REC<11>,VM)
    FOR DTCT = 1 TO DTS
      IF REC<11,DTCT> GT 0 THEN
        VAL<1,-1>=REC<11,DTCT>
      END
    NEXT DTCT
  END CASE
END CASE
RETURN

```

DICT.GET.LEDGER.VALUE

SUBROUTINE (VAL,ATTB)

This dictionary works from the PSUB, LEDGER, AR or the ORDER.QUEUE file. Specify the attribute number from the LEDGER file that you want to pass back. The attribute can also be a code to pull back different information.

- LI.PN will pull back the Line Item Part number.
- LI.SQTY will pull back the line item ship quantity
- LI.SQTY.PER will pull back the uom for the ship quantity.
- LI.OQTY pulls back the line item open quantity
- LI.OQTY.PER pulls back the uom for the open quantity.
- LI.DESC pulls back the description of the line item.
- LI.ENT# pulls back the customer part number (if customers are assigned part numbers in the customer/vendor part number screen)
- LI.PRCLN pulls back the price line for the item on the order.

- LI.PRCEXT pulls back the extended price for the item on the order
- LI.UNIT pulls back the unit price for the item on the order
- LI.CSTEXT pulls back the extended cost of the item
- LI.COMEXT pulls back the unit cost of the item
- LI.TYPE pulls back the sales quantity type such as tag, defective, etc.
- LI.ONHAND pulls back the product's on-hand quantity.
- LI.ONHAND.PER will display the unit of measure for the on-hand quantity.
- LI.PN.STK.FLAG displays the branches stk flag from the Primary inventory maintenance screen. (- = ""; 1 = Y ; 0 = N)
- LI.PN.BUY.ID displays the user id of the buyer for the items on the transaction
- LI.ONPO displays quantity on a purchase order for the line item..
- LI.ONPO.DT displays the expected receiving date of the purchase order
- LI.ONPO.PER displays the unit of measure for the quantity on order from the po.
- LI.ON.XFER displays the quantity on a transfer.
- LI.ON.XFER.PER displays the unit of measure for the quantity on transfer
- LI.TAGDT displays the date the tagged purchase order or transfer is expected to be received in.
- TV.NAME pulls back the full name of the writer of the transaction
- TV.BFLW.DT pulls back the bid follow-up date for the transaction.

Exercise:

Create a dictionary item using the DICT.GET.LEDGER.VALUE subroutine with the knowledge you just learned.

Subroutine Dictionary Examples

The following are some sampling of dictionary items that you will not have in your eclipse application. Let's review the dictionaries and the subroutines being used together.

I-Descriptor Program Maintenance				
File Name : PRODUCT Dict ID : OPEN.ORD.QTY				
=SUBR('DICT.PRD.OPEN.ORD','LD4')				
Test	Find Subr	Edit Subr	Set Common	

The above dictionary will display the order quantity on each open order that displays for a product. This dictionary will simply list the open order quantity amounts for each order. This dictionary will not do math with other columns on a report writer report because it is a vertical listing of quantities not a sum of all of the quantities. Please see next dictionary item.

I-Descriptor Program Maintenance				
File Name : PRODUCT Dict ID : SUM.OPEN.ORD.QTY				
SUM(-SUBR('DICT.PRD.OPEN.ORD','LD4'))				
Test	Find Subr	Edit Subr	Set Common	

The above dictionary item adds all of the open order quantities and provides a summary total. This dictionary can then be used to do math on other columns on a report writer.

I-Descriptor Program Maintenance				
File Name : ENTITY.PN.IDS Dict ID : LH.OPEN.PO				
SUBR("DICT.PRD.OPEN.PO",1,PN)				
Test	Find Subr	Edit Subr	Set Common	

The above dictionary item will display open purchase orders for the PN associated with the Customer Specific/Vendor Specific part number screen.

I-Descriptor Program Maintenance				
File Name : ENTITY.PN.IDS Dict ID : SALES				
SUBR("DICT.PRD.CUS.SALES",1,3,' ',' ',PN,CUST.ID)				
Test	Find Subr	Edit Subr	Set Common	

The above dictionary is used on the ENTITY.PN.IDS file which is where your customer/vendor part numbers are stored. This subroutine will allow you to pass in the eclipse product id and the customer id so that you can return a sales\$ amount for the customer and product from this file.

DICT.BR.VAL.GET

SUBROUTINE (VAL,BR,PN,ATTB.NO,SVM.NO,LOC.FLAG)

Subroutine: DICT.BR.VAL.GET

 This routine pulls branch specific data from the PROD.BR file or the PROD.CALC.BR file calling either PRD.BR.GET.VAL or PRDC.BR.GET.VAL to get a specific attribute for a branch. If no branch is specified then all branches are used.

BR	- Branch	[IN]
PN	- Part Number	[IN]
ATTB.NO	- Attribute Position	[IN]
SVM.NO	- Sub-Value Mark	[IN]
LOC.FLAG	- Flag to set whether to pull from PROD.CALC or PROD.BR	[IN]
VAL	- Value Returned	[OUT]

DICT.CALC.CN.SLS

This subroutine requires 6 arguments to be entered for it to work.

SUBROUTINE (AMT,RANGE,AVN,BRCHS,SD,ED,CN)

** Version# 2 - 05/24/1996 - 02:55pm - STEELI - develop

*-----Date Ranges

- * 1 = User Defined Date Range
- * 2 = MTD Date Range
- * 3 = YTD Date Range
- * 4 = Fiscal MTD Date Range
- * 5 = Fiscal YTD Date Range

*-----AVN Definition

- * 1 = Sales
- * 2 = Gross Profit

DICT.AR.LEDL.INFO

This subroutine is used from the AR file. The two requirements when using this subroutine in an I-descriptor is to pass in the @ID and the word the system needs to find in the change log for the transaction.

The system will return the user, date, time, port and change log as shown below.

S0704030.001

DAVIDB 03/21/00 02:51pm /pts/4 DAVIDB Authorized : Overcommitted

An example of an I-Descriptor using this subroutine is:

File Name : AR

Dict ID : LEDL.OVER

SUBR('DICT.AR.LEDL.INFO',@ID,'Overcommit')

DICT.PRD.AVG.PRC

SUBROUTINE (AVG.PRC)

*** SUBROUTINE - DICT.PRD.AVR.PRC

*-----

*** This routine calculates the Average Selling Price for the Product whose PN is @ID based on the Branches, Start Date and End Date specified through SET.COMMON at TCL.

*-----

*** AVG.PRC (OUT) - The Average Selling Prices (VM by Branch)

DICT.PR.D.CUS.ORDERS

```

SUBROUTINE (ORDATA,VAL,PN,CN)
*****
* VAL = 1 ; Open Order ID's          *
* VAL = 2 ; Required Date            *
* VAL = 3 ; Qty pending shipment     *
* VAL = 4 ; First line of product desc *
* VAL = 5 ; Ship Date                *
*****

```

If you can pass in the @Id of the product file and the @id of the customer, you can return any of the above value numbers.

DICT.PR.D.GET.INFO

SUBROUTINE (VAL,OPT,TYPE)

Works from the PRODUCT file.

The options you can choose are:

LP = Line Point

OP = Order Point

XFER.PT = Transfer Point

DMD = Demand

The different types can be:

WBRS = Warehouse branch

PBRS = Purchasing Branch

DICT.PSUB.LEDL.INFO**SUBROUTINE (VAL,PSUB.ID,WORD)**

This routine will go out to the Ledger log and pull back occurrences you specify. You must enter the word to search on.

Appendix A
Answers to I-Descriptor Exercises



Lesson 12

Exercise 12.1

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	ND.ADDR
FIELD(@RECORD<2>," ",2)	

Exercise 12.2

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ND.KEYWORD
FIELD(@RECORD<4,2>," ",3)	

Lesson 13

Exercise 13.1

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	JH.CUS.SLSM
TRANS('ENTITY',EN,41,'X') TRANS('ENTITY',@RECORD<10>,41,'X') TRANS('ENTITY',STEN,41,'X') TRANS('ENTITY',@RECORD<15>,41,'X')	
Test	Find Subr Edit Subr Set Common

The above answer could be any of the expressions shown above.

Exercise 13.2

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	ND.WRITER
TRANS("LEDGER",ORDER#,73,'X')	

Exercise 13.3

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	ND.WEIGHT
TRANS('PRODUCT',FIELD(@ID,'~',1),10,'X')	

Exercise 13.4

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	ND.SHIPTO.FAX
TRANS('ENTITY',@RECORD<8>,17,'X')<1,1,2>	

Exercise 13.5

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ND.BUYER.NAME
TRANS('BUY.LINE',@RECORD<12>,17,'X'); @1(TRANS('INITIALS',@1,3,'X'))	

OR it could have been:

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	JH.BUYER.NAME
TRANS('INITIALS',TRANS('BUY.LINE',BUY.LINE,17,'X'),'3','X')	

Lesson 14

Exercise 14.1

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	ND.TOTAL.CURRENT
BAL.CURRENT+BAL.FUTURE+BAL.DEPOSITS	

Exercise 14.2

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ND.ST.NSK.LINE
IF STATUS = "1" THEN LINE ELSE STATUS = "2" THEN BUY.LINE ELSE ""	

Exercise 14.3

Report Writer/Mass Load Design						
Design ID : ND.INV.VAL		Created : 01/03/02 By : Nicole Dursi X120				
File Name : PRODUCT		Total Width : 119				
Title : Inventory Value and Difference Between costs						
Col	Dict Item/Formula	Width	Column Heading	Brk	Tot	Format
1	VIEWER.ID	7	ID	N	*	
2	DESCR	35	Description	N	*	
3*	PREV.ONHAND	10	Onhand	N	Y	MR0
4*	BASIS	12	Rep-Cost	N	Y	MR3
5*	BASIS	12	Avg-Cost	N	Y	MR3
6	3*4	12	Ext Rep	N	Y	MR3
7	3*5	12	Ext Avg	N	Y	MR3
8	4-5	12	Difference	N	C	MR3
						9 of 9
Select Build		Adv Select	Eclipse Dict	Column Data	CopY	Del Hdg
Run Report		Dict Summ	Labels	Opts	Notes	View Begin Load Set Val

Exercise 14.4

I-Descriptor Program Maintenance	
File Name : PRODUCT	
Dict ID : JH.HEADER.DATA	
'Desc: ':DESC1:@VM:'Price Line: ':LINE:' Buy Line: ':BUY.LINE:@VM:'Keys: ':KEY1:@VM:'Weight: ':UNIT.WEIGHT	

This answer could be anything you want. Your carriage returns could be anywhere.

Exercise 14.5

I-Descriptor Program Maintenance	
File Name : ENTITY	
Dict ID : SCOTT	
DCOUNT(TRIM(NAME)," ");FIELD(NAME," ",@1)	

Appendix B
File Layouts for Release 7



Entity File Layout

Field #	Field Name:	Just/Width	Field Description
1	NAME	L#35	Entity Name
2	ADDRESS	L#35	Entity Address (Multi Valued)
3	CITY	L#20	City
4	STATE	L#5	State
5	ZIP	R#11	Zip Code
6	COUNTRY	L#10	Country
7	ENTITY.TYPE	L#1	Entity Type MV by position
8	SORTBY	L#11	Sorts entity file
9	INDEX	L#35	Index
10	BILL.TO	R#6	For "Job/Ship-to" entity (Customer)
11	PAY.TO	L#6	Pay To (for ship-froms) (Vendor)
12	SHIP.FROMS	R#6	List of Ship-Froms for "Pay-To" entity (Vendor)
13	CLASS	L#4	Customer Class
14	SHIP.TOS	R#5	List of Ship-Tos For "Bill-To" entity (Customer)
15	EDI.IDS	L#25	EDI Information
16	CONTACTS	L#30	Contacts for this Entity
17	PHONES	L#17	6 Phone numbers - align w/contacts
18	OE.MESSAGE	L#10	Order Entry Message
19	OE.TRIGGERS	L#10	User Defined Order Entry Triggers
21	VALID.WIP.IDS	L#16	Work In Process (Processes)
22	CR.LIMIT	R#14	Customer credit limit
22	PAST.DUE.DAYS	L#14	PAST.DUE.DAYS
23	CREDIT	R#9	Credit Specifications
24	AUTH.BRCHS	L#3	Branches Cus/Vendor has access to
26	OVERRIDE.TERMS	R#1	Ovrd Inv Terms Code for Svc Chg
27	AUTH.REQD	R#1	Authorized personnel only (1=Y)
28	CUST.TERMS	L#12	Customer Payment Terms
29	AUTH.NAMES	L#25	Auth names to place sales orders
30	AMT.ALLOWED	R#12	Max order amt for auth personnel
31	COD.PAY	L#1	Accept Checks? (Company/Personal)
32	TAX.EXP.GRPS	L#10	Tax Exception Groups
33	TAX.STATE.CODE	L#2	Stores Customer State Tax Code
34	TAX.EXP#	L#20	Tax Exempt Number (Line up with state)
35	TYPE	L#12	Customer Pricing Type / Vendor Type
36	TAX.JUR.OVRD	L#10	Tax Jurisdiction Override
37	APPLY.CR	R#1	Apply credits to oldest buckets (Y=1)
38	USE.DFLT.BT.CR	R#1	1=Use Dflt 2=Use Bill-to (Credit information)
39	FGHT.IN.EXEMPT	L#1	Entity is Exempt from incoming freight charges
39	FGHT.OUT.EXEMPT	L#1	Customer is Exempt from outgoing freight charges
40	SEP.AR.FR.BT	L#12	Separate AR from Bill-To
41	SALESMAN	L#20	Salesperson Name (Outside)

42	PRICE.STYLE	L#10	Pricing Print Style for Sales Orders
43	BO.ACTION.CUS	L#1	Backorder Status (C,H,A,L,X)
44	SALESMAN.IN	L#10	Salesperson Name (Inside)
45	PO#.REQ	L#1	PO# or Job Name req'd? - P,R,B=both
46	SHIP.VIA	L#25	Default Ship Via code; See SHIP-VIA
47	CREDIT.STATS	R#10	# Payment Days, Hi AR, Avg AR - multi-valued
48	START.DATE	R#10	Start Date (Create Date)
49	USER.DEFINED	L#10	User Defined
50	INV.COPY.CNT	R#5	# of Invoice copies
51	SEL.CODE	L#20	Customer Pricing Selection Code
52	SHIP.INST.DFLT	L#38	Default Shipping Instructions
53	HOME.BR	R#5	Home Branch
54	HOME.TERR	L#10	Home Territory
55	EXP.DIST.CODE	L#10	Expense Distribution Codes
56	EXP.ACCT#	L#10	Expense Account#
57	EXP.PERC	R#5	Expense Distribution Percentages
58	VEN.TERMS	L#20	A/P Terms (Vendor)
59	FRT.ALLOWED	L#2	Vendor Frt Allowed (Y/N)
60	OVR.SHORT%	R#12	Maximum% for A/P shipment
61	OVR.SHORT\$	R#12	Maximum\$ amt for A/P shipment
62	BO.ACTION.VEN	L#10	Vendor Backorder
63	STMT.TYPE	L#1	alance Fwd/<O>pen Item Statement
64	STMT	L#1	Print Batch Statement Y/N
65	DELETE.FLAG	L#1	Delete Flag
66	ALT.CUST	L#8	Alternate Customer/Vendor (entity-specific) Part #
67	REBATE.DATA	L#10	Rebate Product IDs and Price Line
68	REMOTE.DATA	L#10	Remote OE Data
70	PRC.LINES	L#10	Price Lines
71	INIT.STAT.OVRD	L#1	Initial Status Override
72	SLS.SRC.OVRD	L#10	Sales Source Override
73	SHIP.BR.OVRD	R#3	Shipping Branch Override
74	PRT.KIT.COMPS	L#1	Print Kit Components? (Y/N)
75	REQ.LT	R#3	Req'd Date Leadtime
76	MAX.EARLY.SHIP	R#3	Maximum Early Ship Days
77	SIC.CODE	L#10	SIC Code
78	CUS.SEL.CODE	L#35	User Defined Customer Select Code
79	AUTO.DOC.ID	L#30	Auto Document Printing ID
80	PARENT.ID	R#9	Parent Statement Customer ID
81	RANK	L#1	Customer Ranks
82	PNT.DATA	L#20	Points Data
83	SINGLE.INV.FLAG	R#1	Single Invoice Flag (Y=1)
84	MAX.COLLECT.DAYS	R#5	Maximum Collection Days
85	CONTRACT.PRC.IDS	L#10	Contract Pricing IDs
86	UET.DATA	L#10	UET Data
87	NORMAL.DEL.TIME	R#7	Normal Time
88	D&B#	L#20	D&B#
89	CERTIFICATION.DATE	L#30	Certification Information
90	CURR.TYPE	L#9	Primary Currency Type (Blank=System default)

91	ALT.BT.NAME	L#15	Alternate Bill-To Name
92	ALT.BT.ADDR	L#20	Alternate Bill-to Address
93	ALT.BT.CITY	L#11	Alternate Bill-To City
94	ALT.BT.ST	L#3	Alternate Bill-To State
95	ALT.BT.ZIP	L#10	Alternate Bill-To Zip code
96	ALT.BT.SORT	L#12	Alternate Bill-To Sortby
97	ALT.BT.USE.ST	R#2	Alternate Bill-To - Use Ship-To Flag
99	WOE.FORMS	L#25	WOE Custom Forms
100	UBAP.OVRD	L#9	UBAP Ovrdr GL # (Vendor Add'l screen)
101	CREDIT.CARD	L#30	Credit Card Default Info (Customer)
102	1099.ID	L#15	1099 Tax ID (Vendor)
102	VALID.POS	L#15	Valid PO List (Customer)
103	FRGHT.DAYS	R#3	Freight Days (Vendor)
104	SUBS	L#3	Restrict Substitutes? (Customer)
105	TAX.EXEMPT.CODES	L#16	Tax Exempt Codes (Line up W/ exempt #'s)
106	LINE.ITEM.TAX	L#1	Line Item Tax (""=Disabled; 1=Always prompt; 2=Not)
107	ACL.USERS	L#8	Access Control List - Users
108	ACL.VONLY	L#1	Access Control List - View Only
109	QUAL.LVL	L#2	Acceptable Quality Controls

Product Dynam File Layout

Field #	Field Name	Just/Width	Field Description
1	ONHAND	R#10	On Hand dddd~i~nnnnnnnn : dddd=date; i=<D; (sales & purch transactions)
2	ID	L#40	
3	STOCK.QTY	R#10	STOCK.QTY (related to field 2)
4	REVIEW.QTY	R#10	REVIEW.QTY (related to field 2)
5	BID.ID	L#25	dddd~i~nnnnnnnn : dddd=date; i=<Doc initial>; nnnnnnnn=LDI
6	BID.QTY	R#10	BID.QTY
7	UPD.GEN	R#	Generation# of this record. Must be incremented every time r
8	LOCATION	L#10	Bin & Aisle Location for stock qty in Attb#1 (VM) by Brch
9	CTRL.TYPE	L#1	Loc Ctrl Type
10	LANDED.COST	R#12	Landed Last Cost
11	AVG.COST	R#10	Average Cost
12	LAST.COST	R#10	Last Cost received
13	LOC.STAT	L#1	Location Status
14	AVAIL.CACHE.ID	L#10	AVAIL CACHE ID
15	AVAIL.CACHE.QTY	R#6	AVAIL CACHE QTY
16	LOC.MIN	R#6	Location Min Qty
17	LOC.MAX	R#6	Location Max Qty
18	LOC.LAST.CNT	R#8	Last Count
19	LOC.SORTBY	L#8	Location Sortby
20	AVG.LANDED.COST	L#15	AVG.LANDED.COST
20	LOC.PICK.PRI	R#2	Pick Priority
21	SER.NUMS	L#20	SER.NUMS
22	SER.QTYS	R#8	Open Serial Qtys
23	SER.LOCS	L#20	Serial Number Locations
24	FRZ.AVG.COST	R#12	FRZ.AVG.COST
25	FRZ.LAST.COST	R#12	FRZ.LAST.COST
26	FRZ.DATE	R#10	FRZ.DATE
27	LOC.PICK.STATUS	L#15	LOC.PICK.STATUS
28	LOC.PICK.LOCN	L#12	LOC.PICK.LOCN
29	PREPACK.QTYS	R#12	Pre-Package Qty (By Br By Locations)

Product File Layout

Field #	Field Name	Just/Width	Field Description
1	DESC	L#35	Description
2	GL.TYPE	L#8	General Ledger Product Type
3	STATUS	R#6	Product Status ; 1=stock, 2=non-stock
4	KEYWORDS	L#20	Keywords
5	PRICE.AS	L#10	Product ID of Master Price Item
6	SHEET	L#4	Branch Price Sheet ID
7	UOM.QTY	L#7	Unit of Measure Quantity
8	INDEX.TYPE	L#1	Index Type: P = Primary; C = Catalog
9	LINE	L#8	Price Line
10	UNIT.WEIGHT	R#10	Unit Weight (lbs or kgs) (MR4)
11	FORECAST.PARAMS	L#15	Product Level Forecast parameters (Min Hits...)
12	BUY.LINE	L#8	BUY.LINE
13	LOAD.FACTOR	R#10	Load Factor (MR4)
14	COMM.CODE	L#20	Commodity Code
15	PN.UMS	L#15	Product Level Unit of Measures.
16	DFLT.UMS	L#8	Default Units of Measure
17	BLANK2	L#6	Not Yet Used
18	PROCURE.GRP	L#10	Procurement Group
19	SELECT.CODE	L#25	Product Select Code
20	SORT.CODE	R#9	Sort Code
21	RANK	L#12	Product Rank
22	TAX.EX	L#10	Name of Product Exception Groups -- Hot Key X
23	TAX.EX.CODE	L#2	Tax Exception Code: 1=reduced; 2=always full; 3=always reduced
25	MIN.SELL.QTY	R#3	Sell Package Quantity - selling multiple
26	CREATE.DATA	L#15	Product Creation Info such as Date; User etc
27	MIN	R#6	Minimum (Non Forecasted)
28	MAX	R#6	Maximum (Non Forecasted)
29	LAST.DEMAND.CALC	R#10	Last Date Demand was Calculated; By Branch
30	SALES.HITS	R#5	# of Sales Transactions Per Year
31	DEMAND.DAY	R#10	Demand in Units Per Day
32	LEADTIME	R#3	Lead Time in Days
33	LOW.SALE	R#5	Low Sale Quantity
34	E.O.Q.	L#6	E.O.Q.
35	BUYPKG	R#6	Buy Package Quantity - used for rounding up in Purchasing
36	TREND	R#3	Trend % for Forecasted Demand.
37	SERVICE.STOCK	R#5	Service Stock
38	LOST.SLS.CTRL	R#4	Lost Sales Control%
39	BO.TOL.QTY	R#6	Forecasting Will Exclude Sales >= to this Value
39	EXCP.SLS.%	R#5	Exception Sales Control %
40	EOQ%	R#2	Carrying Cost Percent to Use in EOQ Formula

41	EOQ\$	R#4	Reorder Cost Per Line Item to Use in EOQ Form
42	BLANK3	L#6	Not Yet Used
43	STOCK	L#1	The STK Field in the Inventory Maintenance Sub by Branch
44	DEMAND.PERIOD	R#3	# of Days Used to Forecast Demand
45	RAW.HITS	R#6	Raw Hits in Forecast Period
46	RAW.DEMAND	R#6	Total Quantity Sold Over Demand Period
47	DAYS.OUT	R#8	Days Out of Stock
48	BR.LS.MULT	R#5	Lost Sales Multiplier Actually Used; 1.00=No
49	HI.TRANS	R#6	Quantity of Largest Transaction
50	EXCP.QTY.EXCLUDED	R#15	Largest Sale Within Forecast Period That is excluded
50	TREND.FACTOR	R#12	Trend Factor Used
51	SAFETY.FACTOR	R#3	Manual Safety Factor
52	COMP.QTYS	R#5	Multi-Value List of Components Quantities
53	COMP.PNS	L#10	Multi-Value List of Component ID Numbers
54	COMPONENT.OF.KIT	R#6	Internal ID#s of Kits this product is part of
55	SUBSTITUTES	R#6	List of Substitutes
56	SUBS.BRS	R#3	Branch Listings Where Subs in Attr#55 Are
57	SUBS.NOTES	L#30	Notes Pertaining to Multi-Valued Substitutes
58	DONT.FORGET.MSG	L#35	Remind Hot Key in Product File
59	SEASONAL	R#1	Seasonal Flag Y or N
60	ADD.DMD.BASE	L#1	Add demand of other product basis U=Units
61	BR.MM.EXP	R#10	Min/Max Expiration Date
62	LEAD.FACTOR	R#3	The # of Receipts Looked at Going Back from
63	UPC#	L#20	UPC# & user defined upc#s - 5 multi-values
64	ADD.DMD	R#5	IDs of Other Products Whose Demand is Being
65	BR.LABEL.PER	R#5	Label per Quantity
66	ALT.DESC	L#35	Alternate Description for product
67	MERGE.PN	L#10	Keeper Product this item was merged
68	PASS.DISC	R#5	Pass-along Discount.
69	FREIGHT	R#12	Freight Charge by Branch
70	NO.PRINT	L#1	Price Book No Print Flag. Located Alt-P to Alt-R
71	DISC.CLASS	L#3	Price Sheet Discount Class; For Use in Price Sheet
72	BUY.GRP	L#12	Maint
73	SELL.GRP	L#12	Sell Group for Selling Matrix
74	SER#	R#1	Buy Group for Buy Matrix
75	COMM.GRPS	L#20	Serial # 'I'nbound, 'O'utbound, 'A'uto, 'N'one
76	QTY.BRK	R#5	Commission Groups
77	QB.UM	L#2	Price Break Quantities Fields -- Multi-Valued
78	PKG.DIV	L#1	Quantity Break Unit of Measure
79	CALC.DMD	R#1	Package Divisible: Null or Y.
80	HAZARD.DATA	L#30	If Y, Use Calc Demand for Purchasing
81	BR.HITS	R#4	Hazard Data
82	NET.HITS	R#4	Branch Hits
83	CW.TYPE	L#1	Network Hits
84	FREIGHT.FACTOR	R#10	Central Whse Type T=Top Down; B=Bottom Up
85	COMP.CMTS	L#35	Cost Freight Factor
			Kit Components Comments

86	KIT.OPTS	L#1	Kit Options
87	COMP.SPOIL	R#6	Component of Kit Spoil %
88	POINTS.DATA	R#20	Points Data
89	UET.DATA	L#20	Unquality Event Parameter Control
90	STOCK.ITEM	L#1	Stock Item
91	CERT.REQ	L#11	Cust Certification Required to Buy
92	BUDGET.GRP	L#15	Budget Group
93	XREF.DATA	L#40	Cross Reference data for HTML connection.
94	CUST.SERV.STOCK	L#15	Product Inventory for Customer Service Stock
95	SUBS.TYPE	L#5	Substitute product types
96	SUBS.SUG.QTY	L#8	Substitute Suggested Sell Qty
98	TEMP.NEW.LOC	L#12	TEMP.NEW.LOC - you may trash this after 2-1-98
99	MIN.LEAD.FACTOR	R#2	Minimum Lead Factor

Product Price File Layout

Field #	Field Name	Just/Width	Field Description
1	LIST-PER	R#4	Quantity of selling basis units
2	PER.UM	L#2	Description of price per unit; EA,C,M
3	PER.QTY	R#8	Price Per Qty related to 2; 1,100,1000
11	BASIS-1	R#10	Usually LIST PRICE
12	BASIS-2	R#10	Selling Basis -- By Effective Date (Value)
13	BASIS-3	R#10	Selling Basis -- By Effective Date (Value)
14	BASIS-4	R#10	Selling Basis -- By Effective Date (Value)
15	BASIS-5	R#10	Selling Basis -- By Effective Date (Value)
16	BASIS-6	R#10	Usually Replacement Cost
17	BASIS-7	R#10	Usually Standard/Market Cost
18	BASIS-8	R#10	Selling Basis -- By Effective Date (Value)
19	BASIS-9	R#10	Selling Basis -- By Effective Date (Value)
20	BASIS-10	R#10	Selling Basis -- By Effective Date (Value)
21	BASIS-11	R#10	Selling Basis -- By Effective Date (Value)
22	BASIS-12	R#10	Selling Basis -- By Effective Date (Value)
30	BASIS-20	R#10	Selling Basis -- By Effective Date (Value)

Searching for a Dictionary Item using TCL

In **TCL** you can use the **SEARCH** command to find a dictionary item. The following example returns a list of dictionary items in the AR file that use the TRANS command. You can also do the same search for the string FIELD.

```
;SEARCH DICT AR
STRING:TRANS
STRING:
```

54 record(s) selected to SELECT list #0.



ACTIVANT™

Appendix C

File Layouts for Release 8

Entity File Layout Release 8

ENTITY file Dictionary Items

Dictionary ID.....	Description.....	Typ	Attr	Val.	Just
@ID	@ID	D	0		L
CUST_ID	Customer ID	D	0		R
VEND_ID	Vendor ID	D	0		R
NAME	Entity Name	D	1		L
ADDRESS	Entity Address	D	2		L
ADDRESS2	Address Second Line	D	2	2	L
CITY	City	D	3		L
STATE	State	D	4		L
ZIP_CODE	Zip Code	D	5		R
COUNTRY	Country	D	6		L
ENTITY_TYPE	Entity Type MV by position	D	7		L
IS_BILL_TO	Is Bill To Customer	D	7	1	R
IS_SHIP_TO	Is Ship To Customer	D	7	2	R
IS_BR_ENTITY	Is Branch Entity	D	7	4	R
IS_PAY_TO	Is Pay To Vendor	D	7	5	R
IS_SHIP_FROM	Is Ship From Vendor	D	7	6	R
IS_BR_ACCOUNT	Is Branch Cash Account Customer	D	7	7	R
IS_FREIGHT_VENDOR	Is Freight Vendor	D	7	8	R
IS_PROSPECT	Is Prospect Customer	D	7	9	R
IS_MFR	Is Manufacturer Vendor	D	7	10	R
SORT_BY	Sorts ENTITY file	D	8		L
INDEX	Index (used for searching)	D	9		L
BILL_TO	Bill To ID (for Ship To) (Customer)	D	10		R
PAY_TO	Pay To ID (for ship from) (Vendor)	D	11		L
SHIP_FROMS	List of Ship-From Vendors for Pay-To entity (Vendor)	D	12		R
CLASS	Customer Class	D	13		L
SHIP_TOS	List of Ship-To Customers For Bill-to entity (Customer)	D	14		R
EDI_ID	EDI Group ID	D	15		L
CONTACTS	Contacts	D	16		L
PHONE_NBRS	Phone Numbers	D	17		L
OE_MESSAGES	Order Entry Message (upto 10 lines of text)	D	18		L
OE_TRIGGERS	Order Entry Triggers (Customer) Fax, email etc.	D	19		L
ORDER_LIMIT	Order Limit	D	20		R
WIP_DATA	Work In Process Data	D	21		
WIP_DAYS	Work In Process Days	D	21		2 R
WIP_IDS	Work In Process Ids	D	21		1 L
WIP_SETUP_COSTS	Work In Process Setup Costs	D	21		3 R
WIP_UMS	Work In Process Unit of Measures	D	21		5 L
WIP_UNIT_COSTS	Work In Process Unit Costs	D	21		4 R
CREDIT_LIMIT	Customer credit limit	D	22	1	1 R
JOB_LIMIT	Job Limit	D	22	1	3 R
PAST_DUE_LIMIT	Customer Past Due Limit	D	22	1	2 R
NS_DEPOSIT	Deposit amount required for NonStock items	D	22	2	3 R
PAST_DUE_DAYS	Customer Past Due Days	D	22	2	1 R
STOCK_DEPOSIT	Deposit amount required for Stock items	D	22	2	2 R
CREDIT	Credit Specifications	D	23		R
IS_COD	Is Cod Customer	D	23	1	R

**ACTIVANT®**

IS_COD_CREDIT	Is Cod Customer when credit limit exceeded	D	23	2	R
REQ_APPROVAL	Print approval required message on all shipping tickets	D	23	3	R
REQ_APPROVAL_CREDIT	Print approval required message on ship tickets override	D	23	4	R
NO_OE	No Order Entry	D	23	5	R
NO_OE_CREDIT	No Order Entry when credit limit exceeded unless a	D	23	6	R
NO_OE_AUTH	No Order Entry regardless of credit limit unless a	D	23	7	R
NO_PRINT_CREDIT	No printing of shipped ticket when credit exceeded	D	23	8	R
NO_PRINT_AUTH	No printing of shipped ticket unless authorized	D	23	9	R
AUTH_BR_DATA	Branches & Territories Customer/Vendor has access	D	24		L
AUTH_BR	List of Branches	D	24	1	L
ACTIVE_BR	Active Branches (1 = Yes, Null = No)	D	24	2	R
BR_ACT_LEVEL	Branch activation level	D	24	3	R
ONE_TIME_ACT_LEVEL	One-Time activation level	D	24	4	R
PRICE_CLASS_OVRD	Price Class Override vm1-Matrix Prices, vm2-Type C	D	25		L
OVERRIDE_TERMS	Overridden Invoice Terms Code for Service Charges 1	D	26		R
AUTH_PERSONNEL_REQD	Authorized personnel only 1 = Y	D	27		R
CUST_TERMS	Customer Payment Terms	D	28		L
AUTH_NAMES	Authorized buyers to place sales orders	D	29		L
AMT_ALLOWED	Max order amount for authorized buyers	D	30		R
COMPANY_CHECK	Accept Company Checks 1 = Y	D	31	1	R
PERSONAL_CHECK	Accept Personal Checks 1 = Y	D	31	2	R
TAX_EXC_GROUPS	Tax Exception Groups	D	32		L
TAX_STATE_CODES	Customer State Tax Exempt Code	D	33		L
TAX_EXP_NBR	Tax Exempt Number	D	34		L
PRICING_TYPE	Customer Pricing Type / Vendor Type	D	35		L
TAX_JUR_OVERRIDE	Tax Jurisdiction Override	D	36		L
APPLY_CREDITS	Apply credits to oldest buckets 1 = Y	D	37		R
USE_DFLT_BT_CR	1=Use Dflt 2=Use Billto (Credit information)	D	38		R
FREIGHT_OUT_EXEMPT	Customer is exempt from outgoing freight charges	D	39	1	R
FREIGHT_IN_EXEMPT	Customer is exempt from incoming freight charges	D	39	2	R
SEPERATE_BILL_TO_CREDIT	Separate AR from Bill-To	D	40		R
OUTSIDE_SALES	Outside Salesperson	D	41		L
PRICE_STYLE	Pricing Print Style for Orders	D	42		L
BACK_ORDER_STATUS	Backorder Status	D	43		L
INSIDE_SALES	Inside Salesperson	D	44		L
PO_NUM_REQ	Purchase Order Number required	D	45	1	L
DEFAULT_PO_NBR	Default Purchase Order Number	D	45	2	L
DEFAULT_RELEASE_NBR	Default Release Number	D	45	3	L
SHIP_VIA	Default Ship Via code	D	46		L
PAYMENT_DAYS	Payment Days	D	47		R
START_DATE	Start Date (Create Date)	D	48		R
USER_DEFINED		D	49		
INVOICE_PRINT_COPIES	Number Invoice Print Copies	D	50	1	R
INVOICE_FAX_COPIES	Number Invoice Fax Copies	D	50	2	R
INVOICE_SELECT_CODE	Invoice Select Code	D	51		L
SHIP_INST_DFLT	Default shipping Instructions	D	52		L
HOME_BR	Home Branch	D	53		L
HOME_TERRITORY	Home Territory	D	54		L
EXPENSE_DIST_CODE	Expense distribution codes (Vendors)	D	55		L
EXPENSE_ACCOUNT_NBR	Expense Branch~Account numbers (Vendors)	D	56		L
EXPENSE_PCT	Expense distribution percentages	D	57		R

**ACTIVANT[®]**

VENDOR_TERMS	Vendor Payment Terms	D	58	L
VENDOR_FREIGHT_TERMS	Vendor Freight Terms	D	59	L
OVER_SHORT_PERCENT	Maximum% for A/P Invoice can be over/short	D	60	R
OVER_SHORT_DOLLAR	Maximum dollar amount A/P Invoice can be over/short	D	61	R
VENDOR_BACK_ORDER	Vendor Backorder status	D	62	L
STATEMENT_TYPE	alance Fwd/<O>pen Item Statement	D	63	L
BATCH_STATEMENT	Batch Statement	D	64	L
DELETE_FLAG	Delete Flag	D	65	L
ALT_CUST	Alternate Customer/Vendor Id	D	66	L
REBATE_DATA	Rebate Product IDs and Price Line #	D	67	L
REBATE_PRODUCT_ID	Product, Price Line, Sell Group	D	67	1 L
REBATE_ENTITY_ID	Rebate Vendor	D	67	2 L
REBATE_EXPIRE_DATE	Rebate Expiration Date	D	67	3 L
REBATE_NUMBER	Rebate Contract Number	D	67	4 L
ROE_DATA	Remote Order Entry Data	D	68	L
ROE_LOG_BID	Remote Log Bids For Review	D	68	1 L
ROE_LOGORDERS	Remote Log Orders For Review	D	68	2 L
ROE_DFLT_MSG_LOGIN	Remote Default User Messaged Upon Log In	D	68	3 L
ROE_DFLT_MSG_NEW_ORD	Remote Default User Messaged With New Order Number	D	68	4 L
ROE_SLAVE_PRT	Remote Slave Printer (Blank=No ROE Printing)	D	68	5 L
ROE_VALID_STATUS	Remote Valid Order Statuses	D	68	6 L
ROE_LOGIN_MSG	Remote Log In Message	D	68	7 L
ROE_DFLT_MSG_ORD_CHG	Remote Default User Messaged With Order Changes	D	68	8 L
ROE_SHOW_PROD_AVAIL	Remote Show Product Availability	D	68	9 L
ROE_DFLT_ORD_STATUS	Remote RDC Default Order Status	D	68	10 L
ROE_B2B_PASSWORD	Remote B2B/Web Order Entry Password	D	68	11 L
ROE_LOGORD_ERRORS	Remote Only Log Order With Errors	D	68	12 L
ROE_PKG_QTY	Remote RDC Round To Sell Package Qtys	D	68	13 L
ROE_MSG_VALID_USERS	Remote Message Valid Users	D	68	14 L
ROE_SHOW_AVAIL	Remote Show Availability For Order Branches	D	68	15 L
ROE_DISCOUNT	Remote Order Entry Discount Percent	D	68	16 R
ROE_AR_INFO	Remote Web Order Entry A/R Information Password	D	68	17 L
ROE_SHIP_VIAS	Remote Valid Ship Vias	D	68	18 L
ROE_SHOW_LIST_PRC	Remote Show List Price In Web Order Entry	D	68	19 L
ROE_ORD_QUEUE	Remote Order Queue Users	D	68	20 L
ROE_CC_ENTERED	Remote Force Credit Card Info To Be Entered	D	68	21 L
ROE_REQ_ORD_BY	Remote Required Ordered By	D	68	22 L
ROE_ALLOW_NEW_ST	Remote Allow Creation Of New Ship To Customers	D	68	23 L
ROE_HIDE_LED_BUTTON	Remote Hide Ledger Button	D	68	24 L
ROE_HIDE_ACCT_INQ	Remote Hide Account Inquiry Button	D	68	25 L
ROE_B2B_POST_URL	Remote B2B Post URL	D	68	26 L
ROE_LOGOFF_URL	Remote WOE Logoff URL	D	68	27 L
ROE_HIDE_ZERO_PRC	Remote Hide Products With Zero Pricing	D	68	28 L
ROE_MAX_SHP_DISPLAY	Remote Maximum Number of ST's to Display Per Page	D	68	29 R
ROE_PDW	Remote PDW flag	D	68	30 L
ROE_CATALOG	Remote Catalog flag	D	68	31 L
ROE_ALLOW_PRC_VAR	Remote Allow Pricing Variance Percentage	D	68	32 L
ROE_PREVENT_BID_MODS	Remote Prevent Mod of Non-WOE Bids	D	68	33 L
PRICE_LINES	Valid Price Lines	D	70	L
INIT_STAT_OVERRIDE	Initial Status Override	D	71	L
SALES_SRC_OVERRIDE	Sales Source Override	D	72	L

**ACTIVANT[®]**

SHIP_BR_OVERRIDE	Shipping Branch Override	D	73	1	L
PRICE_BR_OVERRIDE	Pricing Branch Override	D	73	2	L
PRINT_KIT_COMPS	Print Kit Components? (Y/N/Default)	D	74		L
REQUIRED_LEAD_TIME	Required Lead Time days	D	75		R
MAX_EARLY_SHIP_DAYS	Maximum Early Ship Days	D	76		R
SIC_CODE	SIC Code	D	77		L
CUSTOMER_SELECT_CODE	Customer Select Codes	D	78		L
SOE_USER_DEF_DOC	SOE Document Printing ID	D	79		L
PARENT_STATEMENT_ID	Parent Statement Customer ID	D	80		L
RANK	Customer Ranks (A to G)	D	81		L
POINTS_DATA	Points Data	D	82		L
SINGLE_INVOICE_FLAG	Single Invoice Flag (Y=1)	D	83		R
MAX_COLLECT_DAYS	Maximum Collection Days	D	84		R
CONTRACT_PRICING_IDS	Contract Pricing IDs	D	85		L
UET_EARLY_DAYS	UET Early Days Allowed	D	86	1	R
UET_LATE_DAYS	UET Late Days Allowed	D	86	1	R
NORMAL_DELIVERY_TIME	Normal delivery time of day	D	87		R
DUNN_BRADSTREET_NBR	Dunn and Bradstreet Number	D	88		L
CERTIFICATION_DATA	Certification Information	D	89		L
CERTIFICATION_CODE	Certification Code	D	89	1	L
CERTIFICATION_NAME	Person who holds certification	D	89	2	L
CERTIFICATION_NBR	Certified persons certification number	D	89	3	L
CURRENCY_TYPE	Primary Currency Type (Blank=System)	D	90		L
ALT_BILL_TO_NAME	Alternate Bill-To Name	D	91		L
ALT_BILL_TO_ADDRESS	Alternate Bill-to Address	D	92		L
ALT_BILL_TO_ADDRESS1	Alternate Bill-to Address1	D	92	1	L
ALT_BILL_TO_ADDRESS2	Alternate Bill-to Address2	D	92	2	L
ALT_BILL_TO_ADDRESS3	Alternate Bill-to Address3	D	92	3	L
ALT_BILL_TO_ADDRESS4	Alternate Bill-to Address4	D	92	4	L
ALT_BILL_TO_ADDRESS5	Alternate Bill-to Address5	D	92	5	L
ALT_BILL_TO_CITY	Alternate Bill-To City	D	93		L
ALT_BILL_TO_STATE	Alternate Bill-To State	D	94		L
ALT_BILL_TO_ZIP	Alternate Bill-To Zip-code	D	95		L
ALT_BILL_TO_SORTBY	Alternate Bill-To Sort-by	D	96		L
ALT_BILL_TO_USE_ST	Alternate Bill-To - Use Ship-To Flag	D	97		R
EMAIL_PREFERENCE	Email Preference (Plain or Html)	D	98		L
WOE_FORMS	WOE Custom Forms	D	99		L
VEND_UBAP_GL_ACCT	UBAP Override GL # (Vendor Addl screen)	D	100		L
CC_DATA	Credit Card Data	D	101		L
CC_TYPE	Credit Card Type	D	101	1	L
CC_NUMBER	Credit Card Number	D	101	2	R
CC_EXP_DATE	Credit Card Expiration Date	D	101	3	R
CC_NAME	Credit Card Name	D	101	4	L
CC_ADDL_INFO	Credit Card Additional Information	D	101	5	L
CC_ZIP	Credit Card Zip Code	D	101	6	L
CC_AUTH_TYPE	Credit Card Authorization Type	D	101	7	L
CC_ADDRESS	Credit Card Address	D	101	8	L
CC_INFO_TYPE	Credit Card Info Type (None, Prompt, Required)	D	101	11	L
CC_REF_NBR_TYPE	Credit Card Ref Num Type (Release #, Cust PO, Invoice etc)	D	101	12	L
CC_OVR_TERMS	Credit Card Override Terms	D	101	13	L
CREDIT__CARDCHG_TAX	Credit Card Charge Tax	D	101	14	L

**ACTIVANT®**

1099_TAX_ID	1099 Tax ID (Vendor)	D	102	L
VALID_POS	List of Valid Purchase Order Numbers	D	102	L
FREIGHT_DAYS	Freight Days (Vendor)	D	103	L
RESTRICT_SUBS	Does Customer accept substitutes? (1=Y)	D	104	L
TAX_EXEMPT_CODES	Tax Exempt Codes (Customer)	D	105	L
LINE_ITEM_TAX	Line Item Tax (^=Disabled; 1=Always Prompt;2=Not	D	106	L
ACL_USERS	Access Control List - Users	D	107	L
ACL_LEVEL	Access Control List - User Level	D	108	R
CAGE_CODE	Cage Code (Vendor)	D	109	L
COMMISSION_TYPE	Split Commission Type	D	110	L
COMMISSION_SLSPERSON	Split Commission Salespeople	D	111	L
COMMISSION_PCT	Split Commission Percent	D	112	R
CUST_ALLOW_DUP_ORD	Disable Duplicate Order Check	D	113	L
WEB_ADDRESS	World Wide Web Address	D	114	L
EMAIL_ADDRESS	Email Addresses	D	115	L
EMAIL_TYPE	Email Type	D	116	L
FREIGHT_DATA	Freight Data	D	117	L
FREIGHT_EXEMPT_HANDL	Freight Exempt Handling	D	117	1 L
FREIGHT_INSR_REQR	Freight Insurance Required	D	117	2 L
FREIGHT_XTRA_HANDL	Extra Handling Charge	D	117	3 L
FREIGHT_MIN_EXEMPT	Minimum Amount for Freight Exempt	D	117	4 R
CARRIER_ACCOUNT_DATA	Carrier account number and type	D	118	L
WRITER_COMM_PCT	Writer Overridden Commission Percent	D	119	1 R
INSIDE_COMM_PERCENT	Inside Salesperson Overridden Commission Percent	D	119	2 R
OUTSIDE_COMM_PCT	Outside Salesperson Overridden Commission Percent	D	119	3 R
COMM_REVIEW_DATE	Overridden Commission Percent Review Date	D	119	4 L
OVERRIDE_COMM_REASON	Overridden Commission Reason	D	119	5 L
ALT_BILL_TO_FAX_NBR	Alternate Bill To Fax	D	121	L
INTERNAL_NOTES	Default Internal Notes	D	122	L
LAST_PRICE_CHECK	Flag for using Last Price	D	123	L
CONSIGNED_INVENTORY	Does customer use consignment inventory (1=Y)	D	124	1 L
CONSIGNED_SHIP_TO	Ship-To used for consignment inventory	D	124	2 L
CREDIT_MANAGER	Credit Manager for Credit Queue	D	125	L
ANTICIPATION_CREDIT	Anticipation Credit (Y/N)	D	126	L
UPDATED_DATE	Last date and time record was updated	D	127	L
INCLUDE_SVC_CHARGES	Include service charges in service charge calculation	D	128	L
KEYWORDS	Keywords used for searching	D	129	L
COMMISSION_PLAN	Commission Plan	D	130	L
EMAIL_INDEX	Number of E-mail records	D	131	L
EXCL_FROM_INDEX	Exclude from Index searching	D	132	R
DROP_POINT	Drop Point Branch (Customer)	D	133	L
TIME_ZONE	Time Zone	D	134	L
ROE_USER_LOGIN	User to be messaged upon Login	D	136	1 L
ROE_USER_ORD_CHANGE	User to be messaged with order changes	D	136	2 L
ROE_USER_NEW_ORDER	User to be messaged with new orders	D	136	3 L
ROE_ORDER_QUEUE_USER	User to be messaged for Remote Order Queue	D	136	4 L
PRICE_PRECISION	Price Precision Decimal Places	D	137	R
EXCL_SERVICE_CHARGE	Exclude Service Charge from Service Charge Calc (1	D	138	R
PRODUCT_ZONES	Product zones customer is allowed to buy from	D	139	L
ECOMM_VENDOR_ID	Ecomm Vendor ID	D	140	1 L
ECOMM_WWW_ADDRESS	Ecomm WWW Address	D	140	2 L

**ACTIVANT®**

ECOMM_PRODUCTS	Products, Lines, & Groups allowed for Ecomm	D	140	4	L
ECOMM_CUST_ID	Ecomm Customer ID	D	140	5	L
ECOMM_PASSWORD	Ecomm Password to access account	D	140	6	L
REMIT_TO_OVERRIDE	Remit to override address ID	D	141		L
MIN_CHECK_AMOUNT	Minimum Check Amount	D	142	1	R
MAX_DAYS_SINCE_POSTED	Maximum Days Since Posted	D	142	2	R
BO_PRINT_OPTION	Backorder Print Option	D	143		R
DEMAND_BR	Demand Branch Override	D	144		L
TEMPORARY_EXPORT	Temporary Export	D	145		L
COLLECT_Q_LAST_CALL	Last time customer was called from A/R Collection	D	146	1	R
COLLECT_Q_NEXT_CALL	Next called scheduled from A/R Collection Queue	D	146	2	R
COLLECT_LETTER_SENT	Was collection letter sent to customer	D	146	3	R
COLLECT_LETTER_TYPE	Type of collection letter sent to customer	D	146	4	L
PRINT_STAT_OVERRIDE	Print Status Override	D	147		L
CATEGORY_ID	Category Id	D	148		L
EDI_GRP_ID	EDI ISA Group ID	D	149		L
SOLAR_PROCURE_PRI	Solar Procurement Priority	D	150		R
ROUTE_TYPES	Route Types	D	151		L
ROUTE_VENDORS	Route Vendors	D	152		L
ROUTE_CUSTOMERS	Use Routing Customer	D	153		L
EXCL_CONSIGN_CREDITS	Exclude Consignment Credits (1=Y)	D	154		R
ECOMM_IDS	ECommerce Unique Identifiers for this account	D	155		L
RESTRICT_PRICING	Restrict Pricing in SOE (1=Y)	D	156		R
DROP_POINT_BR	Vendor Drop point branch	D	157		L
ADDITIONAL_NAME	Additional Name	D	158		L
SEPARATE_CHECK_PRINT	Vendor requires separate check printing for each I	D	159		R
STAGING_LOCS	Staging Locations (Customer)	D	160		L
VENDOR_GL_FREIGHT_OVERRID	Freight G/L Account Override (Vendor)	D	160		L
VEND_GL_FREIGHT_OVER	Freight G/L Account Override (Vendor)	D	160		L
CREDIT_RELEASE_PCT	Customer Credit Release Percentage	D	161		R
W9_RECEIVED_DATE	W-9 Forms Received Date	D	162	1	R
CERT_INS_RCVD_DATE	Certificate of Insurance Received Date	D	162	2	R
STAGING_LOCS_BR	Staging Locations Branch	D	163		L
PASS_ALONG_DISC_PCT	Pass Along Discount Holdback Percent	D	164		R
PO_TARGET	Purchase Order Target & Value	D	165		L
REQUIRE_CHECK_VERIFY	Require check verification (1=Y)	D	166		R
EXCL_COLLECT_INVOICE	Exclude Collect Invoices (1=Y)	D	167		R
MASTER_JOB_BID_NBR	Master Job Bid Order Number	D	167		L
CONSOLIDATE_INVOICE	Consolidated Invoice Flag (1=Y)	D	168		R
STATEMENT_CYCLE	Statement Cycle	D	169		L
PULL_CODE	Pull Code	D	170		L
PAY_VIA_EFT	Payment Via Electronic Funds Transfer	D	171	1	L
EFT_PAY_METHOD	Electronic Funds Transfer Payment Method	D	171	2	L
ACH_FORMAT	ACH Format	D	171	3	L
ACH_ROUTING	ACH Routing/Transit #	D	171	4	R
ACH_ACCOUNT	ACH Account #	D	171	5	R
ACH_ACCOUNT_TYPE	ACH Account Type	D	171	6	L
TAX_EXEMPT_EXP_DATE	Tax Exemption Expiration Date	D	172		R
ROUTE_BR	Route Branches	D	173		L
INVOICE_PRINT_STYLE	Invoice Print Style	D	174		L
TAX_CITY_CODE	Tax City Code	D	175	1	L

**ACTIVANT[®]**

TAX_COUNTY_CODE	Tax County Code	D	175	2	L
ASN_VENDOR_TYPE	ASN Vendor Type (Vendor)	D	176		L
VENDOR_HOLD_LIST	Vendor Hold List (Customer)	D	176		L
ACCOUNT_MANAGER	Account Manager	D	177	1	L
HTTP_ADDRESS	HTTP Address	D	177	2	L
ASN_SHIP_CTNR_NBR	ASN Accept shipment container number level	D	178		R
SOE_FILL_RATE_DATA	SOE Fill Rate Data	D	179		L
SOE_FILL_RATE_PCT	SOE Fill Rate Percentage	D	179	1	R
SOE_FILL_RATE_MEASUR	SOE Fill Rate Measurement	D	179	2	L
CONTRACT.UPLOAD.DFLT	Contract Upload Dflts	D	190		L
AP_BAL_120_DAYS	A/P 120 Day Balance	I			R
AP_BAL_30_DAYS	A/P 30 Day Balance	I			R
AP_BAL_60_DAYS	A/P 60 Day Balance	I			R
AP_BAL_90_DAYS	A/P 90 Day Balance	I			R
AP_BAL_CURRENT	A/P Current Balance	I			R
AP_BAL_FUTURE	A/P Future Balance	I			R
AR_BAL	A/R Balance	I			R
AR_BAL_120_DAYS	A/R 120 Day Balance	I			R
AR_BAL_30_DAYS	A/R 30 Day Balance	I			R
AR_BAL_60_DAYS	A/R 60 Day Balance	I			R
AR_BAL_90_DAYS	A/R 90 Day Balance	I			R
AR_BAL_CURRENT	A/R Current Balance	I			R
AR_BAL_DEPOSITS	A/R Deposit Balance	I			R
AR_BAL_FUTURE	A/R Future Balance	I			R
BAL_OVER_60_DAYS	A/R Balance Over 60 Days	I			R
BAL_OVER_90_DAYS	A/R Balance Over 90 Days	I			R
CITY_ST_ZIP	City, State, Zip	I			L
COGS	COGS	I			R
COUNT	COUNT	I			R
CR_LMT_JOB_LIMIT	Credit Limit Job Limit	I			R
CR_LMT_JOB_TOTAL	Credit Limit Job Total	I			R
CR_LMT_PAST_DUE_AMT	Credit Limit Past Due Amount	I			L
CR_LMT_PAST_DUE_DAYS	Credit Limit Past Due Days	I			R
CR_LMT_PAST_DUE_LMT	Credit Limit Past Due Limit	I			R
CR_LMT_TOTAL_AR	Credit Limit Total AR	I			R
GP	GP	I			R
GP%	GP%	I			R
MTD_COGS	Month to Date Cost of Goods Sold	I			R
MTD_GP	Month to Date Gross Profit Dollars	I			R
MTD_GP_PCT	Month to Date Gross Profit Percent	I			R
MTD_SALES	Month to Date Sales	I			R
POINTS_ADJ	Points Adjustments	I			R
POINTS_BAL	Points Balance	I			R
POINTS_EARN	Points Earned	I			R
POINTS_USED	Points Redeemed	I			R
SALES	Sales	I			R
V/C	Vendor / Customer Flag	I			L
VIEWER_ID	Viewer ID	I			R
YTD_COGS	Year to Date Cost of Goods Sold	I			R
YTD_GP	Year to Date Gross Profit Dollars	I			R
YTD_GP_PCT	Year to Date Gross Profit Percent	I			R

Product File Layout Release 8

PRODUCT File Listing

Dictionary ID.....	Description.....	Typ	Attr	Val.	Just
@ID	Item ID	D	0		L
PRODUCT_ID	Product ID	D	0		R
DESC	Product Description	D	1		L
DESCRIPTION	Product Description	D	1		L
GL_PRODUCT_TYPE	General Ledger Product Type	D	2		L
STATUS	Status 1-stock,2-non,3-misc,4-delete,5-review,6-co	D	3		R
KEYWORDS	Product Keywords	D	4		L
PRICING_PRODUCT_ID	Use Pricing From The Following Product	D	5		L
UOM_QTY	The Divisor Used To Determine The Unit Of Measure	D	7		R
INDEX_TYPE	Lookup Index Type P-Primary, C-Catalog, null-Prima	D	8		L
PRICE_LINE	Price Line	D	9		L
UNIT_WEIGHT	Weight Of A Quantity Of 1 Of This Product	D	10		R
PDW_ID	PDW internal product ID	D	11		R
BUY_LINE	Buy Line	D	12		L
LOAD_FACTOR	Load Factor Of A Quantity Of 1 Of This Product	D	13		R
COMMODITY_CODE	Commodity Code	D	14		L
UOM_ABBR	Unit Of Measure Quantity Abbreviation, this field	D	15		L
UOM_QTY_SALES	Default Unit Of Measure Quantity Used In Sales	D	16	1	R
UOM_QTY_PURCH	Default Unit Of Measure Quantity Used In Purchasing	D	16	2	R
UOM_QTY_TRANS	Default Unit Of Measure Quantity Used In Transfers	D	16	3	R
	Default Unit Of Measure Quantity Used In				
UOM_QTY_ADJUST	Adjustments	D	16	4	R
UOM_QTY_INVTY	Default Unit Of Measure Quantity Used In Inventory	D	16	5	R
DEFAULT_UOMS	MV list of 5 default areas to use UOM from attr 15	D	16		L
PROCURE_GROUP	Product procure group ID	D	18		L
SELECT_GROUP	Product select group	D	19		L
SORT_CODE	Product sort code	D	20		R
EXEMPT_GROUPS	Tax Exemption Group, corresponds with PRD<23>	D	22		L
	Tax Exemption Number; Reduced Rate=1; Always				
EXEMPT_CODES	Tax=2	D	23		R
CATALOG_NBR	Product catalog number	D	24		R
CREATE_DATE	Product creation date	D	26	1	L
CREATE_TIME	Product creation time	D	26	2	L
CREATE_USER	Product creation user	D	26	3	L
CREATE_INFO	Product creation information	D	26		L
PER_QTY	Stores default per qty for product	D	35	2	R
BR_PER_QTY	Stores default per qty for product	D	35		L
HAZARD_PACK_GRP	HAZMAT Pack Group	D	40	4	L
COMPONENT_QTYS	Kit component quantites VM per attr 53	D	52		R
COMPONENT_IDS	Kit component product ID's	D	53		R
KIT_IDS	Kit product ID's that this product is a component	D	54		R
SUBSTITUTES	Substitute product ID's	D	55		R
SUBSTITUTES_BRS	Substitute branches corresponding to attr 55	D	56		L

**ACTIVANT™**

SUBSTITUTES_NOTES	Substitute notes corresponding to attr 56	D	57	L
OE_REMINDER_MSG	VM list of reminder notes	D	58	L
ADD_DEMAND_BASE	W - Weight L - Load otherwise - Units	D	60	L
SERIAL_LAST_ASSIGNED	Last serial number assigned to this product	D	61	L
PRIMARY_UPC_CODE	Primary UPC code	D	63	1 L
UPC_CODE_USER_2	UPC code - user defined 2 - cross ref in PU.IDX file	D	63	2 L
UPC_CODE_USER_3	UPC code - user defined 3 - cross ref in PU.IDX file	D	63	3 L
UPC_CODE_USER_4	UPC code - user defined 4 - cross ref in PU.IDX file	D	63	4 L
UPC_CODE_USER_5	UPC code - user defined 5 - cross ref in PU.IDX file	D	63	5 L
UPC_CODE_USER_6	UPC code - user defined 6 - cross ref in PU.IDX file	D	63	6 L
UPC_CODE_USER_7	UPC code - user defined 7 - cross ref in PU.IDX file	D	63	7 L
UPC_CODE_USER_8	UPC code - user defined 8 - cross ref in PU.IDX file	D	63	8 L
UPC_CODE_USER_9	UPC code - user defined 9 - cross ref in PU.IDX file	D	63	9 L
UPC_CODE_USER_10	UPC code - user defined 10 - cross ref in PU.IDX file	D	63	10 L
UPC_CODE_USER_11	UPC code - user defined 11 - cross ref in PU.IDX file	D	63	11 L
SECONDARY_UPC_CODES	SVM list of all secondary UPC codes for product	D	63	12 L
PU_IDS	MV list of all UPC codes for this product	D	63	L
ADD_DEMAND_PNS	VM list of product ID's to use in calculating demand	D	64	R
ALT_DESC	Alternative product description	D	66	L
MERGE_PN	Product merge part number to keep	D	67	R
HAZMAT_CLASS	Hazardous class for product	D	80	1 L
HAZMAT_CLASS_DESC	Hazardous class description	D	80	2 L
HAZMAT_ID_NBR	Hazardous data ID number	D	80	3 L
HAZMAT_CODE	Hazardous data code	D	80	5 L
HAZMAT_DATA	Hazardous data for product	D	80	L
CNTR_WHSE_TYPE	Central Warehouse Type: Top-Down, Bottom-Up	D	83	L
COMPONENT_CMTS	Kit component comments - MV by component stored in	D	85	L
KIT_PRICE_OPTS	Kit price options are (2,3 only valid for non-dynamic)	D	86	1 L
PRINT_KIT_COMPS	Flag to print kit components	D	86	2 R
KIT_COGS_OPTS	Kit cogs options are (1,2 only valid for non-dynamic)	D	86	3 L
KIT_COMM_OPTS	Kit comm options are (1,2 only valid for non-dynamic)	D	86	4 L
KIT_OPTIONS	Kit options	D	86	L
COMPONENTS_SPOIL	Spoilage per component in attr 53 - VM delimited	D	87	R
PN_CALC_TYPE	Flag that will determine if points are calculated	D	88	L
GL_INV_ACCT_OVRD	Inventory Account Override	D	89	R
CERTIFICATION_REQD	VM list of certification ID's	D	91	L
BUDGET_GROUP_ID	Sales Budget Group ID	D	92	L
XREF_AGENT	XREF Agent	D	93	1 L
XREF_DESC	XREF Description	D	93	2 L
XREF_PARAM_DATA	XREF Parameter Data	D	93	3 L
XREF_DATA	External Reference Data	D	93	L
SUBSTITUTES_TYPE	VM list of substitute types corresponding to attri	D	95	L
SUBSTITUTES_SUG_QTY	VM list of substitute products suggested selling q	D	96	R
WIP_ORDER_TEMPLATE	Product work order template	D	100	1 L
WIP_INCOMING_QTY	WIP incoming qty that will be built when this temp	D	100	2 R
WIP_PRICING	WIP pricing method default	D	100	3 R
WIP_COSTING	WIP costing method default	D	100	4 R
WIP_SERIAL_PROMPT	Product work order serial number prompt check	D	100	6 R
WIP_SERIAL_MASK	Set serial number mask to use when auto calcing ne	D	100	7 L
WORK_ORDER_INFO	Product work order information - Please see indivi	D	100	L
TREAD_DEPTH	Original product tread depth	D	103	R

**ACTIVANT™**

PREPAID_FET	Y/N flag for prepaid FET	D	104		R
DIRECT_SHIP_ITEM	Y/N flag for direct ship item	D	105	1	R
DIRECT_SHIP_VENDOR	Vendor to direct ship item from	D	105	2	R
DIRECT_SHIP_INFO	Flag for direct shipment item	D	105		L
IS_DYNAMIC_KIT	Y/N flag if product is a dynamic kit	D	106		R
DUTY_CODE	Duty harmonizing code	D	107	1	L
DUTY_CODE_COUNTRY	Duty harmonizing code country of manufacture	D	107	2	L
DUTY_CODE_INFO	Duty harmonizing code information	D	107		L
LAST_UPDATE_DATE	Product last update date	D	108	1	L
LAST_UPDATE_TIME	Product last update time	D	108	2	L
LAST_UPDATE_INFO	Product last update information	D	108		L
MSDS_ID	MSDS Sheet internal ID	D	109		L
ADD_DEMAND_EXP_DTS	VM list of expire dates to use with attr 64 to calculate dmd	D	111		L
ADD_DEMAND_PNS_TBRS	VM list of branches to use with attr 64 to calculate dmd	D	113		L
ADD_DEMAND_CN_INFO	VM list of customer information to use when calculating customer demand	D	114		L
SUB_PRODS	Associated sub-products used for master	D	116		R
SUB_PROD_OPTIONS	Associated sub-product options MV to attr 116	D	117		L
AUTH_BRANCHES	VM list of branches authorized for product. If null, all branches are authorized	D	118		L
MASTER_PN	If this product is a sub-product, store the master PN	D	119		R
EROUTE_DELIVER_TIME	E-route time to deliver product	D	120		R
EROUTE_VOLUME	E-route special volume	D	121		R
ORD_INV_FACTOR	Order to inventory factor in SOE	D	122		R
SECONDARY_BLINES	VM list of secondary buy-lines	D	124		L
BLINE_BUY_PACKS	VM list of buy pack overrides corresponding to attr. 124	D	125		R
BLINE_BUY_DIVS	VM list of buy pack divisible corresponding to at	D	126		R
DISABLE_DUP_CHECK	Flag to disable duplicate product check in SOE	D	127		R
PRODUCT_ZONES	Zones product is valid to sell in - all valid zone	D	128		L
PRODUCT_ZONES_INCL	VM list of Include/Exclude corresponding to attr 1	D	129		L
RESTRICT_PRICE_CHNG	Y/N flag to restrict price changes in SOE	D	130		R
SECONDARY_SERIAL_NBR	Product secondary serial number	D	131		L
OE_REMINDER_BR	Per reminder note in VM position in attr 58, this	D	132		L
OE_REMINDER_AREA	Per reminder note in VM position in attr58, this w	D	133		R
PN_LENGTH	Product length used for laminate cut mod	D	134		R
PN_WIDTH	Product width used for laminate cut mod	D	135		R
PN_DEPTH	Product depth used for laminate cut mod	D	136		R
DETAIL_LOT_HOLD	Exclude detail lot from available if on hold	D	137	1	L
DETAIL_LOT_QUALITY	Exclude from detail lot if quality or rank are not	D	137	2	L
DETAIL_LOT_INFO	Detail lot information	D	137		
MOD_PROC_STEPS	Modified product processing steps	D	138		L
MOD_PROC_LEAD	Modified product processing leads	D	139		L
WIZARD_SUBROUTINE	Product wizard subroutine	D	140		L
WIZARD_TAG_DATA	Tag data used for wizard subroutine - free form data	D	141		L
TAG_ALONG_PNS	Required tag along products for substitute items.	D	143		R
PARENT_TAG_ALONG	Product that this product is a tag along for	D	144		R
KIT_KEY_FLAGS	VM list of flags setting for each component.	D	145		R
PRORATE_KIT_PRICE	Prorate kit price/cost across line items using LIS	D	146		R
PROD_BR_ID	Internal Id for the Product Branch File	I	0		R

**ACTIVANT™**

LEAD_TIME	Lead Time in Days	I	0	R
SELL_QTY	Minimum Sales Quantity	I	1	R
MANUAL_MIN	Manual Minimum Inventory Level for Purchasing	I	2	R
MANUAL_MAX	Manual Maximum Inventory Level for Purchasing	I	3	R
BUY_PKG	Purchase Package Quantity	I	4	R
TREND	Trend Percentage	I	5	R
SERVICE_STOCK	Service Stock Quantity	I	6	R
LOST_SALE	Lost Sales Percentage	I	7	R
LOST_SLS_%	Lost Sales Percentage	I	7	R
EXCP_SLS_%	Exceptional Sales %	I	8	1 L
FORECAST_METHOD	Forecast Method S = Standard M = Median	I	8	2 L
BTQ	Backorder Tolerance Quantity	I	8	3 R
BCKRD_TLRNC_QTY	Backorder Tolerance Quantity	I	8	R
FCAST.METH	Forecast Method	I	8	L
EOQ_PERCENT	Economic Order Quantity Carrying Cost Percentage	I	9	R
EOQ_%	EOQ %	I	9	R
EOQ_DOLLARS	Economic Order Quantity Line Item Purchasing Cost	I	10	R
EOQ_\$	EOQ \$	I	10	R
PROD_CONTROL	Location Control for the Product	I	11	L
STOCK_FLAG	Branch Specific Stocking Control Y Always Stock, N	I	12	L
SAFETY_FACTOR	Safety Factor	I	14	R
SEASONAL	Item Seasonality	I	15	L
MANUAL_EXPIRE_DATE	Expiration Date for Manual Min and Max	I	16	R
LEAD_FACTOR	Lead Factor Determines the number of receipts to be	I	17	R
LABEL_PER	Label Per controls the number of labels to be printed	I	18	R
FREIGHT_FACTOR	DELETE ME	I	19	R
FREIGHT_CHG	Freight Charge	I	19	R
PRICE_SHEET	Vendors Price Sheet	I	20	L
DFLT_LOCATION	Default Location for the Product	I	21	L
DISCOUNT_CLASS	Vendor's Discount Class	I	22	L
BUY_GROUP	Buy Group	I	23	L
SELL_GROUP	Sell Group	I	24	L
SERIAL	Serial Number Tracking Control	I	25	L
COMM_GROUP	Commission Group	I	26	L
BUY_PKG_DIVISIBLE	Divisibility Flag for Buy Package Quantity	I	27	L
CALC_DEMAND	Include Item in Demand Calculation for Central Purch	I	28	L
BR_HITS	Minimum Number of Hits to Stock Item in the Branch	I	29	R
NETWORK_HITS	Minimum Number of Hits throughout the Central Dist.	I	30	R
POINTS_DATA	Customer Points Program Data	I	31	R
CUST_SERVICE_STOCK	Customer Specific Service Stock Data, Multi-Valued	I	32	R
MIN_LEAD_FACTOR	Minimum Number of Receipts to use for lead time calc	I	33	R
EDI_VMI	EDI Vendor Managed Inventory	I	34	L
EDI	Minimum Lead Factor	I	34	L
PRICE_SHEET_PRT	Price Sheet Print Flag	I	35	L
QTY_BRK_QTY	Multiple value quantity break quantities by matrix	I	36	R
QTY_BRK_UOM	Multiple value quantity break quantities unit of m	I	37	L
PASS_DISC	Percentage for Pass Along Vendor Discount	I	38	R
FREIGHT_FACTOR_PCT	Freight Factor Percentage	I	39	R
BR_LIST	Multiple Value listing of branches and territories	I	40	L
FP_REG_HIT	Forecasting parameter for non-seasonal items	I	41	R
FP_REG_MAX_DAY	Forecasting parameter for non-seasonal items	I	42	R

**ACTIVANT[®]**

FP_REG_MIN_DAY	Forecasting parameter for non-seasonal items	I	43	R
FP_MAX	Forecasting parameter for seasonal items	I	44	R
FP_SNL_HIT	Forecasting parameter for seasonal items	I	45	R
FP_SNL_MAX_DAY	Forecasting parameter for seasonal items	I	46	R
FP_SNL_MIN_DAY	Forecasting parameter for non-seasonal items	I	47	R
FP_AUTOTREND	Use Auto-Trend Percentage Calculations	I	48	L
	Include Direct Shipments in the calculation of daily demand	I	49	L
FP_INCLUDE_DIR		I	49	L
MIN_GP_PERCENT	Minimum Gross Profit Percentage	I	50	R
LEAD_DAYS	Lead Days	I	51	R
RESTRICT_PRC_CHNG	Restrict Price Change in Sales Order Entry Flag	I	58	L
SALES_HITS	# of Sales Transactions Per Year	I		R
AVAIL	Available Inventory	I		R
AVG_PRICE	AVERAGE PRICE	I		R
AVG_SALE_QTY	Average Sale Quantity	I		R
COUNT	COUNT	I		R
DAYS_OUT	Days Out of Stock	I		R
DEMAND_DAY	Demand in Units/day	I		R
DEMAND_PERIOD	Demand Period	I		R
DOLLAR_SIGN	DOLLAR_SIGN	I		L
EOQ	EOQ	I		R
EXCP_QTY_EXCLUDED	Largest Sale within forecast period That is excluded	I		R
LAST_DEMAND_CALC	Last Demand Calc Date	I		R
LOW_SALE	Low Sale Qty	I		R
AVG_COST	Moving Average Cost	I		R
NO_PRINT	Price Book No print Flag	I		L
BASIS	Price Line Basis; needs column data or set common	I		R
PRICE_LINE_SEQ	Price Line Sequence	I		R
RANK_BR1	Prod Rank in Branch 1	I		L
RANK	Product Rank	I		L
HI_TRANS_QTY	Qty of Largest Transaction	I		R
RAW_DEMAND	Raw demand for forecast period	I		R
RAW_HITS	Raw Hits for Demand Period.	I		R
SUM_SALES_HITS	Summation of Sales Hits in all branches.	I		R
VIEWER_ID	VIEWER_ID	I		L

AR File Layout Release 8

Dictionary ID.....	Description.....	Typ	Attr	Val.	Just
AR_ID	AR ID	D	0		L
AP_CHECK_NBR	Disbursement Check Number	D	1		L
CHECK_NBR	Customer's Check Number from Cash Receipt	D	1		L
INV_NBR	Accounts Payable Vendor Invoice Number	D	1		L
PO_NBR	PO Number from Sales Order Header (Indexed)	D	1		L
APPLIED_ID	ID of Ledger record applied to this transaction. F	D	2		L
APPLIED_DATE	Application Dates MV'd by Application	D	3		L
BR_GL	General Ledger Postings: Branch~G/L ID MV'd by App	D	4		L
GL_AMT	General Ledger Amounts Posted MV'd by Application	D	5		R
BALANCE_DUE	Balance Due	D	8		R
AR_BAL_DUE	AR Balance Due	D	8	1	R

**ACTIVANT™**

AP_BAL_DUE	AP Balance Due	D	8	2	R
UBAP_BAL_DUE	Unbilled AP Balance Due	D	8	3	R
DISCOUNT_AMT	Discount Amount (Original Amount)	D	9		R
BILL_TO_CUST_ID	Bill-to Customer ID	D	10		R
PAY_TO_VEND_ID	Pay-to Vendor ID	D	10		R
DISCOUNT_DATE	A/R Discount Date (A/R, A/P, XFER)	D	11		R
DUE_DATE	Invoice Due Date (A/R, A/P, XFER)	D	12		R
SHIP_DATE	G/L Posting Date (A/R, A/P, XFER)	D	13		R
SHIP_FROM_VEND_ID	Ship-from Vendor ID	D	15		R
SHIP_TO_CUST_ID	Ship-to Customer ID	D	15		R
AP_DEDUCT_AMT	A/P Deduct Amount	D	16		R
AP_DEDUCT_REASON	A/P Deduct Reason	D	17		L
HANDLING_CODE	Special Handling Code	D	18		L
AP_DISC_OVRD_FLAG	A/P Discount Override Flag (1 = Disc has been over	D	19		R
STATUS	Status Code (I=Invoice, \$=Payment, S=Shipped Xfer, A/P Payment Info (MV1=Payment Amount, MV2=Discount	D	20		L
AP_PMT_INFO		D	21		L
PAY_ON_DATE	A/P Pay On Date(s) (for Payable records it is Pay	D	22		L
AP_IS_PAID_OFF	A/P Invoice Paid Off Flag (1=Payable Paid in Full)	D	23		R
CR_AMOUNT	Cash Receipt Amount	D	24		R
GRACE_DAYS	Number of Grace Days from Terms Code	D	26		R
INV_HOLD_CODE	Invoice Hold Code	D	27		L
FRT_INFO	Freight vendor information.	D	28		L
FRT_PAYABLE_PO	Associated Purchase Order number for Freight Vendor	D	29		L
CURRENCY_INFO	Currency Information (MV1=Currency Type, MV2=Exchange	D	31		R
AP_AMTS	AP Amounts	D	33		R
GL_SHORT_DESC	G/L Postings Short Description (Displays in Alt ~)	D	34		L
REV_CHECK_DISBID	Reversed Check Disbursement ID for a Payable	D	35		L
APPL_IDS	Application Ids	I			L
AR_AMT	AR Amount	I			R
BILL_IN_FGHT	BILLABLE INCOMING FREIGHT	I			R
BILL_IN_HANDL	BILLABLE INCOMING HANDLING	I			R
BILL_OUT_FGHT	BILLABLE OUTGOING FREIGHT	I			R
BILL_OUT_HANDL	BILLABLE OUTGOING HANDLING	I			R
CASHBOX_AMT	Cashbox Amount	I			R
CASH_AMT	Cash Amount	I			R
COGS_AMT	Cost of Goods Sold Amount	I			R
COUNT	Counter: Returns a "1" which can be total to addr	I			R
DISC_AMT	Cash Receipts - Discount Amount	I			R
EXP_IN_FGHT	EXPENSE INCOMING FREIGHT	I			R
EXP_IN_HANDL	EXPENSE INCOMING HANDLING	I			R
EXP_OUT_FGHT	EXPENSE OUTGOING FREIGHT	I			R
EXP_OUT_HANDL	EXPENSE OUTGOING FREIGHT	I			R
FGHT_IN_AMT	Freight In Amount	I			R
INVOICE_GEN	Invoice Generation	I			R
LEDGER_ID	AR ID	I	0		L
PAYMENT_DAYS	Payment Days	I			R
SALES_AMT	Total Sales Amount	I			R
TAX_AMT	Sales Tax Amount	I			R
TAX_JUR	Tax Jurisdiction	I			R
TRANS_TYPE	Transaction Type	I			L

VIEWER_ID

VIEWER ID



PSUB File Layout Release 8

PSUB File Release 8

Dictionary ID.....	Description.....	Typ	Attr	Just	Width
PSUB_ID	Unique string of concatenated data	D	0	L	70
QTY	Quantity Moved	D	1	R	10
PRICE	Selling Price / Purchasing Cost / Adjustment	D	2	R	12
ACCOUNTING_COST	Accounting Cost (Cost of Goods Sold) / N/A on Purchase Order	D	3	R	12
DOLLAR_ENTITY_ID	Dollar Entity @ID	D	4	R	8
PRODUCT_ENTITY_ID	Product Entity @ID	D	5	R	8
OVERRIDE_PRICE	Original (Matrix Determined) Override Selling Price	D	6	R	12
OVERRIDE_COST	Original (Matrix Determined) Override Accounting C	D	7	R	12
KIT_PROD_ID	Kit Product @ID	D	8	L	10
SERIAL_NBR	Multi-valued list of serial numbers	D	9	L	30
PRICE_BR	Pricing Branch	D	10	L	4
CONSIGN_FLAG	Consignment Flag: S - Shipping Into Consigned Inventory	D	11	L	1
LOT_ITEM	Lot Billing Flag	D	12	L	1
ORDER_TIME	Time order was processed	D	14	L	10
@Ak.0	@Ak.0	I		L	10
BR	Primary Branch (shipping or pricing based on record)	I		L	4
COMPONENT_POS	Kit Component Position	I		L	3
DIFF_BR	Other branch when pricing and shipping branch are	I		L	4
EXT_COST	Extended Cost	I		R	14
EXT_PRICE	Extended Price	I		R	14
FULL_ORD_ID	Full Order ID	I		L	14
INVOICE_NBR	Invoice Number	I		L	3
LED_DET_ID	Ledger Detail ID	I		L	4
LOCATION	LOCATION	I		L	12
ORD_ID	ORD_ID	I		L	10
PRODUCT_ID	PRODUCT_ID	I		R	10
QTY_TYPE	Quantity Type	I		L	1
SALE_AMT	Sale Amount	I	0	R	12
SALE_QTY	Sale Quantity	I	0	R	9
SHIP_DATE	Shipping Date	I		R	10

Appendix D

Subroutines

The following subroutines include explanations and examples:



ACTIVANT™

Dictionary Subroutines

DICT.COMMON = Dictionary Common Prompts which is the prompts hotkey in dictionary maintenance. Within this document you will see reference to this terminology with a numeric value after the DICT.COMMON. The numbers represent the following information:

- 1 Branch/Territory/All (Multi value list of branches)
- 2 Enter Br/Tr/All
- 3 Enter Start Date
- 4 Enter End/As of Date
- 5 Discount Class
- 6 Product Location
- 7 Price Basis Name
- 8 Location Number
- 9 Multi Value Pos
- 10 Enter Br
- 11 Generic Prompt
- 12 Ignore Branch Hierarchy

DICT.AR.FIRST.SALE

This subroutine is used from the AR file and will return a “1” if the order is the first generation of the invoice. A dictionary using this subroutine can be used for statistical reporting on the occurrences of a first generation order to the total number of generations that occurred for an order.

The I-Descriptor screen of the dictionary item would look like:

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	FIRST.SALE
SUBR('DICT.AR.FIRST.SALE')	

DICT.AR.ITEM.CNT

This subroutine is used from the AR file and counts the number of items on the generation of the invoice passed back. A dictionary using this subroutine can be used for statistical reporting relative to how many items are on each invoice generation. (Have you ever wanted to find out your average number of items that ship out on a generation? This is data that will help you to obtain this statistic.

The I-Descriptor screen of a dictionary using this subroutine would look like:

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	ITEM.CNT
SUBR('DICT.AR.ITEM.CNT')	

DICT.AR.JLI

This subroutine is used from the AR file and is part of the indexing. The information that is returned consists of three segments delimited by a period. The first segment is a numeric value that returns the type of transaction which is listed below. The second segment is the date of the posting and the third segment will return the last two digits of the order number (not the generation of the order).

Example of data returned from S1008886.005 is: 1.11176.86

- 1 = Sales Order
- 2 = Cash Receipt
- 3 = Payment on a Sales Order
- 4 = Purchase Order
- 5 = Transfer
- 6 = Sales Order that is paid through cash receipts
- 7 = Payable
- 8 = Disbursement
- 9 = Journal Entry
- 10 = Inventory Adjustment
- 11 = Work Order
- 12 = Rental

DICT.AR.LEDL.INFO

This subroutine is used from the AR file. The two requirements when using this subroutine in an I-descriptor is to pass in the @ID and the word the system needs to find in the change log for the transaction.

The system will return the user, date, time, port and change log as shown below.

S0704030.001

DAVIDB 03/21/00 02:51pm /pts/4 DAVIDB Authorized : Overcommitted

An example of an I-Descriptor using this subroutine is:

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	LEDL.OVER
SUBR('DICT.AR.LEDL.INFO',@ID,'Overcommit')	

DICT.AR.LEDL.PRT

Similar to the subroutine above it is used only from the AR file. This routine will return the date, time and user when a ship ticket, pick ticket, invoice, cash sale etc. has been printed. All occurrences will display for a generation of the order.

For example the data would return:

S0422603.001 DAVIDB 09/04/96 11:54am /tty10
Printed **** Invoice **** on 0

DAVIDB 09/19/96 03:06pm /tty10
RePrinted **** Invoice **** on 0

DAVIDB 11/04/96 11:41am /tty10
RePrinted **** Invoice **** on 0

An example of the I-Descriptor screen for a dictionary item using this subroutine would look like:

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	INV.PRINT
SUBR('DICT.AR.LEDL.PRT' ,@ID)	

DICT.AR.PAY.DAYS

This subroutine is used from the AR file. It will return the number of days it took for the customer to pay the invoice.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	PAYMENT_DAYS
SUBR("DICT.AR.PAY.DAYS")	

DICT.AR.PRT.STAT

From the AR file this routine will display the print status of a transaction.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	PRT.STATUS
SUBR('DICT.AR.PRT.STAT')	

DICT.AR.SALE

Basically will return a « 1 » for each transaction that is an AR sale. Items excluded from this are Journal entries, first generation of Transfers, etc.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	AR.SALE
SUBR('DICT.AR.SALE')	

DICT.AR.UNPD.AMTS

This subroutine will return unpaid amounts for split payments.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	AMT.TO.PAY
SUBR('DICT.AR.UNPD.AMTS')	

DICT.AR.UNPD.DISC

This subroutine will return the discount amount to pay on split payments

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	DISC.TO.PAY
SUBR('DICT.AR.UNPD.DISC')	

DICT.ASUB.AMT

SUBROUTINE (AMT,GLCODES)

This routine requires the GL Code for which you want to report amounts. Also, this routine calls in another routine called ASUB.GET.AMT.

This routine is design to return balance for a given set of auto-postings account (AR/AP/UBAP) Generally only one account in passed in and the AR record is processed parsing through strings in ARREC<4> and AREEC<5> filtering by start date, end date, branch, and VM positions finally returning a balance in AMT variable

```
*-----
*** AMT  - Total balance for GAS account for given selection  [OUT]
*** ARREC - Dynamic AR record                                [IN]
*** POS   - Start and end @VM positions for balance calculation  (IN)
*** BRCHS - Valid Branches                                    (IN)
*** GAS   - will represent GL Account #s to return the GL balance (IN)
              (Could be MV)
*** SDT   - Start Date                                        (IN)
*** EDT   - End Date                                         (IN)
*** ADDL.DATA - Additional information                        (IN)
*** @AM1 = CN - Bill-To customer (UBAP only)
```


An example of an I-descriptor using this subroutine:

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	A/R
SUBR('DICT.ASUB.AMT','AR')	

DICT.ASUB.ARBAL

This routine calls in another routine called AR.GET.BAL

The AR.GET.BAL calls in another routine called ASUB.GET.BAL

An example of an I-descriptor using this subroutine:

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	ARBAL
SUBR('DICT.ASUB.ARBAL')	

DICT.BR.VAL.GET

SUBROUTINE (VAL,BR,PN,ATTB.NO,SVM.NO,LOC.FLAG)

Subroutine: DICT.BR.VAL.GET

This routine pulls branch specific data from the PROD.BR file or the PROD.CALC.BR file calling either PRD.BR.GET.VAL or PRDC.BR.GET.VAL to get a specific attribute for a branch. If no branch is specified then all branches are used.

BR	- Branch	[IN]
PN	- Part Number	[IN]
ATTB.NO	- Attribute Position	[IN]
SVM.NO	- Sub-Value Mark	[IN]
LOC.FLAG	- Flag to set whether to pull from PROD.CALC or PROD.BR	[IN]
VAL	- Value Returned	[OUT]

An example of an I-descriptor using this subroutine is:

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	DAYS_OUT
SUBR('DICT.BR.VAL.GET',' ',@ID,11,' ','C')	

DICT.BR.VALS

This subroutine is used with the PROD.LIFO file.

SUBROUTINE (VAL,ATTB)

You need to enter the attribute number that is storing branch specific data as part of this subroutine.

DICT.CALC.AVAIL

This routine can only be used when the @id of the file you are working in is the Internal ID number of the product. Ex. PRODUCT.NOTES, PROD.DYNAM, PRODUCT

-----*

This dictionary subroutine calculates the available quantity of a given product for selected branches.

-----*

An example of an I-descriptor using this subroutine is :

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVAIL
SUBR("DICT.CALC.AVAIL")	

DICT.CALC.AVAIL.PN

SUBROUTINE (STK.LEVELS,PN)

This routine can be used from any file for which you can pass in the Internal ID number of an Eclipse product.

Works the same as DICT.CALC.AVAIL.PN shown above.

An example of an i-descriptor using this subroutine is :

I-Descriptor Program Maintenance	
File Name :	ENTITY.PN.IDS
Dict ID :	AVAIL
SUBR('DICT.CALC.AVAIL.PN',PN)	

DICT.CALC.CDC.AVAIL

SUBROUTINE (CDCAV)

Routine to calculate CDC Availability (Central Distribution Center)

An example of an I-descriptor using this subroutine is :

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVAIL.CDC
SUBR("DICT.CALC.CDC.AVAIL")	

DICT.CALC.CN.SLS

Any file for which you can pass in the Customer's internal ID can use this subroutine. This subroutine requires 6 arguments to be entered for it to work. If you want to use the user defined date range and you are not specifying a specific branch, make sure you load the prompts on your dictionary item. (Prompts hotkey from dictionary maintenance). The three prompts to add is the Branch, Starting date and End as of Date.

```

SUBROUTINE (AMT,RANGE,AVN,BRCHS,SD,ED,CN)
*-----Date Ranges
*      1 = User Defined Date Range
*      2 = MTD Date Range
*      3 = YTD Date Range
*      4 = Fiscal MTD Date Range
*      5 = Fiscal YTD Date Range

*-----AVN Definition
*      1 = Sales
•      2 = Gross Profit

```

An example of an I-descriptor you can create using this subroutine is :

I-Descriptor Program Maintenance	
File Name :	ENTITY.PN.IDS
Dict ID :	CUST.SALES
SUBR('DICT.CALC.CN.SLS','1','1','','',' ',CUST.ID)	

DICT.CALC.PBR.AVAIL

Can only be run from a file where the @ID of each record is the internal ID of the product. Ex. PRODUCT, PROD.DYNAM and PRODUCT.NOTES
Routine to calculate Pricing Branch's Availability.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVAIL.PBR
SUBR("DICT.CALC.PBR.AVAIL")	

DICT.CALC.SLS

Can only be run from the ENTITY file

SUBROUTINE (AMT,RANGE,AVN,BRCHS,SD,ED)

Version# 1 - 03/24/1996 - 08:30pm - SCOTTR - **

-----Date Ranges

- 1 = User Defined Date Range
- 2 = MTD Date Range
- 3 = YTD Date Range
- 4 = Fiscal MTD Date Range
- 5 = Fiscal YTD Date Range

-----AVN Definition

- 1 = Sales
- 2 = Gross Profit
- 3 = Gross Profit Percentage
- 4 = Cost of Goods Sold

I-Descriptor Program Maintenance	
File Name : ENTITY	
Dict ID : COGS	
SUBR('DICT.CALC.SLS',1,4,'',' ','')	

DICT.CALC.SLS.ST

Can only be run from the ENTITY file and will return values if the ENTITY is a Ship to customer.

SUBROUTINE (AMT,RANGE,AVN,BRCHS,SD,ED)

-----Date Ranges

- 1 = User Defined Date Range
- 2 = MTD Date Range
- 3 = YTD Date Range
- 4 = Fiscal MTD Date Range
- 5 = Fiscal YTD Date Range

-----AVN Definition

- 1 = Sales
- 2 = Gross Profit
- 3 = Gross Profit Percentage
- 4 = Cost of Goods Sold

I-Descriptor Program Maintenance	
File Name : ENTITY	
Dict ID : SALES.STONLY	
SUBR('DICT.CALC.SLS.ST',1,1,'',' ','')	

DICT.CMP.COST.CALC

SUBROUTINE (CMP.COST)

This routine will only work from the AR file.

This will return the cost for the entire generation of the AR record.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	CMP.COST
SUBR("DICT.CMP.COST.CALC")	

DICT.CONV.CHAR

SUBROUTINE (NEW.STR,OLD.STR,OLD.CHAR,NEW.CHAR)

This routine will convert a string of characters to a new string of characters. You will need to pass in the data element that stores the string of characters as well as the original characters and the new characters that should be outputted.

DICT.CONV.VM

SUBROUTINE (VAL,ATTB)

This subroutine will work on any file for a data element that has multiple value string of information. You need to pass in the Attribute value that you want to convert from multi-value marks to a " " (space) delimited.

For example, the product file has multiple values in the description. Therefore if I wanted to return the last word of the description, I would first need to convert the multi-value marker to a space so that I can count all of the separate words. Once I have completed that task I can return the last word.

This is useful if you want to pull back the last value in a multi-valued field.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LAST_WORD
SUBR('DICT.CONV.VM',1); DCOUNT(@1," "); FIELD(@1," ",@2)	

DICT.COST.RATIO

SUBROUTINE (PER.QTY,PN)

This subroutine is used from the PRODUCT file and will provide the Per.qty uom that is located on the price sheet screen from product file maintenance

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	COST_RATIO
SUBR('DICT.COST.RATIO',@ID)	

DICT.CRED.CTRL.GET

This routine works off of the ENTITY file and will return the full descriptions of the credit control parameters that are set for the customer.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	CREDIT.CTRL
SUBR('DICT.CRED.CTRL.GET')	

and will return:

36557

Print approval required message when
credit limit exceeded
Company checks accepted
Personal checks accepted

DICT.CUS.FAX.NO

This subroutine will do a search in the contacts area of the entity file to find the terminology "FAX". It will do a Dcount to count down the number of value markers until it finds "FAX". Then it will go to the phone numbers and return the correct phone number that is next to the FAX.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	FAX_NUMBER
SUBR('DICT.CUS.FAX.NO')	

DICT.CUS.NOTES

SUBROUTINE (NTS,NOTE.ID)

This subroutine will go out to the ENTITY.NOTES file and return the note number specified. This will only work from the ENTITY file.

DICT.CUS.POINTS

This subroutine needs to know an option that you want to pull back for the Customer points program. This routine works from the ENTITY file

SUBROUTINE (AMT,OPT)

The options range from 1 to 5.

1 = Ending Balance

2 = YTD Earns

3 = YTD Use

4 = YTD Adjustments

5 = YTD Beg Balance\

DICT.DISB.AMT.GET

Routine to return the amount paid to this vendor in the date range specified

This subroutine can only be used from the ENTITY file.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	VEND.DISB.AMT
SUBR('DICT.DISB.AMT.GET')	

DICT.EARLIEST.SHPDT

SUBROUTINE (EARLIEST.DATE,OID,GID)

This subroutine will pull back the product's earliest available date on a transaction if the order id and the generation id can be passed in. This is pulling back the earliest avail date for the entire generation.

I-Descriptor Program Maintenance		View Only
File Name :	ORDER.QUEUE	
Dict ID :	EARLIEST.DATE	
SUBR('DICT.EARLIEST.SHPDT',OID,GID)		

DICT.ENT.45.AR.AMTS

This subroutine is only used on the Entity file and will pull back the balance greater than or equal to 45 days and less than and equal to 60 days.

I-Descriptor Program Maintenance		View Only
File Name :	ENTITY	
Dict ID :	BAL.45DAYS	
SUBR("DICT.ENT.45.AR.AMTS")		

DICT.ENT.ALL.SLS

SUBROUTINE (AMT,STD,EDD,AVN)

 This subroutine gets all the sales dollars for a particular customer for all of the user's authorized branches between the supplied start and end dates.

AMT - Total amount returned.. (OUT)
 STD - Start date to get the amounts for... (IN)
 EDD - End date to get the amounts for. (IN)
 AVN - What dollar amount you want returned. [IN]

 AVN = 1 : Sales Dollars are returned.

 AVN = 2 : GP Dollars

 AVN = 3 : GP%

 AVN = 4 : COGS Dollars

DICT.ENT.AP.AMTS

SUBROUTINE (VAL,VAL.NO)

This subroutine can only be used from the ENTITY file and needs a value number passed in. The value number is used in another subroutine that is used called AR.PREV.LIST.
 Subroutine - AR.PREV.LIST

-----*
 This subroutine will retrieve open AR/AP items for the Customer (CN) and the As of Date (AOD) passed in and parses out certain data, such as the AR Record Open Balances, Payment Dates, and GL Branches.
 -----*

CN - Customer Number to get data for. [IN]
 AOD - As of Date to get data for. (IN)
 AGD - Age as of Date -> Age data as of this date. (IN)
 AGT - Aging Balance Summary information, delimited by age bckt (OUT)
 IDS - AR Records we picked up for the Customer & As of Date. (OUT)
 DTS - G/L Date for the AR Records picked up. (OUT)
 BALS - Open Balance for each AR Record passed back in IDS (OUT)
 CRS - Credit Amount if the AR Record if it is in fact a credit (OUT)
 PDTS - Payment Dates (OUT)
 AGS - Aged Buckets that each of the AR Recs fall into (OUT)
 - 1 = Future
 - 2 = Current
 - 3 = 31-60 Days
 - 4 = 61-90 Days
 - 5 = 91-120 Days
 - 6 = Over 120 Days
 - 7 = Deposits

DICT.ENT.AR.AMTS

SUBROUTINE (VAL,VAL.NO)

This subroutine can only be used from the ENTITY file and needs a value number passed in. The value number is used in another subroutine that is used called AR.PREV.LIST. See above AR.PREV.LIST for the argument values.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	90DAY+120DAY
SUBR("DICT.ENT.AR.AMTS","5") + SUBR("DICT.ENT.AR.AMTS","6")	

DICT.ENT.BTST.NAME

SUBROUTINE (NAMES,ATTR)

This subroutine can be used from any file and will return the Bill to and or ship to name and address if the attribute number for the internal id of the customer is passed in.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	BILLTO_NAME
SUBR('DICT.ENT.BTST.NAME',10)	

\$1012115.004 Frank & Joe Plumbing & Heating
135 SANDY AVENUE
NEW ROCHELLE NY 10800

DICT.ENT.CRLIM.PERC

SUBROUTINE (VAL,VAL.NO)

This subroutine will advise the percentage of the credit limit that is being used. The value number that needs to be used is 8. This subroutine will only work from the Entity file.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	CRLIM.PERC
SUBR("DICT.ENT.CRLIM.PERC","8")	

DICT.ENT.GET.TZ

This subroutine can only be used from the ENTITY file and will return the time zone that the customer belongs to.

DICT.ENT.INV.CT

SUBROUTINE (INV.CT,NOX)

 This routine will return the invoice count. If NOX is specified then it will not include cancelled invoices.

* INV.CT - Value returned which is the invoice count. (OUT)

* NOX - If null it will include cancelled invoices. (IN)

NOX means no x or no cancelled invoice. By indicating a "1" you are telling the system that you do not want to see cancelled invoices. This subroutine can only be run from the entity file.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	INV.CT
SUBR("DICT.ENT.INV.CT", ' ')	

DICT.ENT.PAY

Will return the payment amount and can only be used from the Entity file. Need to pass in the Beginning date and an End As of Date.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	PAYABLES
SUBR('DICT.ENT.PAY')	

DICT.ENT.PN.XREF.COMP

SUBROUTINE (VAL,REF)

This subroutine will display component information of a Kit item that is assigned to the Customer Part Number screen in Customer File Maintenance.

CN = FIELD(@ID,"~",1)

XPN = FIELD(@ID,"~",2)

LOC = FIELD(@ID,"~",3)

PN = FIELD(@ID,"~",4)

VAL = "

REF is either 1 – 4. If a product is on the Customer Part number screen with a customer part number, this routine will return the (1) Component Quantity (2) Component Customer Part number (3) Component Description (4) Component Comments

I-Descriptor Program Maintenance	
File Name :	ENTITY.PN.IDS
Dict ID :	COMP.DESC
SUBR("DICT.ENT.PN.XREF.COMP",3)	

DICT.ENT.PURCH

This routine can only work from the ENTITY file and will return the dollar amount purchased from a vendor during a time frame specified.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	PURCH
SUBR("DICT.ENT.PURCH")	

DICT.ENT.SLS

SUBROUTINE (AMT,STD,EDD,AVN)

 This subroutine Gets all the sales dollars for a particular customer for the active branch between the supplied start and end dates. The active branches come from BR\$ which is part of DICT.COMMON.

If BR\$ is not set, all authorized branches are used and BR\$ is set to all authorized branches.

If information for all authorized branches is required, use DICT.ENT.ALL.SLS...

AMT - Total amount returned..	(OUT)
STD - Start date to get the amounts for...	(IN)
EDD - End date to get the amounts for.	(IN)
AVN - What dollar amount you want returned.	[IN]
AVN = 1 : Sales Dollars are returned.	
AVN = 2 : GP Dollars	
AVN = 3 : GP%	
AVN = 4 : COGS Dollars	

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	2MTH.AGO.COGS
SUBR('DICT.ENT.SLS','FM-3/26/YR','FM-2/25/YR',4)	
-	

DICT.ENT.VAR.AR.AMTS

SUBROUTINE (VAL,SDAYS,EDAYS,EXC.DEP)

-----*

This routine is used to return a balance for a specified date range.
It uses Br/Tr/All and End/As Of Date

-----*

VAL - The AR balance returned (OUT)
SDAYS - The number of days back to start the date range [IN]
EDAYS - The number of days back to end the date range (IN)
EXC.DEP - Exclude deposits "" or 0 = NO 1 = yes exclude deposits (IN)

DICT.ENT.VC

This routine is used only in the ENTITY file and will return a "V" if Vendor or a "C" if customer.

DICT.GET.AVAIL.BR

This routine will work from the PRODUCT file and will return the products available quantity based upon the branch provided.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVAIL.BR
SUBR("DICT.GET.AVAIL.BR")	

DICT.GET.AVAIL.CREDIT

SUBROUTINE (CRAVAIL,CN)

This routine will calculate the amount of Credit that the given customer (CN) has available. Also returns the customers past due and other amounts due, as well as their credit limit. This routine calls in GET.AVAIL.CREDIT

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	AVAIL_CREDIT
SUBR('DICT.GET.AVAIL.CREDIT',@ID)	

DICT.GET.BASE

SUBROUTINE (BASE,TYPE)

Used in the PRODUCT file. Must enter the basis name so that the correct base amount is returned. Returns base amount for a product.

BASE - base amount of type is null, or base/per if TYPE is 1 [OUT]
 TYPE - flag to determine BASE above (IN)

PN = @ID
 BRS = DICT.COMMON<1>
 DT = DICT.COMMON<4>
 BASIS = DICT.COMMON<7>

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	BASIS
SUBR("DICT.GET.BASE", "")	

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	REP-COST
SUBR('DICT.GET.BASIS', "REP-COST")	

DICT.GET.BASE.BR

SUBROUTINE (VAL)

This routine is used from the PRODUCT file. This routine is smart enough to get the base amount per branch if using branch specific price sheets. The name of the local basis value comes from the dictionary common data.

PN = @ID
 BRS = DICT.COMMON<1>
 DT = DICT.COMMON<4>
 BASIS = DICT.COMMON<7>
 VAL = "
 IF DT = " THEN DT=DATE()
 BRN = DCOUNT(BRS,VM)

DICT.GET.BASE2

SUBROUTINE (BASE,PN,BR,DT)

This routine can be run from a file where the internal id of the product is available to pass in.
This subroutine is smart enough to pull the sheet that is in effect as of the date specified.

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	BASIS
SUBR("DICT.GET.BASE2",FIELD(@ID,'~',1),FIELD(@ID,'~',2),FIELD(@ID,'~',3))	

I-Descriptor Program Maintenance	
File Name :	MATRIX
Dict ID :	BASIS
FIELD(@ID,"~",3); IF NUM(@1) THEN SUBR("DICT.GET.BASE2",@1) ELSE ""	

DICT.GET.BASE3

SUBROUTINE (BASE,PN,BR)

This routine will pull the current price sheet. This routine returns the Price Base based on PN, BR and as of date which passed by common variable DICT.COMMON<4>.

* BASE - BASE Amount [OUT]
* PN - Product ID [IN]
* BR - Branch ID [IN]

I-Descriptor Program Maintenance		View Only
File Name :	MATRIX	
Dict ID :	BASIS.ASOF	
FIELD(@ID,"~",3); IF NUM(@1) THEN SUBR("DICT.GET.BASE3",@1) ELSE ""		

DICT.GET.BASIS

Used only from the PRODUCT file.

This will get the basis value using the Global Basis. Therefore this one is smart enough to map the global basis to the local basis it is pointing to in price line maintenance to return the correct base amount.

SUBROUTINE (PBASE,GBASIS)

 This routine is called by I-descriptors in Eclipse Dictionaries to get the local price base

PBASE - The price base corresponding to the Global Basis OUT
 GBASIS - Global Basis you want the price base for [IN]

I-Descriptor Program Maintenance		View Only
File Name :	PRODUCT	
Dict ID :	PURC-BREAK	
SUBR("DICT.GET.BASIS", "PURC-BREAK")		

DICT.GET.BR.COST

SUBROUTINE (VAL,CST.BN)

This routine will work from the PRODUCT file and calls in another subroutine. You must call in the cost basis name that you are looking to have returned.

 This routine is used to return the COST (arg) for the given Cost Basis requested by the user via the cost basis #. The Product Log will be searched for the price for the given As Of Date /As Of Time and assign the cost <= to that date. If there is no pricing information for the product found in the product log, it will just return the current cost by calling GET.BASE.

NOTE:

Cost Basis Number Association to PROD.LOG

COST.BN = BASIS

8	= 1	= Average Cost
9	= 2	= Last Cost
22	= 3	= Average Landed Cost
21	= 4	= Landed Cost
23	= 5	= Frozen Average Cost
24	= 6	= Frozen Last Cost
26	= 7	= Frozen Average Landed Cost
27	= 8	= Frozen Landed Cost



DICT.GET.BR.NAME

SUBROUTINE (BR.NAME,BR)

You must provide the branch number. If a file has an attribute that is storing the branch the attribute will be passed in. That will then return the name of the branch.

I-Descriptor Program Maintenance	
File Name :	QUAL.LOG
Dict ID :	BR.NAME
SUBR("DICT.GET.BR.NAME",BR)	
-	

`DICT.GET.BYLINE.VALUE`

SUBROUTINE (VAL,ATTB)

This will return branch specific buy line data. Specify the attribute number from the BUY.LINE.BR file that needs to be passed back.

I-Descriptor Program Maintenance	
File Name :	BUY.LINE
Dict ID :	GRACE
SUBR('DICT.GET.BYLINE.VALUE',8)	

DICT.GET.CLASS

SUBROUTINE (PRICES,QSIGN,PRC.UOM)

This subroutine returns the price of a product for a given date, branch, Qsign and price class.. If more than one branch is specified in the VM delimited list of BRS from DICT.COMMON<1>, a corresponding list of prices is returned..

If the PRC.UOM flag is not passed in, the price is returned per 1 of the lowest UOM.

PRICES - List of Prices for the Product. (OUT)
 QSIGN - Type of price requested...(-1) - Sell price [IN]
 (1) - Buy price
 PRC.UOM - Flag to return the price for the current pricing (IN)
 unit of measure..

COMMONS:

@ID - Id of the current product record. (READ)
DICT.COMMON - Where the Branches, date and Price class (READ)
are stored.
JUST ABOUT EVERY ARRAY (MODIFIED)

```
PRICES  = "  
PN      = @ID  
BRS     = DICT.COMMON<1>  
PRC.DATE = DICT.COMMON<4>
```


CLASS = DICT.COMMON<5>

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	CLASS.SELL.PRICE
SUBR('DICT.GET.CLASS',' -1','')	

DICT.GET.CONTACT.ATTB

SUBROUTINE (VAL,CN.IDS,ATTR1,ATTR2,MVAL,SVAL)

 This routine is excellent for a custom file that displays contacts from the contact file. This will get two attributes from the Contact File and will concatenate them together (ie. First Last Name). This will work for multiple contact ids passed in a multi-valued list.

VAL - Attributes concatenated together from contact file [OUT]
 CN.IDS – Multi-valued list of contact ids [IN]
 ATTR1 - First contract Attribute to be concatenated [IN]
 ATTR2 - Second contract Attribute to be concatenated [IN]
 MVAL - Multi Value position
 SVAL - Sub-value position

DICT.GET.CUS.LSHP.DT

SUBROUTINE (VAL,CN,PN)

Dictionary subroutine that will allow you to pass in a customer ID (Eclipse Internal ID) and Eclipse internal PN and it will pass back the last ship date (VAL)

I-Descriptor Program Maintenance	
File Name :	ENTITY.PN.IDS
Dict ID :	LAST_SHP_DT
SUBR('DICT.GET.CUS.LSHP.DT',FIELD(@ID,'~',1),FIELD(@ID,'~',4))	

DICT.GET.CUST.PN

This routine works from the ENTITY.PN.IDS file return the customer part number, product description and internal eclipse product id when running for a particular customer.

```
SUBROUTINE (DISPS,VAL)
*****
* VAL = 1 Customer Part Number  *
* VAL = 2 Product Description   *
* VAL = 3 Product @ID          *
*****
CN = FIELD(@ID,'~',1)
```

DICT.GET.CUST.PN.PRD

SUBROUTINE (VALS,OPT,PN)

This subroutine will need to have the option and the Product ID number passed in. It can be used from the PRODUCT file to return customer part numbers.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	CUST.PN
SUBR("DICT.GET.CUST.PN.PRD",1,@ID)	

DICT.GET.CUST.PN.PSUB

This routine works from the PSUB file and will return the customer's part number or the product description, which ever value is passed in to the subroutine.

SUBROUTINE (DISPS,VAL)

```
* VAL = 1 Customer Part Number  *
* VAL = 2 Product Description   *
```

I-Descriptor Program Maintenance	
File Name :	PSUB
Dict ID :	CUST.PN
SUBR('DICT.GET.CUST.PN.PSUB','1')	

DICT.GET.CUSTOMER.CREDIT

SUBROUTINE (VAL,TYPE)

This routine can only be used from the ENTITY file and needs to have the type of credit limit you are looking to pass back. Valid types are:

```
PAST.DUE
TOTAL.AR
JOB.TOT
CREDIT.LIMIT
PAST.DUE.LIMIT
```

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	CR_LMT_JOB_TOTAL
SUBR('DICT.GET.CUSTOMER.CREDIT' , 'JOB.TOT')	

DICT.GET.DISCOUNT

SUBROUTINE (DISC.AMT)

 Returns true discount amount from AR record by totaling all of the discounts on this AR ID.
 This routine can only be used from AR file.

DISC.AMT - Total Discount Amount Returned (OUT)

 * NOTE: @RECORD is assumed to be the AR file.

I-Descriptor Program Maintenance View Only	
File Name :	AR
Dict ID :	DISC.TAKEN
SUBR('DICT.GET.DISCOUNT')	

DICT.GET.ENT.CONTACTS

SUBROUTINE (VAL,CN,PASSER)

Because you are passing in the CN number, this subroutine can be used from any file for which the ENTITY id can be passed in. You must fill in the “passer” which is listed below as to the type of information from the contact file you want to return.

-----*
 This subroutine returns a multi-valued list of contact information from the contact file for a given entity. The information returned is based upon the value passed into the routine in the PASSER argument. For example, if PASSER<1> is 'ID', then the return list contains a list of contact IDS for the entity.

This routine is currently only called from CONTACT_LIST I-Descriptor in ENTITY.

-----*
 VAL - Multi value list of information specified in PASSER [OUT]
 CN - Entity ID [IN]
 PASSER - Specifies data to return as follows: [IN]
 Numeric or comma-delimited numeric - Attribute, Value, Sub-value to retrieve
 'ID' - Returns list of contact IDS for this Entity
 'NAME' - Returns list of full contact names (first middle last)
 'LNAME' - Returns list of full contact names (last, first middle)
 'PHONE#' - Returns list of phone number defined where # is
 (ex: PHONE1 returns list of first number of every contact)
 'PDESC#' - Returns list of phone descriptions at # specified
 'PTYPE=TYPE' - Returns phone number that corresponds to TYPE where TYPE defines what type we're looking for.

I-Descriptor Program Maintenance		View Only
File Name :	ENTITY	
Dict ID :	CONTACT_LIST	
SUBR('DICT.GET.ENT.CONTACTS',@ID,'ID')		

DICT.GET.ENTITY.ATTR

SUBROUTINE (VAL,CN,ATTR,MVAL,SVAL)

If you have then entity id, this subroutine can be used from any file. You can specify the attribute number, multi-value and sub-value position to pull back.

I-Descriptor Program Maintenance		View Only
File Name :	MATRIX	
Dict ID :	CUST.INSIDE	
SUBR("DICT.GET.ENTITY.ATTR",CN,44,"","")		

DICT.GET.ENTITY.BR.VAL

SUBROUTINE (VAL,ATTR,USE.CN)

This routine will return the branch specific value for the attribute passed in from the ENTITY.BR file. If branch is "ALL", we will retrieve the value from the ENTITY file. The attributes are defined control records, CUS.BR.CATEGORY and EN.BR.CATEGORY, depending upon entity type.

- * VAL - Value passed out stored in record [OUT]
- * ATTR - Attribute to retrieve value for in ENTITY.BR [IN]
- * USE.CN - Customer to get data for (blank = @ID) (IN)

DICT.GET.FIRST.SOLD

SUBROUTINE (FIRST.VAL,OPT,PN)

Option is either "" or "1". "" will return the first purchase cost and "1" will return the first purchase date. This subroutine can be used on any file where you can specify the internal id number of the product.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	FIRST.COST
SUBR('DICT.GET.FIRST.SOLD','2',@ID)	

DICT.GET.ID.LIST

SUBROUTINE (ID.LIST,FILENAME,ID,ATTR,CONV)

Returns a list of values separated with a value marker.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	EXP.NAME
SUBR('DICT.GET.ID.LIST','ENTITY',@ID,'56','G1~1');	
TRANS("GENLED",@1,"3","X")	

DICT.GET.LAST.PURCH

SUBROUTINE (VAL,OPT,PN,BR,PRC.DATE)

IF OPT = 1 returns LAST.DATE else returns LAST.COST

This routine will return a null LAST.DATE if there is no last P/O

Found...

I-Descriptor Program Maintenance	
File Name :	PROD.DYNAM
Dict ID :	LAST.PURCH.DATE
SUBR('DICT.GET.LAST.PURCH',1,@ID,@VM,'')	

DICT.GET.LAST.REC

SUBROUTINE (LAST.VAL,OPT,PN)

This routine can work from any file where you have the internal id number of the product.

Opt 1 = LAST.DATE

Opt 2 = LAST.PURCH

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LAST.PO.DATE
SUBR("DICT.GET.LAST.REC","1",@ID)	

DICT.GET.LASTDATE

SUBROUTINE (LPDATE)

Can only be run from the ENTITY file and will return the last transaction date for the entity.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	LAST.PAY.DT
SUBR('DICT.GET.LASTDATE')	

DICT.GET.LASTPAY

SUBROUTINE (LPAMT)

Can only be run from the ENTITY file and will return the last payment amount for the entity.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	LAST.PAY.AMT
SUBR('DICT.GET.LASTPAY')	

DICT.GET.LED.LOG.VALUE

SUBROUTINE (VAL,ATTB)

OID = FIELD(@ID,',',1)

GN = DCOUNT(@ID,',')

This subroutine requires the attribute number from the LEDGER.LOG file you want to pull back.

This subroutine works from the AR or the ORDER.QUEUE file.

I-Descriptor Program Maintenance		View Only
File Name :	PSUB	
Dict ID :	MANIFEST	
SUBR('DICT.GET.LED.LOG.VALUE' ,17)		

DICT.GET.LEDGER

SUBROUTINE (VAL,OID,GEN,ATTB)

This routine is used from the AR file to return data from the LEDGER file. The order id and generation need to be passed in as well as the attribute number from the ledger file that you want to return. There are some other ATTB types that can be entered in lieu of an attribute number to pull back line item information.

LI.SQTY = Line item ship quantity
 LI.OQTY = Line item order quantity
 LI.DESC = Line item description
 LI.CT = the total number of line items on the generation for the AR record
 LI.PRCLN = line item price line
 LI.PRCEXT = Line item ship quantity extended price.
 LI.UNIT = Line item unit price
 LI.CSTEXT = Line item ship quantity extended cost of goods sold cost
 LI.COMEXT = Line item ship quantity extended commission cost.
 LI.TYPE = line item product type
 LI.PN = Line item internal id number
 LI.DMD = Line item monthly demand
 LI.ONHAND = Line item on-hand value

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	PROD.DESCS
SUBR('DICT.GET.LEDGER.VALUE' , 'LI.DESC')	

DICT.GET.LEDGER.AMT

SUBROUTINE (AMT,SP.ID,MULT)

Standard posting ids from the ledger file. This can work from the AR

Pass in the Standard posting id and the dollar value posted will appear. This will also want to know what multi-value you want.

I-Descriptor Program Maintenance	
File Name :	AR
Dict ID :	TOTAL.AMT
SUBR('DICT.GET.LEDGER.AMT' , 'AR' ,1)	

DICT.GET.LEDGER.AMT.PQ

SUBROUTINE (AMT,SP.ID)

This routine will pull back all of the postings to a standard posting you identify. This subroutine works from the PRINT.QUEUE file.

I-Descriptor Program Maintenance	
File Name :	PRINT.QUEUE
Dict ID :	TAX.AMT
SUBR('DICT.GET.LEDGER.AMT.PQ' , 'TAX')	

DICT.GET.LEDGER.BASE

SUBROUTINE (VAL)

-----*

This is a dictionary routine that is designed to find the value of a particular basis from the price sheet based on the branch, date, and basis that is passed in through a common dictionary array for each item found for specific orders within the ORDER.QUEUE file

-----*

```

BRS  = DICT.COMMON<1>
DT   = DICT.COMMON<4>
BASIS = DICT.COMMON<7>
IF DT = " THEN DT=DATE()

```

DICT.GET.LEDGER.DET.VALUE

SUBROUTINE (VAL,OID,LDID,GID,ATTB)

Get a ledger detail value from the LEDGER.DET file. If Generation ID (GID) is null it will just get the entire value (attribute number)... GID can be the INVN (LENGTH = 3) or the Generation ID (LENGTH = 4)

This subroutine can work from the ORDER.QUEUE, and the PSUB file.

I-Descriptor Program Maintenance		View Only
File Name :	PSUB	
Dict ID :	REASON.CODE	
SUBR('DICT.GET.LEDGER.DET.VALUE',FIELD(@ID,'~',4),FIELD(@ID,'~',6),FIELD(@ID,' ',5)"R%3",'46')		

DICT.GET.LEDGER.FGHT

SUBROUTINE (VAL,OID,INVN,ATTB)

Returns the freight associated with the invoice.

Must pass in the attribute. (36)

I-Descriptor Program Maintenance		View Only
File Name :	PSUB	
Dict ID :	BILL.OUT.FGHT	
SUBR("DICT.GET.LEDGER.FGHT",OID,INVN "R%3",4)		

DICT.GET.LEDGER.LOG

SUBROUTINE (VAL,ATTB)

This will return information from the LEDGER.LOG file and can be used from the AR and the ORDER.QUEUE file

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	PRT
SUBR('DICT.GET.LEDGER.LOG' ,9)	

DICT.GET.LEDGER.VALUE

SUBROUTINE (VAL,ATTB)

This dictionary works from the PSUB, LEDGER, AR or the ORDER.QUEUE file. Specify the attribute number from the LEDGER file that you want to pass back. The attribute can also be a code to pull back different information.

- LI.PN will pull back the Line Item Part number.
- LI.SQTY will pull back the line item ship quantity
- LI.SQTY.PER will pull back the uom for the ship quantity.
- LI.OQTY pulls back the line item open quantity
- LI.OQTY.PER pulls back the uom for the open quantity.
- LI.DESC pulls back the description of the line item.
- LI.ENT# pulls back the customer part number (if customers are assigned part numbers in the customer/vendor part number screen)
- LI.PRCLN pulls back the price line for the item on the order.
- LI.PRCEXT pulls back the extended price for the item on the order
- LI.UNIT pulls back the unit price for the item on the order
- LI.CSTEXT pulls back the extended cost of the item
- LI.COMEXT pulls back the unit cost of the item
- LI.TYPE pulls back the sales quantity type such as tag, defective, etc.
- LI.ONHAND pulls back the product's on-hand quantity.
- LI.ONHAND.PER will display the unit of measure for the on-hand quantity.
- LI.PN.STK.FLAG displays the branches stk flag from the Primary inventory maintenance screen. (- = ""; 1 = Y ; 0 = N)
- LI.PN.BUY.ID displays the user id of the buyer for the items on the transaction
- LI.ONPO displays quantity on a purchase order for the line item..
- LI.ONPO.DT displays the expected receiving date of the purchase order
- LI.ONPO.PER displays the unit of measure for the quantity on order from the po.
- LI.ON.XFER displays the quantity on a transfer.
- LI.ON.XFER.PER displays the unit of measure for the quantity on transfer
- LI.TAGDT displays the date the tagged purchase order or transfer is expected to be received in.
- TV.NAME pulls back the full name of the writer of the transaction
- TV.BFLW.DT pulls back the bid follow-up date for the transaction.

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	GROUP
SUBR('DICT.GET.LEDGER.VALUE',51)	

DICT.GET.LEDGER.VALUE.PQ

Similar to the previous subroutine, this one can only be used from the PRINT.QUEUE file. Enter the attribute number from the LEDGER file you wish to pull back or use one of the following codes instead:

- LI.SQTY = line item sales quantity
- LI.OQTY = line item open quantity
- LI.DESC = line item description
- LI.PRCLN = line item price line
- LI.PRCEXT = line item extended price
- LI.CSTEXT = line item extended cost of good sold cost
- LI.COMEXT = line item extended commission cost
- LI.TYPE = line item sales quantity type
- LI.PN = line item part number
- LI.DMD = line item monthly demand
- LI.ONHAND = line item on-hand
- TV.NAME = Full name of the writer
- TV.BFLW.DT = bid follow-up date

DICT.GET.MATRIX.DATES.VAL

SUBROUTINE (VAL,ATTR)

This dictionary can only be used from the Matrix Dates file. It returns an attribute number you specify.

I-Descriptor Program Maintenance	
File Name :	MATRIX
Dict ID :	ORIG.EXP.QTY
SUBR("DICT.GET.MATRIX.DATES.VAL",5)	

DICT.GET.MATRIX.VAL

SUBROUTINE (VAL,ATTR,MVAL)

This subroutine is used in the matrix file and will return the attribute number and multi-valued position you specify.

In the ATTR argument you can also pass in the following codes:

- ALPHA
- SLSM
- SLSM.IN
- NAME
- INDEX
- MISC.WORK
- MISC.WORK,CUST.NAME

- DATES
- TYPE
- ABBR
- BRANCH

I-Descriptor Program Maintenance	
File Name :	MATRIX
Dict ID :	QUOTE.TITLE
SUBR('DICT.GET.MATRIX.VAL','MISC.WORK',1)	

DICT.GET.MISC.DATA.VAL

SUBROUTINE (VAL,KEY,ATTR,MVAL,SVAL)

This subroutine will pull back the attribute, multi-value and sub-value from the MISC.DATA file. You are required to specify the key to the record for which you need to extract data.

I-Descriptor Program Maintenance		View Only
File Name :	MATRIX	
Dict ID :	QUOTE.CUST.NO	
SUBR("DICT.GET.MISC.DATA.VAL",QUOTE.KEY,2,"","")		

DICT.GET.ML.CUR.VAL

SUBROUTINE (VAL,REC.ID)

Returns the current MAINT.LOG value

I-Descriptor Program Maintenance		View Only
File Name :	MAINT.LOG	
Dict ID :	CURRENT.VAL	
SUBR("DICT.GET.ML.CUR.VAL",@ID)		

DICT.GET.NO.LOCS

SUBROUTINE (VALS)

This subroutine works from the PRODUCT, or PRODUCT.NOTES file and will display products that do not have a bin location in a branch. The prompt hotkey in dictionary maintenance must prompt for a branch.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	NO.LOCS
SUBR("DICT.GET.NO.LOCS")	

DICT.GET.PRD.XFER.HITS

SUBROUTINE (VAL, PN, OPT)

This is a custom routine which will loop through all of the brs for a start and end date and add up the number of hits for xfers... This subroutine needs to have the internal ID of the Product.

Therefore it can only be run from the PRODUCT, PRODUCT.NOTES and PROD.DYNAM files where the key is the Product's internal ID number. You can fill in either option 1 or 2.

Option 1 = Hits and Option 2 = Qty

VAL = "

BRCHS = DICT.COMMON<1>

ST.DT = DICT.COMMON<3>+0

END.DT = DICT.COMMON<4>

I-Descriptor Program Maintenance		View Only
File Name :	PRODUCT	
Dict ID :	XFER.HITS	
SUBR('DICT.GET.PRD.XFER.HITS', @ID, '1')		

DICT.GET.PREV.CNT

This subroutine is run from the product file and will return the previous on-hand

I-Descriptor Program Maintenance		
File Name :	PRODUCT	
Dict ID :	ONHAND.BR	
SUBR("DICT.GET.PREV.CNT")		

DICT.GET.PREV.ONHAND

This subroutine will return the on-hand value for a product for a given date. This subroutine calls in another subroutine called GET.PREV.ONHANDS. This routine works from the Product File, Prod.dynam file as well as the product.notes file.

The prompts need to be as of date and Branch/Tr/All

I-Descriptor Program Maintenance		
File Name :	PRODUCT	
Dict ID :	PREV.ONHAND	
SUBR("DICT.GET.PREV.ONHAND")		

DICT.GET.PREV.ONHAND.TYP

SUBROUTINE (TOT.QOH,QTYPES,NEG,INPROCESS)

* Subroutine: DICT.GET.PREV.ONHAND.TYP

When using this subroutine you need to supply what types of location (listed below), whether to include negative quantities as well as in-process quantities.

 * Calc ONHAND for branches & as of date in DICT.COMMON

* TOT.QOH - Total quantity on-hand for location types

* QTYPES - Types of location

* NEG - Include negative quantities

* INPROCESS - Include in-process quantities

* QTYPES:

* S - Stock

* T - Tagged

* F - DeFective

* R - Review

* O - Overstock

* L - DispLay

* C - Consignment

* NEG & INPROCESS:

* O - Only

* E - Exclude

- null for include

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	OH.TYPE
SUBR("DICT.GET.PREV.ONHAND.TYP", ' ')	

DICT.GET.PREV.ONHANDS

SUBROUTINE (OH,PN)

This routine can be called from anywhere rather than just the PRODUCT file to get previous on-hands - Pass the eclipse PN in the PN field. It will get the branches and effective dates from common and pass back the on-hand in OH variable...

DICT.GET.PRNK

SUBROUTINE (RANK,PN,BR)

This routine can be used to pull back the Rank of an item in a branch. The product's internal ID needs to be passed in, so this can be used from any file where you have this information to pass into the subroutine.

You also need to pass in the Branch number you are looking to retrieve the rank for, otherwise the system will pull back branch 1's rank.

I-Descriptor Program Maintenance	
File Name :	QUAL.LOG
Dict ID :	PRNK
SUBR(' DICT.GET.PRNK' ,@RECORD<8> ,@RECORD<12>)	

DICT.GET.PROD.INFO

SUBROUTINE (VAL,PN,OPT)

Version# 1 - 03/24/1996 - 08:31pm - SCOTTR - **

VAL = "

BRS = DICT.COMMON<1>

BRN = DCOUNT(BRS,VM)

SD = DICT.COMMON<3>

ED = DICT.COMMON<4>

You must supply the part number in question and the option to pull back

AVEQOH = Average quantity on-hand

SLS = sales quantity

SL\$\$ = Sales dollars

COGS = Cost of goods sold

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVEQOH
SUBR(' DICT.GET.PROD.INFO' ,@ID, "AVEQOH")	

DICT.GET.PUD

Routine to get attb 5 of PU.IDX file and wrap to 35 characters

When you do an auto price update the description is written to the pu.idx file and is stored in attribute 5. This subroutine is used to display the auto price update description instead of the product's description in the product file.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	PUD
SUBR(' DICT.GET.PUD')	

DICT.GET.RANK

SUBROUTINE (RANK,CN)

This routine pulls back the customer's rank. This routine can be run from any file where you can pass in the key to the ENTITY file.

I-Descriptor Program Maintenance	
File Name :	QUAL.LOG
Dict ID :	VRNK
SUBR(' DICT.GET.RANK' ,VEND)	

DICT.GET.VAL

SUBROUTINE (ANS,FILENAME,ID,ATTR,VAL,SVAL)

This subroutine can be used from any file. Pass in the filename, the key, attribute number, value and sub-value you want to pass back.

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	CARRY.COST\$
SUBR('DICT.GET.VAL' , "LEDGER" , FIELD(ORDER# , ' ' , 1) , "87" , "1" , "")	

DICT.GLOBAL.RANK.GET

SUBROUTINE (RANK,VAL)

 This routine returns the rank at sub-value (VAL) for a given product.

VAL - Subvalue mark [IN]

RANK - Value returned [OUT]

Common Variables - None used in this subroutine.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	GLOBAL.RANK#1
SUBR('DICT.GLOBAL.RANK.GET' , '1')	

DICT.JOB.TRACKING.GET.CMT

If you are running a report from the TRACKING.LOG file, this dictionary will return the comments back on a report.

I-Descriptor Program Maintenance	
File Name :	TRACKING.LOG
Dict ID :	COMMENTS
SUBR('DICT.JOB.TRACKING.GET.CMT' , '')	

DICT.LAST.CYCLE.COUNT

```

SUBROUTINE (LAST.CYCLE.COUNT)
BR      = DICT.COMMON<10>
START.DT = DICT.COMMON<3>
END.DT   = DATE()

```

This dictionary will return the last time a product was cycle counted. The branch prompt must be the Enter Br: not the Branch/Tr/All prompt to work.

I-Descriptor Program Maintenance — View Only	
File Name :	PRODUCT
Dict ID :	LAST.CYCLE.COUNT
SUBR("DICT.LAST.CYCLE.COUNT")	

DICT.LED.LF.GET

I believe this counts the number of products on a transaction when you are writing a report off of the ORDER.QUEUE file.

I-Descriptor Program Maintenance — View Only	
File Name :	ORDER.QUEUE
Dict ID :	ORDER.LF
OCONV(SUBR('DICT.LED.LF.GET',@ID),'MR04')	

DICT.LED.WEIGHT.GET

I believe this retrieves the Weight of a transaction from the LEDGER file when you are writing a dictionary from the ORDER.QUEUE file.

I-Descriptor Program Maintenance — View Only	
File Name :	ORDER.QUEUE
Dict ID :	ORDER.WGT
SUBR('DICT.LED.WEIGHT.GET',@ID)	

DICT.LEDL.INFO

```

SUBROUTINE (VAL,ID,WORD)

```

Set WORD to Printed or Overcommit etc

This subroutine works from files that have the Order number somewhere in the @ID. Provide the subroutine with a word from the change log of the transaction and it will return the date, time, user who did the activity.

I-Descriptor Program Maintenance — View Only	
File Name :	ORDER.QUEUE
Dict ID :	BID.CHANGE
SUBR("DICT.LEDL.INFO",@ID,"** Bid")	

DICT.MATRIX.PROD.DESC

If the matrix cell is product specific, this subroutine will pull back the description of the product from the product file.

I-Descriptor Program Maintenance	
File Name : MATRIX	
Dict ID : PROD.DESC	
SUBR('DICT.MATRIX.PROD.DESC')	

DICT.MATRIX.PROD.LINE

This subroutine will pull back the product's price line.

I-Descriptor Program Maintenance	
File Name : MATRIX	View Only
Dict ID : PROD.LINE	
SUBR("DICT.MATRIX.PROD.LINE")	

DICT.MV.CONV

SUBROUTINE (VAL,ATTR)

-----*

- This routine will convert VM to spaces in dictionary items. Currently being used in SQL data warehouse product

DICT.ORD.PT.GET

This routine is used in the PRODUCT file or any file where the @id is the PN. Subroutine to retrieve the order point value for a product

```

SUBROUTINE (OPS)
IS.WHSE = 1
BRS    = DICT.COMMON<1>
OPS = "
MATREAD PRD FROM PRDFILE,@ID ELSE MAT PRD="
PN = @ID

```

I-Descriptor Program Maintenance	
File Name : PRODUCT	
Dict ID : OP.BR	
SUBR('DICT.ORD.PT.GET')	

DICT.ORD.QUE.CUSNAME

This subroutine will pull back the customer's name when used on the ORDER.QUEUE file.

I-Descriptor Program Maintenance	
File Name :	ORDER.QUEUE
Dict ID :	CUST.NAME
SUBR("DICT.ORD.QUE.CUSNAME")	

DICT.PAY.AMT.GET

SUBROUTINE (AMT)

BEG.DT = DICT.COMMON<3>

ASOF = DICT.COMMON<4>

This subroutine works from the ENTITY file and will show the amount the customer owes during the start and end date specified.

I-Descriptor Program Maintenance	
File Name :	ENTITY
Dict ID :	VEND.PAY.AMT
SUBR("DICT.PAY.AMT.GET")	

DICT.PQ.HNDL.FGHT

SUBROUTINE (AMT,TYPE)

 * This routine retrieves the various Freight and Handling amounts for and order in PRINT.QUEUE

* AMT (OUT) - The Freight or Handling amount specified by TYPE
 * TYPE (IN) - A string telling this routine what type of total to
 * return.

* COMMON VARIABLES

* @ID is used but not changed

The different types you can pull back are the following:

FGHT.IN.BILL

FGHT.OUT.BILL

FGHT.IN.EXPENSE

FGHT.OUT.EXPENSE

HNDL.IN.BILL

HNDL.OUT.BILL

HNDL.IN.EXPENSE

HNDL.OUT.EXPENSE

I-Descriptor Program Maintenance		View Only
File Name :	PRINT.QUEUE	
Dict ID :	HNDL.IN.BILL	
SUBR('DICT.PQ.HNDL.FGHT','HNDL.IN.BILL')		

DICT.PRD.ATTR

Retrieves the attributes from the PROD.BR file.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	HITS
SUBR('DICT.PRD.ATTR',30)	

DICT.PRD.AVG.PRC

SUBROUTINE (AVG.PRC)

*** SUBROUTINE - DICT.PRD.AVR.PRC

*** This routine calculates the Average Selling Price for the Product whose PN is @ID based on the Branches, Start Date and End Date specified through SET.COMMON at TCL.

*** AVG.PRC (OUT) - The Average Selling Prices (VM by Branch)

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVG_PRICE
SUBR('DICT.PRD.AVG.PRC')	

DICT.PRD.AVGSALE

This subroutine will pull back the average sale quantity from the branch specific data.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	AVG_SALE_QTY
SUBR("DICT.PRD.AVGSALE")	

DICT.PRD.BABY.SURPLUS.PT

Subroutine: DICT.PRD.BABY.SURPLUS.PT

-----*

This routine is an I-descriptor for the PRODUCT FILE to calculate the surplus point for a baby branch. It uses the mother branches line point days * the baby branches demand/day + the baby branches economic order quantity. This is very similar to DICT.PRD.SURPLUS.PT except for the calculation on the baby branches, they use the baby's line point day.

DICT.PR.D.CMP.COST

SUBROUTINE (REPLCOST)

Routine to display kit item component costs

Variables:

REPLCOST - Individual component cost (Out)

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	CMP.COST
SUBR("DICT.PR.D.CMP.COST")	

DICT.PR.D.CMP.DESC

This subroutine will display the description of the components of a kit item.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	CMP.DESC
SUBR("DICT.PR.D.CMP.DESC")	

DICT.PR.D.CMP.LIST

This routine will display the list price of the components of a kit item.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	CMP.LIST
SUBR('DICT.PR.D.CMP.LIST')	

DICT.PR.D.COSTS

SUBROUTINE (COSTS,ATTR)

Subroutine: DICT.PR.D.COSTS

Program is passed a value and pulls this attribute position from the PROD.CALC.BR file which contains all calculated product information and then returns the result.

COSTS - Result from what is stored PROD.CALC.BR at position ATTR [OUT]
ATTR - Attribute Position [IN]

I-Descriptor Program Maintenance View Only	
File Name :	PRODUCT
Dict ID :	LAND.AVG.COST
SUBR('DICT.PR.D.COSTS','20')	

DICT.PR.D.CUS.ORDERS

SUBROUTINE (ORDATA,VAL,PN,CN)

* VAL = 1 ; Open Order ID's *

* VAL = 2 ; Required Date *

* VAL = 3 ; Qty pending shipment *

* VAL = 4 ; First line of product desc *

* VAL = 5 ; Ship Date *

If you can pass in the @Id of the product file and the @id of the customer, you can return any of the above value numbers.

I-Descriptor Program Maintenance	
File Name :	ENTITY.PN.IDS
Dict ID :	OPEN.ORDERS
SUBR('DICT.PR.D.CUS.ORDERS',1,PN,EN.ID)	

DICT.PR.D.CUS.SALES

SUBROUTINE (QTY,RANGE,ATTR,SD,ED,PN,CN)

-----ATTR Definition

3 = Sales

4 = Purchases

5 = Transfers

6 = Sales \$

7 = Purch \$

8 = Xfer \$

9 = Sales COGS \$

10 = Sales GP \$

11 = Sales GP %

-----Range Definition

1 = User Defined

2 = MTD

3 = YTD

4 = Fiscal MTD

5 = Fiscal YTD

I-Descriptor Program Maintenance	
File Name :	ENTITY.PN.IDS
Dict ID :	SALES
SUBR("DICT.PR.D.CUS.SALES",1,3,' ',' ',PN,EN.ID)	

DICT.PRD.CUS.SLS

SUBROUTINE (QTY,PN,CN,MODE,ATTR)

BR = DICT.COMMON<1>

SD = DICT.COMMON<3>

ED = DICT.COMMON<4>

This routine is used to get the number of sales of a specific PN during a given period. By sending in a CN, the programmer is saying they want to know what the sales data for the PN were for that CN. If there is no CN, the programmer just wants to know in general how many sales were made for that PN in general. There is also new functionality so that if you send this routine null BRS, then it assumes you want to know what happened across all pricing branches.

The ATTR parameter informs the subroutine about what aspect of the sales data you are looking for... The table is as follows:

THIS ROUTINE RETURNS SALES DATA FOR PRICE BRANCH, NOT SHIP BRANCH

ATTR Definition

2 = Transaction Count

3 = Sales

4 = Purchases

5 = Transfers

6 = Sales \$

7 = Purch \$

8 = Xfer \$

9 = Sales COGS \$

10 = Sales GP \$

11 = Sales GP %

PN - The PN that we are interested in... (IN)

BRS - The Pricing branches we are interested in... If
null, then want data for ALL pricing branches. (IN)

RANGE - Ummm... I'll have to get back to you on that.. (IN)

ATTR - The integer trigger that let's the routine know
which specific totals you're interested in. (IN)

SD - The start date of the date range. (IN)

ED - The end date of the date range. (May be null.) (IN)

QTY - The summation of the data that you're looking for. (OUT)

CN - The CN that you are looking for summary data for this PN. (May be null.) (IN)

I-Descriptor Program Maintenance		View Only
File Name :	PRODUCT	
Dict ID :	PO.COUNT	
SUBR('DICT.PRD.CUS.SLS',@ID,'"', 'P',2)		

DICT.PRD.EOQ

This routine only works from a file where the Product ID is the key to each record and will pull back the EOQ per branch.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	EOQ
SUBR('DICT.PRD.EOQ')	

DICT.PRD.GET.INFO

SUBROUTINE (VAL,OPT,TYPE)

Works from the PRODUCT file.

The options you can choose are:

LP = Line Point

OP = Order Point

XFER.PT = Transfer Point

DMD = Demand

The different types can be:

WBRS = Warehouse branch

PBRs = Purchasing Branch

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LN.POINT
SUBR('DICT.PRD.GET.INFO' , 'LP' , '')	

DICT.PRD.GET.PER

SUBROUTINE (PER.QTY,MODE)

PN = @ID

Subroutine: which calls in another subroutine called DFLT.PER.GET

This returns the correct UoM type and quantity for the given mode.

 MODE - UoM mode - S,P,T,A,I (IN)

DFLT.PER - UoM quantity for this mode (OUT)

DFLT.ALPHA - UoM alpha code (OUT)

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SALES.PER.UOM
SUBR('DICT.PRD.GET.PER' , 'S')	
-	

DICT.PRD.HITS

This routine will find the number of hits calculated using the PSUB file.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	HITS.CALC
SUBR('DICT.PRD.HITS')	
-	

DICT.PRD.INV.MAINT

SUBROUTINE (VALU,OPTN)

This routine calls in the DFLT.PER.GET routine for “P” type (purchasing branch information)

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	XFER.PT
SUBR('DICT.PRD.INV.MAINT', '1')	
-	

DICT.PRD.LASTACT

This subroutine will look through the PSUB file of the selling branch to display the last activity date of a product.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LASTACT
SUBR('DICT.PRD.LASTACT')	

DICT.PRD.LASTSALE

This routine will look through the PSUB file for the selling branch to display the last sales order for an item.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LASTSALE
SUBR('DICT.PRD.LASTSALE')	
-	

DICT.PRD.LASTSALE.PN

SUBROUTINE (LASTSALE,PN,CN)

This routine allows you to pass in the internal id of the product and the internal id of a customer to find the last sales transaction for the customer for the item.

DICT.PRD.LEADDAYS

This routine will obtain the product lead time, however it uses the user's home branch to give a better estimate of the lead time that is needed.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LEADDAYS
SUBR('DICT.PRD.LEADDAYS')	

DICT.PRD.LOCS

SUBROUTINE (VAL,OPT,LOCN)

** OPT = -1: All Loc info if qty # 0

** OPT = 0 ; All Location info

** OPT = 1 ; Location Types

** OPT = 2 ; Location Codes

** OPT = 3 ; Location Tag

You need to pass in the location that you want to display the information. The options above will determine the information for the location you want to pass back.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	LOCS
SUBR('DICT.PRD.LOCS',0,' ')	

DICT.PRD.OH.DS

SUBROUTINE (ANS,DAYS.SUP)

Routine will display a Yes or a No. If the on-hand is less than the days supplied then the answer will be yes otherwise, no.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	OH<14DS
SUBR('DICT.PRD.OH.DS', '14')	

DICT.PRD.ON.BID

This routine will display the bid(s) a product is on.

I-Descriptor Program Maintenance	
File Name : PRODUCT	
Dict ID : ON.BID	
SUBR("DICT.PRD.ON.BID")	
-	

DICT.PRD.ONHAND

SUBROUTINE (ONH,TYPE)

** Type : 1 = Total

** 2 = Stock

** 3 = Tagged

** 4 = Other

** 5 = All

* This gets called from PSUB I-Descriptors sometimes, too.

* So, @ID could be a PSUB ID, not a part number. Either way, this step will yield a PN from whatever was passed in.

DICT.PRD.ONHAND.LOC

This routine will provide the on-hand value for each location.

I-Descriptor Program Maintenance	
File Name : ENTITY.PN.IDS	
Dict ID : ONHAND	
SUBR('DICT.PRD.ONHAND.PN','1',PN)	

DICT.PRD.ONHAND.PN

SUBROUTINE (ONH,TYPE,PN)

** Type : 1 = Total

** 2 = Stock

** 3 = Tagged

This routine allows you to pass in the part number. It will display the on-hand value for any of the types specified above.

I-Descriptor Program Maintenance	
File Name : ENTITY.PN.IDS	
Dict ID : ONHAND	
SUBR('DICT.PRD.ONHAND.PN','1',PN)	

DICT.PR.D.ONHAND.VAL

```

SUBROUTINE (VAL,TYP)
  BRS = SEL.BR
  PN = @ID
  TYP = 1 ; * Stock On-Hand
  TYP = 2 ; * Tagged On-hand
  TYP = 3 ; * Stock Committed
  TYP = 4 ; * Tagged Committed
  TYP = 5 ; * Stock In PO
  TYP = 6 ; * Tagged In PO
  TYP = 7 ; * Stock In Transfer
  TYP = 8 ; * Tagged In Transfer

```

I-Descriptor Program Maintenance		View Only
File Name :	PRODUCT	
Dict ID :	STK.COMMIT	
SUBR('DICT.PR.D.ONHAND.VAL' , '3')		

DICT.PR.D.ONPO

This routine will work from any file where the product id is the key to the record and will pull back the quantity on an open purchase order for the shipping branch.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ONPO
SUBR("DICT.PR.D.ONPO")	

DICT.PR.D.ONWORK

SUBROUTINE (ONWORK)

* Returns the on work order quantity.

This routine works the same as the ONPO dictionary listed above. Any file for which the internal id is the product id will display the quantity on a Work Order for the shipping branch.

DICT.PR.D.OPEN.ORD

SUBROUTINE (ORDATA,VAL)

- *
- * ORDATA - The order information for the product (OUT)
 - * VAL - The type of information you want passed back in ORDATA (IN)
 - * NOTE : You must have a prompt on the dict that calls this for BRANCH
 - * Valid entries for "VAL" :
 - * VAL = 1 - Open Order ID's
 - * VAL = 2 - Required Date
 - * VAL = LD# - Ledger detail information from the attb specified by #
 - * VAL = LED# - Ledger information from the attb specified by #

- * VAL = STNAME - Shipto Name from the ENTITY in LED(5)<1,GEN>
- * VAL = BTNAME - Billto Name from the ENTITY in LED(1)<1,GEN>

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	OPEN.ORD
SUBR('DICT.PRD.OPEN.ORD' ,1)	

DICT.PRD.OPEN.PO

```

SUBROUTINE (ORDATA,VAL,PN)
*****
* VAL = 1 ; Open Order ID's          *
* VAL = 2 ; Required Date            *
* VAL = 3 ; Qty pending receipt      *
* VAL = 4 ; Receive Date             *
*****

```

You can pass in the part number and specify the value you want to return with this routine.

I-Descriptor Program Maintenance View Only	
File Name :	ENTITY.PN.IDS
Dict ID :	OPEN.PO.RECV.DATE
SUBR('DICT.PRD.OPEN.PO' ,4,PN)	

DICT.PRD.ORD.PT

SUBROUTINE (VAL)

This subroutine returns a VM list of order points for the branches in BR\$. The Network type to use for the order points is retrieved from the generic dict common prompt. This solves the problem of having to have three different order point dictionaries for the different purchasing network types.

* COMMONS:

- DICT.COMMON - Read from - BR\$ will get populated if empty.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ORDER_POINT
SUBR("DICT.PRD.ORD.PT")	

DICT.PRD.PIL

This routine works from a file for which the product id is the key to the records and will display the product's projected inventory level.



File Name : PRODUCT
Dict ID : PIL

File Name : PRODUCT
Dict ID : PIL.NO.XFER

File Name : PRODUCT
Dict ID : PIL<100

Page 277

I-Descriptor Program Maintenance

File Name : PRODUCT Dict ID : PER.UM
SUBR("DICT.PROD.PRICE",'2')

DICT.PR.D.SALES

SUBROUTINE (QTY,RANGE,ATTR,SD,ED,INC.DIR,SLS.UM)

For this routine to work you need to pass in the attribute definition and the range definition.

-----ATTR Definition

- 3 = Sales Qtys
- 4 = Purchases Qtys
- 5 = Transfers Qtys
- 6 = Sales \$
- 7 = Purch \$
- 8 = Xfer \$
- 9 = Sales COGS\$
- 10 = Sales GP\$
- 11 = Sales GP%
- 12 = Work Order Qtys
- 13 = Rental Qtys
- 14 = Work Order \$
- 15 = Rental \$

-----Range Definition

- 1 = User Defined
- 2 = MTD
- 3 = YTD
- 4 = Fiscal MTD
- 5 = Fiscal YTD

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	MTD.PURCHASES
SUBR('DICT.PR.D.SALES',2,4,'',' ',' ',' ')	

DICT.PR.D.SGRP.BR

SUBROUTINE (SGRPS)

- This program returns all the sell groups a product is in for the branch designated.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SELL.GRP.BR
SUBR("DICT.PR.D.SGRP.BR")	

DICT.PRD.SHORT.COMM

SUBROUTINE (SHORT.COMM)

Returns the Short Commodity Code that corresponds to the product's Commodity Code in the Valid Product Commodity Codes control record.

DICT.PRD.SLS

This routine will pull back the product sales using the PSUB file. You can only use this subroutine from the Product File or a file for which the Product ID is the @id.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SLS.DEC
SUBR("DICT.PRD.SLS","12/01/PY","12/31/PY")	

DICT.PRD.STK

SUBROUTINE (STKS)

Checks the status of a product in all branches in SEL.BR and returns 1 (YES) if stock and 0 (NO) if not if multiple branches are selected, a multi-value string is returned. It uses the standard formula which includes implied stock YES/NO logic.

If no branches are selected, all authorized branches are used.

STKS - VM list of stock flags for each BR in SEL.BR.(BOOLEAN) [OUT]

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	STK.YN
SUBR('DICT.PRD.STK')	

DICT.PRD.SUB.DESC

This routine will display the description of the substitute items a product is linked to.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SUBS_DESC
SUBR('DICT.PRD.SUB.DESC')	

DICT.PRD.SURPLUS

From the PRODUCT file or any file that the internal id is the product internal id, this routine will pull back the surplus amount.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SURPLUS
SUBR("DICT.PR.D.SURPLUS")	
-	

DICT.PR.D.SURPLUS.NOXF\$

SUBROUTINE (SURPLUS\$,BASIS)

 This routine will get the value of a product's surplus, using the Basis passed in.

Parameters:

SURPLUS\$ - Value of a product's surplus, Value Mark delimited by Branch.
 BASIS - Basis to use when getting Base Price of a product.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SURPLUS.NO.XFER\$
SUBR('DICT.PR.D.SURPLUS.NOXF\$', '')	

DICT.PR.D.SURPLUS.PT

This routine displays the surplus point. This routine can only be used from a file where the @ID is the internal id of the product.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	SURPLUS.PT
SUBR("DICT.PR.D.SURPLUS.PT")	

DICT.PR.D.UOM.MULTI

SUBROUTINE (VAL)

 This program is used to report qty concatenated to unit of measure written for report writer use.

VAL - Multi-value list of qty: unit of measure (OUT)
 DICT.COMMON is loaded when report writer runs. We are using fld 2 from dict value with is the mv postion for data.

I-Descriptor Program Maintenance		View Only
File Name :	PRODUCT	
Dict ID :	UOM.MULTI	
SUBR("DICT.PR.D.UOM.MULTI")		

DICT.PRDC.UOM.QTY

This routine counts how many unit of measures are assigned to an item.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	UOMQTY
SUBR("DICT.PRDC.UOM.QTY")	

DICT.PRDC.BR.GET.VAL

SUBROUTINE (VALUE,PN,ATTB.NO,SUM.TERRS)

 This routine is for getting a value from the PRDC.BR file from
 I-Descriptors in PRODUCT dictionaries

 VALUE - This is the value requested (OUT)
 PN - This is the part number that the value is being
 - requested for [IN]
 ATTB.NO - This is the value being requested [IN]
 SUM.TERRS - This is a flag whether to sum to sum the values (IN)
 - for the brs in a territory if a territory is
 - included in the br list to get values for

 Common Variables: DICT.COMMON<2> is used to get the branch(s) that the value is being
 requested for

DICT.PRDD.BR.GET.VAL

This dictionary is used in the product file to go out to the PROD.DYNAM file to pull back the
 branch specific date. The Prompt that you need on your dictionary for this to work is the Enter
 Br prompt, NOT the Enter Br/Tr/All.

You need the subroutine to call in the PN and the attribute number that you want to pull back.
 Example: Attribute 12 is the date the item was last counted.

SUBROUTINE (VAL, PN, ATTB)

*-----
 *** This routine is used as a wrapper for the dictionary to get values for the dictionary from the
 branch specific product array given the id and the attribute.
 *-----
 *** Parameters:
 *** VAL - The value to return.
 *** PN - The ID to get the VAL from
 *** ATTB - The ATTB # that the user is wanting a value for.
 *-----
 *** COMMON VARIABLES:
 *** DICT.COMMON<10> - This is used to get the value for the branch

DICTIONARY.GET.BRANCH.LOC.VALUE

```
SUBROUTINE (VAL,PN,ATTB)
VAL = "
VAL.CT = 0
SLOC = DICTIONARY.COMMON<6>
```

This routine will pull back branch specific data relative to the product dynam file. If you are using this subroutine in a file where the PN is accessible, you must pass that information. Then specify the attribute number you need to pull back.

DICTIONARY.GET.BRANCH.VALUE

```
SUBROUTINE (VAL,PN,ATTB,BR,SVM.NO)
```

Pass in the Part number, attribute from the prod.dynam file you want, branch you are in and the sub-value number.

I-Descriptor Program Maintenance	
File Name :	PRODUCT
Dict ID :	ONHAND.LOC
SUBR("DICTIONARY.GET.BRANCH.VALUE",@ID,1,"","")	

DICTIONARY.PRICE.CLASS

```
SUBROUTINE (SELL.PRC)
```

This subroutine tries to get a Price for a Product and Class, and is used by the PRICE.CLASS I-Descriptor off the Product File.

DICTIONARY.PRICE.PER

```
SUBROUTINE (PER.QTY,PN)
```

This routine will display the price per the unit of measure listed on the items price sheet. You can pass in the PN on the arguments.

DICTIONARY.PROD.STATUS

This routine will display the description of the product status

DICTIONARY.PROD.WGHT

```
SUBROUTINE (UNIT.WGHT,PN,PER)
```

DICTIONARY.PRODUCT.LOAD.KEY.ALTERNATE

```
SUBROUTINE (FLNM,REC.ID,RPL.STR,UPD.ERR,PASSER)
```

This routine will load the first n words of the Product description to the Product Keywords and Alt Desc fields.

NOTE: Designed for (S)et option only.

Format of RPL.STR - "W#" or "#" where # is the number of words to be taken from the front of the description field.

FLNM - File that we are updating
REC.ID - Record ID that needs updated
RPL.STR - Description after update
UPD.ERR - 1 - File not found
 2 - Record not found
PASSER - Reserved for future use

DICT.PRODUCT.UPD.DESC

SUBROUTINE (FLNM,REC.ID,RPL.STR,UPD.ERR,PASSER)

This routine will update the description in the Product File.
It is used with the replacement string all in double quotes and Designed for (S)et option only.

Opt1 - "W3,[WHITE]=[WHT] ... starts with the 3rd word in first value of description, and then moves everything from the 3rd word on in the 1st value along with the remaining values to the conversion of WHITE to WHT. Truncating 1st value before Word 3 and putting conv the following value positions.

Opt2 - "[WHITE]=[WHT]" ... does not touch 1st value position.
The remaining values are checked and all occurrences of the word
WHITE is exchanged for WHT.

FLNM = File that we are updating
REC.ID = Record ID that needs updated
RPL.STR = Description after update
UPD.ERR = 1 - File not found
 2 - Record not found
PASSER = Reserved for future use

DICT.PSUB.CALC.PRICE

This routine will calculate the best price when using this from the PSUB file. Looks like a way to compare the calculated price with an overridden price.

DICT.PSUB.COST

SUBROUTINE (VAL,OID,INVN,LDID,BASE)

Subroutine to be used in I-type dictionaries to return the cost of a transaction from the LD array associated to the LEDGER. This was written to be used on the PSUB file.

VAL - [OUT] - Cost determined by program using BASE number.
OID - [IN] - Ledger record ID (key)
INVN - [IN] - Invoice number
LDID - [IN] - Line item number
BASE - [IN] - Which field in the LD array to get the cost from.

DICT.PSUB.LEDL.INFO

SUBROUTINE (VAL,PSUB.ID,WORD)

This routine will go out to the Ledger log and pull back occurrences you specify. You must enter the word to search on.

DICT.PSUB.PRT.STAT

Display the print status of a transaction. This routine works from the PSUB file.

DICT.TAG.GET.VAL

SUBROUTINE (VAL,ATTB)

This routine is used to pull back the tagged information on the PSUB file.

DICT.TRANS.BR

SUBROUTINE (VAL,FLNM,REC.ID,DICT.ID,TERR.POS)

DICT.VEN.NOTES

SUBROUTINE (NTS,NOTE.ID)

This routine will pull back the vendor notes from the ENTITY.NOTES file. Supply the Note id you want to display.

DICT.WHSE.OP.QUEUE.IMREPL

SUBROUTINE (VALUE,COMPLETED)

-----Program Description-----

This routine can be used by dictionaries on the WHSE.OP.QUEUE file in order to select replenishments assigned to a particular user.

-----Parameter Definition-----

VALUE - The value of the dictionary for the current (OUT)
- record.

COMPLETED - Boolean, whether or not completed replenishment (IN)
- should be included.

DICT.X.AXIS.DESC

This routine will display the description of the X axis from your matrix file. Customer, Customer Type, Class, Quote etc.

DICT.Y.AXIS.DESC

This routine will display the description of the Y axis from your matrix file. Group or Product.

DICT.Z.AXIS.DESC

This routine will display the Z axis, such as territory and branch specific information.

DICT.ZERO.HISTORY

Used on the product file, this routine will display a "1" if a product does not have any history for a branch defined.

DICT.ZERO.ONHAND

Used on the product file, this routine will display a “1” if a product does not have an on-hand for a branch defined.

DICT.ZERO.ORDERS

Used on the product file, this routine will display a “1” if a product does not have any open orders for a branch defined.