PRICE

A Lockheed Martin Company

PRICE Enterprise™ Release 1.0
Client Reference

FIRST EDITION
JUNE 1997

PRICE Systems
Mt. Laurel, New Jersey

TABLE OF CONTENT

TABLE OF FIGURES

1. Overview

This document is a supplement to your existing PRICE H and RDD-100 manuals. It describes the PRICE Enterprise cost modeling extensions. PRICE Enterprise adds a powerful interface capability to your PRICE desktop software products. Using the Enterprise tools, you are able to build and execute seamless links to other software tools vital to your business. It will enable you to better serve the decision makers within your company by providing real time access to the design information resident in the design automation tools that your Engineers use. This document focuses on the RDD-100 to PRICE interface that we call Integrated Design To Cost (IDTC). This document is organized as follows:

1. **Overview:** An introduction to the individual tools that comprise IDTC, IDTC process concepts and the software components of PRICE Enterprise
2. **PRICE Enterprise Process Overview:** A review of the traditional cost estimating process, an introduction to the IDTC cost process and a top level view of a generic IDTC process.
3. **PRICE Enterprise Process Details:** A detailed decomposition of the IDTC process, including some information on the internal processes that accomplish the translation from the engineering tool to PRICE.
4. **Creating and Maintaining the CA File and Sync File:** Instructions on how to create and maintain the cost estimator's cost input files and information on how to use these files effectively.
5. **PRICE Rule Language (PRL):** A language reference guide that details the functionality and syntax of PRL and contains example PRL scripts.

The key capability provided by IDTC is traceability. As Figure 1-1 illustrates, IDTC enables you to create and maintain traceability from requirements to functions to component architecture to cost.

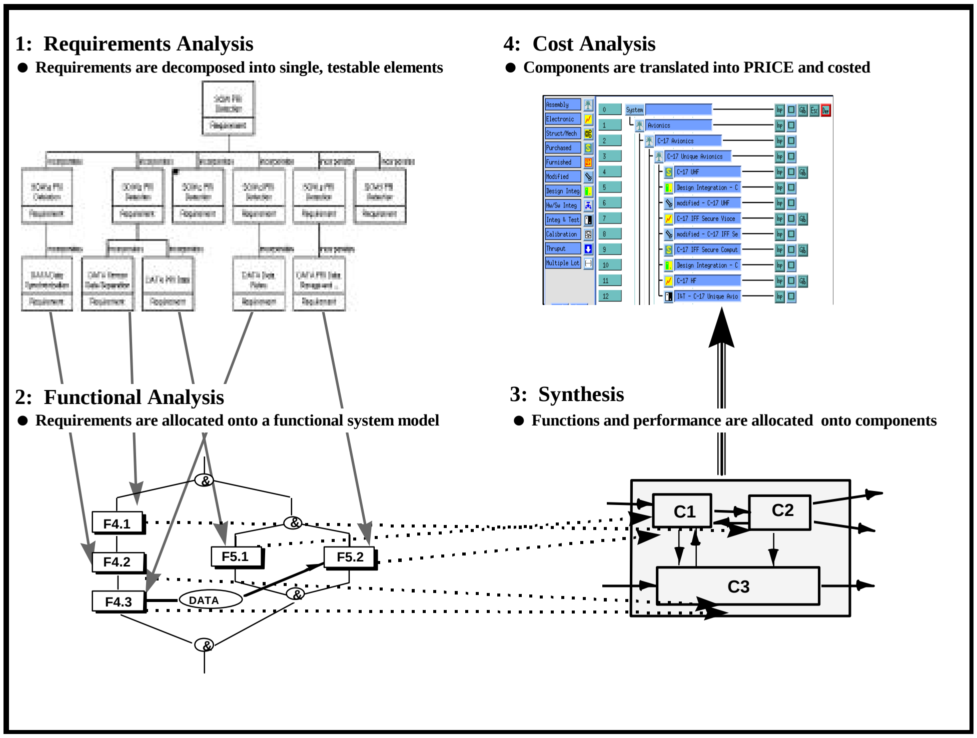


Figure 1-1 IDTC work flow

1.1 PRICE Models

The PRICE Systems cost tool suite consists of four integrated cost estimating models. PRICE H: hardware development and production. PRICE HL: hardware life cycle. PRICE M: modules and microcircuits. PRICE S: software development and life cycle. These models use systems of equations that implement Cost Estimating Relationships (CERs). CERs relate non-cost parameters like size and manufacturing process to cost and schedule. The PRICE tools have been in continuous use by cost estimators and engineers in government and industry for over twenty two years. At the time of this writing, the PRICE H & HL tools have been integrated into the Enterprise tool. A prototype PRICE S integration exists and is available, contact us directly if you wish to know more about the status and functionality of that feature, as development is ongoing. In addition, an effort to integrate PRICE M is also underway. The PRICE M integration is focused on the Mentor Graphics BoardStation toolset.

The PRICE strategy for interfacing with RDD was to build a mechanism that supports the transfer of parametric inputs into PRICE from any design tool and the transfer of cost and schedule information out of PRICE. The mechanism we use for this is called PRL (PRICE Rule Language). PRL is used to encapsulate the estimating rules that translate design parameters into PRICE inputs. PRL is an interpreted C-like language that easily translates various file formats and can map multiple design parameters into PRICE inputs. It is tied into the PRICE API, allowing it to embed the translated parameters into the model, run it and iterate if needed, and export (again in multiple formats). PRL eliminates the problems that usually occur when two tools are integrated with a program because it is not tied to a PRICE file format and is extensible enough to read the interchange formats of the tools it interfaces with. Because it is not compiled, PRL scripts remain viable as each point tool follows its upgrade path. In addition, PRL merges Cost Analyst information from two sources: a default file and an override file.

1.2 RDD-100

RDD-100 is a systems engineering tool built on an ERA (Entity, Relationship, Attribute) database with substantial graphical capabilities. RDD-100 supports requirements analysis, functional analysis, and physical decomposition. Using RDD, a systems engineer can decompose requirements down to single, testable units, specify and test the associated functionality, and allocate functions onto hardware and software components. This results is three hierarchical system views that are interrelated: requirements, functions, and components. The component view is actually an equipment breakdown structure. PRICE and Ascent created a set of database extensions to the component view that support the System Engineer's cost needs through the life of a program. The details of the RDD schema extension are found in the accompanying RDD IDTC guide.

1.3 IDTC Process

The IDTC methodology begins in system definition. With IDTC, the estimating process is carried on electronically. Once a candidate design (component architecture) has been made, the systems engineer exports the physical description of the design from RDD-100. This description is read by PRICE and translated into cost estimating parameters. The cost estimating parameters are then merged with information from the Cost Analyst to produce a complete data set which is sent to the parametric estimating engine. The engine produces a cost and schedule estimate for the system and exports that data back to the engineer. The engineer then reads that data into RDD-100 where it aligns with the existing structure. The IDTC process can be initiated by the estimator or the Analyst.

IDTC integrates the engineer and estimator electronically, codifying estimating rules into computer applied estimating relationships and eliminating keypunch errors. The IDTC estimating process is an improvement to the organizational process in every way. It is faster, enabling more alternatives to be explored. It is more accurate and repeatable because the rules that are applied are controlled by the estimator, codified into a PRICE Rule Language script, and executed by a computer. Because the rules are codified the engineer doesn’t need to meet with the estimator every time an estimate is desired. That doesn’t mean they aren’t both involved, they are just out of each other’s critical path. Because it is parametrically based, it does not use a bill of materials. With IDTC an estimate can be turned around in minutes instead of days or weeks.

But where IDTC really pays off is after the initial estimate. The initial estimate is back populated to RDD in a "budgeted cost" field associated with each component. As the design matures and alternatives are explored, the cost estimate is back populated into a "predicted cost" field within each component. Through the use of RDD consistency checks, the systems engineer can then validate each cost estimate against the component cost budgets automatically. And, if a subsystem reallocation is required, the integrated requirements, functions, and component hierarchies can be automatically traced to determine everything that is impacted. That doesn’t eliminate judgment, it adds to your ability to make good judgments. The reallocation estimate can be accomplished in minutes, not days, meaning that decisions can be based on cost and the estimate will be for the entire system.

1.4 PRICE Enterprise Software Components

1.4.1 PRL

The PRICE Rule Language allows you to:

- Define the format of incoming data
- Import and translate data into PRICE parameters
- Execute and iterate the cost model
- Export cost and schedule estimates in any format

1.4.2 CA File

The Cost Analyst File contains default data from the cost analyst

1.4.3 Sync File

The Sync File contains cost input overrides.

1.4.4 License Manager

PRICE models (UNIX version) rely on the PRICE License Manger to verify licensing. The PRICE License Manger is a network and client server based program. When a user starts a PRICE model (PRICEH or PRICES) it sends out a request to the PRICE License Manager to determine if the network has the license to run the model. The PRICE License Manager will then read the price license file, /price/lib/pricekey, to determine if the key to running the model is correct. It then answers the request based on the data in the license file. Only one machine on the network needs to run the PRICE License Manager. This machine must have access to the license file /price/lib/pricekey. An environment variable, PRICEHOST, must be set to the host name where PRICE License Manager is running.
The following are some of the error messages related to PRICE license manager:

- ***“Could not get environment variable PRICEHOST for license manager”***
  This may be caused if the user’s **PRICEHOST** environment variable is not being properly set. PRICEHOST value should be set to the name of the machine where license manager is running.
- ***“Failed to connect with the license server on* <system name>*”***.
  This might be due to an incorrect value for the **PRICEHOST** environment variable, or PRICE License Manager is not up. Check to make sure PRICEHOST is correctly set, and/or the PRICE License Manager is up and running. If not currently running, PRICE License Manager can be manually started using the following command

/price/bin/PRICElmd &

It is strongly suggested to login as root to do so.

- ***"Could not open PRICE license key file"***
  The reason may be that **/price/lib/pricekey** does not exist. Make sure the license file you received from PRICE Systems is in the correct directory.
- ***"PRICE H/L/M License expires in … days"*** or ***"PRICE H/L/M License expired."***
  This means PRICE E license is about to expire or has expired and needs a renewed license key file.
- ***"Incorrect key to PRICE H/L/M"***
  This means /price/lib/pricekey file is damaged. Contact PRICE Systems Enterprise Customer Support for help.

1.4.5 Command Line Interface

```
pricee   [-help] [-NB] [-SW] [-x]
         [-c <ca_file>] [-s <sync_file>]
         [-I <input_template>] [-i <input_file>]
         [-O <output_template>] [-o <output_file>]
```

GENERAL OPTIONS

-help	Display command line options for pricee.
-NB	Suppress starting banner.
-SW	Turns on the Software Beta Test Switch.

OPTIONS WITHOUT GRAPHICAL USER INTERFACE (GUI)

-x	Run model without GUI. If this option is not selected you will not be able to do an import /export without the GUI. In this case PRICE E will start with its GUI.
-c <ca_file>	CA File for import.
-s <sync_file>	Sync File for import.
-I <input_template>	*PIRL import file.
-i <input_file>	*File to be imported.
-O <output_template>	*PIRL export file.
-o <output_file>	*File to be exported.

*Please note that the import and export processes are tied together. You cannot do one without the other. This means that the -I, -I, -O, -o options must be specified together or the import and export will not process. In this case PRICE E will just return to the command prompt.

2. PRICE Enterprise Process Overview

2.1 Traditional Cost Process

Traditional cost estimating processes have relied upon estimators and engineers to work within a functionally-oriented organizational process to meet cost needs. This process supported the creation of program deliverables but fell short of enabling cost effective designs.

From the perspective of a cost estimator, the process requires the communication of engineering specifications, a translation of those specifications into model inputs and an estimating breakdown structure (EBS), and execution of the model to obtain some preliminary results. These results are then iterated back to engineering to resolve any questions that arise. Because of the manual nature of the data transfer, there are often several iterations.

Cost Estimator (Traditional Role)

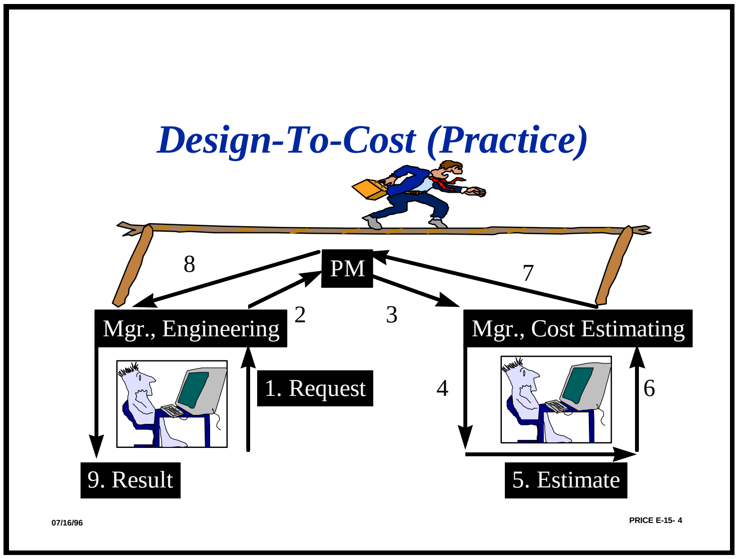
- Interpret technical information supplied by engineer and transform data into Parametric Input Data
- Run model to obtain preliminary results
- Analyze results and
- Re-interview engineer ("Is this what you meant?" or "The packaging density of electronics came out as 120 lb./cu ft. Can we review your weight and volume calculations?")
- Iterate estimate based on revised information from engineers.

From the perspective of the engineer, the traditional process requires the creation of an engineering estimate based upon labor hours and a proposed bill of materials (BOM) that is supported with vendor quotes. In addition, the engineering group also supports the cost estimator in the creation of a parametrics-based estimate.

Engineer (Traditional Role)

- Construct engineering labor hours estimate
- Construct proposed bill of materials(BOM)
- Provide labor hours estimate and BOM to pricing department to obtain vendor quotes and apply negotiated labor rates and loadings.
- Supply technical information about new project in support of parametric estimate for internal cross check or customer deliverable.

Generally, these estimates are compared and a resolution of some sort is reached. Exactly how this occurs and which estimate is used as the basis for moving forward varies from company to company, but the general process is the same. Two estimates are built and compared. This process is valuable in that it provides two cost perspectives from varying viewpoints: the parametric (top down) and the engineering (bottom-up). However, this process has proven too slow and costly to be responsive in a Design To Cost (DTC) environment.



Design-To-Cost (Practice)

PM

8

7

2

3

Mgr., Engineering

Mgr., Cost Estimating

1. Request

4

6

9. Result

5. Estimate

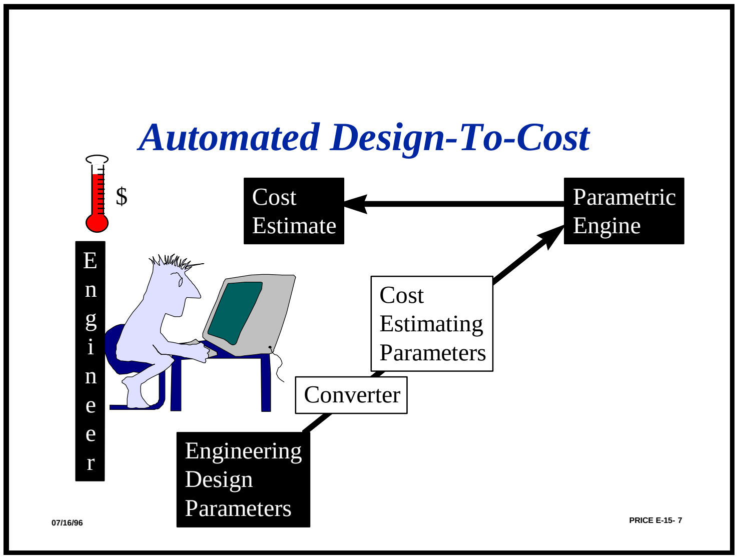
07/16/96

PRICE E-15- 4

Figure 2-1 Design-to-Cost in Practice

2.2 Integrated Design To Cost (IDTC) Process

The IDTC toolset and process is a response to the cost and speed problems associated with traditional, functionally oriented cost processes. Customer pressures (both government and commercial) for decreased cost and product development cycles (time to market) aren't just increasing. In fact, they are increasing at an increasing rate. In response, businesses have reoriented their processes away from functional models toward cross functional Integrated Product Teams (IPTs). IPTs have been successful but, in order to realize the complete benefits of cross functional teaming, businesses dependent upon new product development are finding that cross functional software tool sets must be employed. IDTC is responsive to this requirement. How does that work? Let's begin by looking at the automated process shown in Figure 2-2.



Automated Design-To-Cost

$

Engineer

Cost Estimate

Parametric Engine

Cost Estimating Parameters

Converter

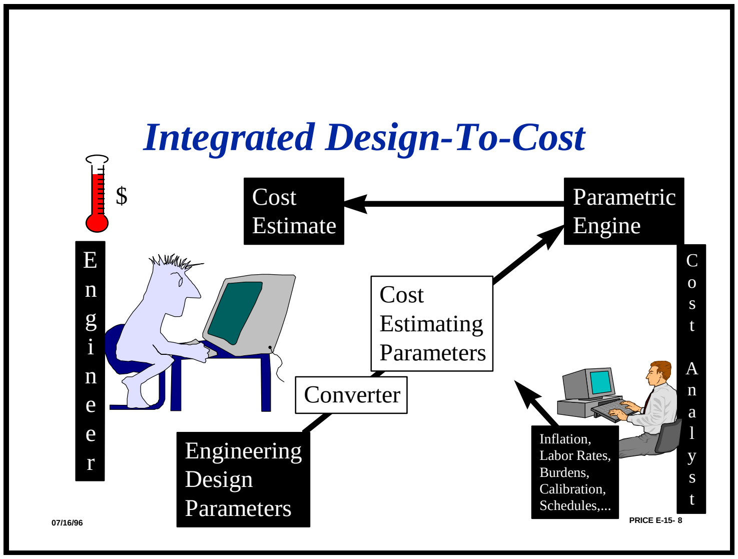
Engineering Design Parameters

07/16/96

PRICE E-15- 7

Figure 2-2 Automated Design-to-Cost

In an automated Design To Cost process, design parameters from an engineering tool are translated through a software converter into cost estimating parameters, run through a parametric cost engine, and returned to the engineer. But is that enough? In fact, it is not. Why not? Because the cost estimator is missing. An IDTC process is shown in Figure 2-3.


Integrated Design-To-Cost

$
Engineer
Cost Estimate
Parametric Engine
Cost Estimating Parameters
Converter
Engineering Design Parameters
Inflation, Labor Rates, Burdens, Calibration, Schedules,...
Cost Analyst

07/16/96

PRICE E-15- 8

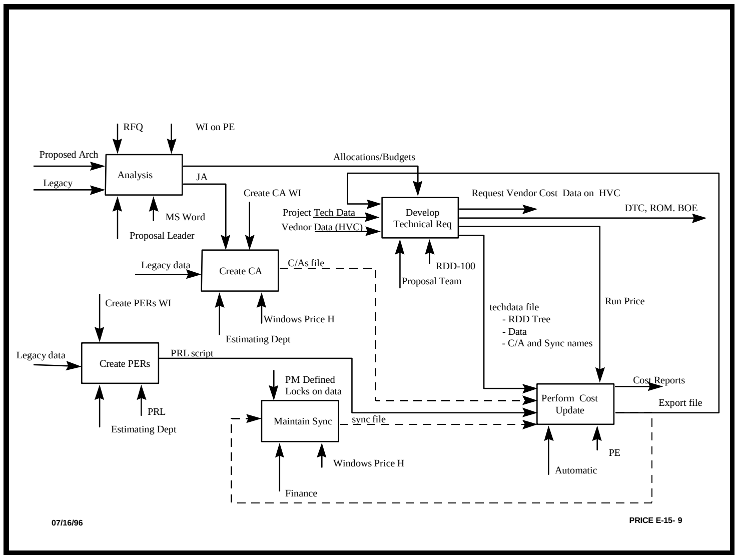
Figure 2-3 Integrated Design-to-Cost

IDTC considers the engineer and the estimator electronically and organizationally. The power inherent in this paradigm is that, although both functions are consulted for each estimate, neither function sits in the critical path of the other.

2.3 A Generic IDTC Process

Figure 2-4 uses an IDEF format to show a generic IDTC process. This contains six steps that are performed in the following order.

1. Define PRL script (Create PERs)
2. Organizational analysis
3 & 4. Create CA File / Create component architecture in RDD (concurrent)
5. Obtain cost estimate
6. Create and maintain Sync File



07/16/96

PRICE E-15- 9

Figure 2-4 IDTC Process Overview

2.3.1 Define PRL Script (Create PERs)

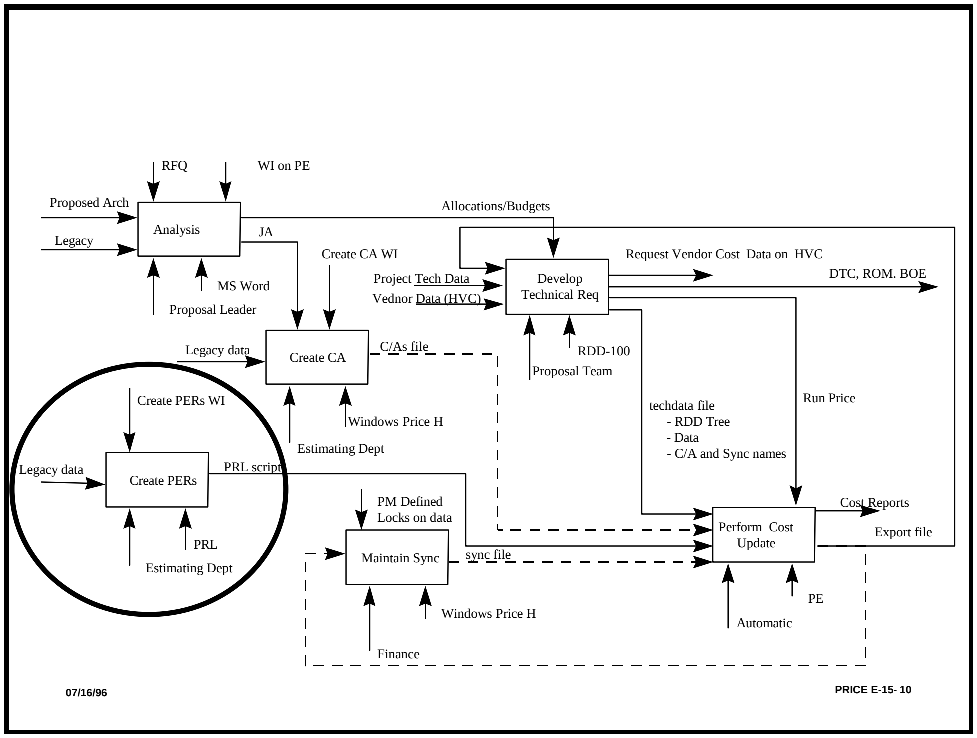


Figure 2-5 IDTC Process Overview Focusing on Definition of PRL Script

PERs are Parameter Estimating Relationships. This is a codification of the translations that the estimator performs after interviewing the engineer to create cost estimating parameters for use in the cost model. A simple example of this is the translation of dimensions like length, width, and depth into volume. Creation of a PRL script is something done outside of the project process. It is accomplished once and should address all product lines and application domains for which IDTC is intended to be used. If your product line expands or your use of IDTC is expanded to incorporate additional product lines the PRL script can be expanded accordingly. Changes to the PRL script should not, however, be considered part of the estimating process. Once the PRL script is defined it is added to your company's configuration management system so that it can be accessed by projects.

2.3.1.1 Inputs

Legacy calibration data is the primary input to a PRL creation task. This is the mapping of product line information on technology, process, application domain and personnel qualifications that will drive the cost model to create costs that are consistent with history. The translation function implemented in PRL will instantiate a set of PERs in a system of equations regressed from the calibration history. In this way, engineering descriptions are mapped to parametric model inputs that, when run through the cost model, will result in accurate costs.

2.3.1.2 Controls

You company's work instructions and procedures will determine how this is accomplished, who performs the work, what approvals are required, and the fidelity level (perhaps specified in statistical terms) demanded.

2.3.1.3 Resources

This work is generally carried out by personnel in the estimating department as they generally have access to the historical data and tools required, are trained with the cost models, and are the long term custodians of the cost estimating system.

2.3.1.4 Outputs

The output of this process is a PRL script which can be put under configuration control and be made available to future projects.

2.3.2 Organizational Analysis

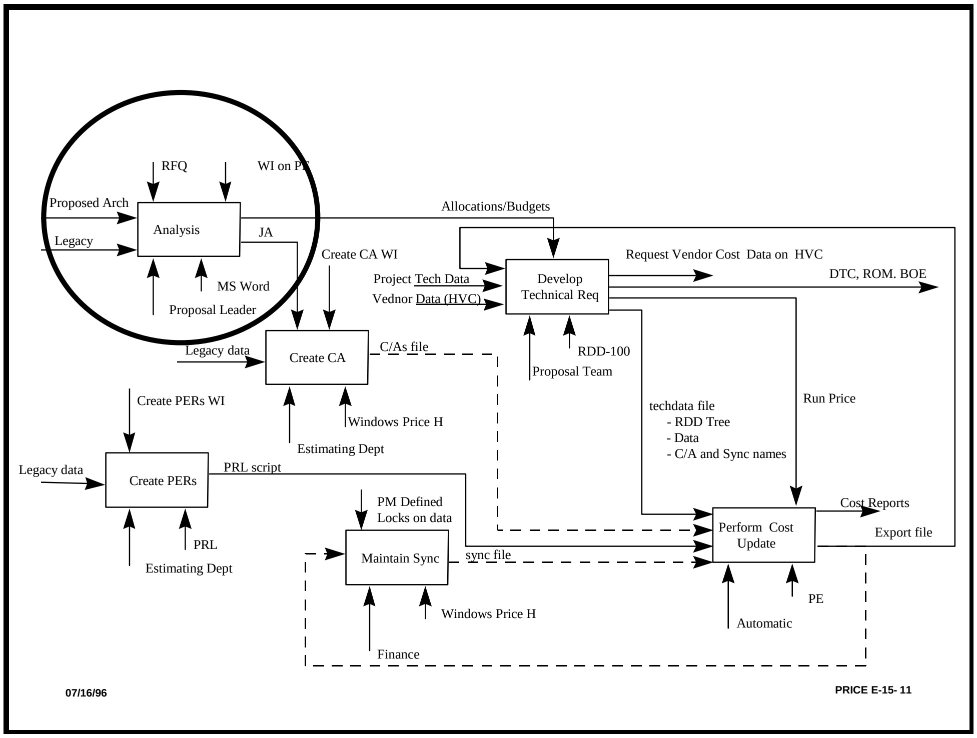


Figure 2-6 IDTC Process Overview Focusing on Organizational Analysis

For each project that your organization considers, an organizational analysis takes place. This is the basis for bid/no bid decisions, and, if a decision to proceed is made, this analysis becomes the originating source of program goals.

2.3.2.1 Inputs

A proposed architecture and legacy data (technical, cost, and organizational) are the primary inputs.

2.3.2.2 Controls

Internal work instructions and the RFP.

2.3.2.3 Resources

A proposal leader, generally supported by a top level cross functional team and office automation tools (MS word for example).

2.3.2.4 Outputs

A job analysis document that identifies the key performance parameters required for success. From a cost perspective, this document contains top level budgetary allocations and references the relevant work instructions for down stream processes.

2.3.3 Define CA File

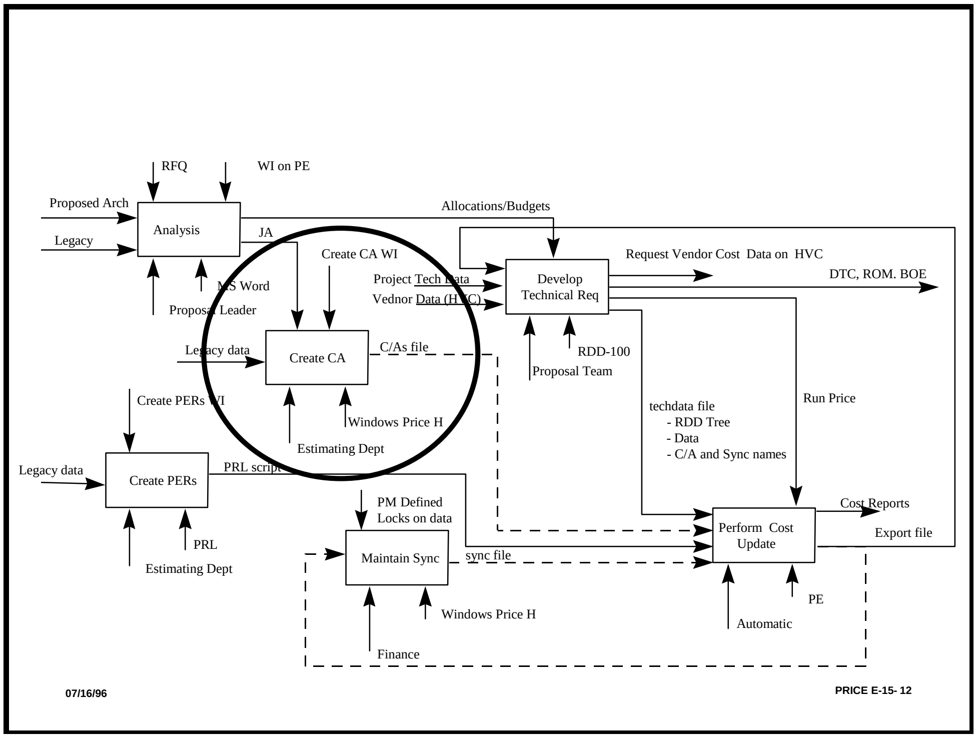


Figure 2-7 IDTC Process Overview Focusing on Definition CA File

Once the Job Analysis is complete, the cost department sets about creating a “Cost Analyst” file that contains model parameters that will not be supplied by the engineering tools.

2.3.3.1 Inputs

Calibration data (globals), inflation tables, labor rate and loading information (financial factors), and deployment data (if LC costing is desired).

2.3.3.2 Controls

The Job Analysis and relevant work instructions.

2.3.3.3 Resources

Cost estimators, using the PRICE models.

2.3.3.4 Outputs

A Cost Analyst File (CA File) which should be placed under configuration control for use on the program.

2.3.4 Develop Technical Requirements

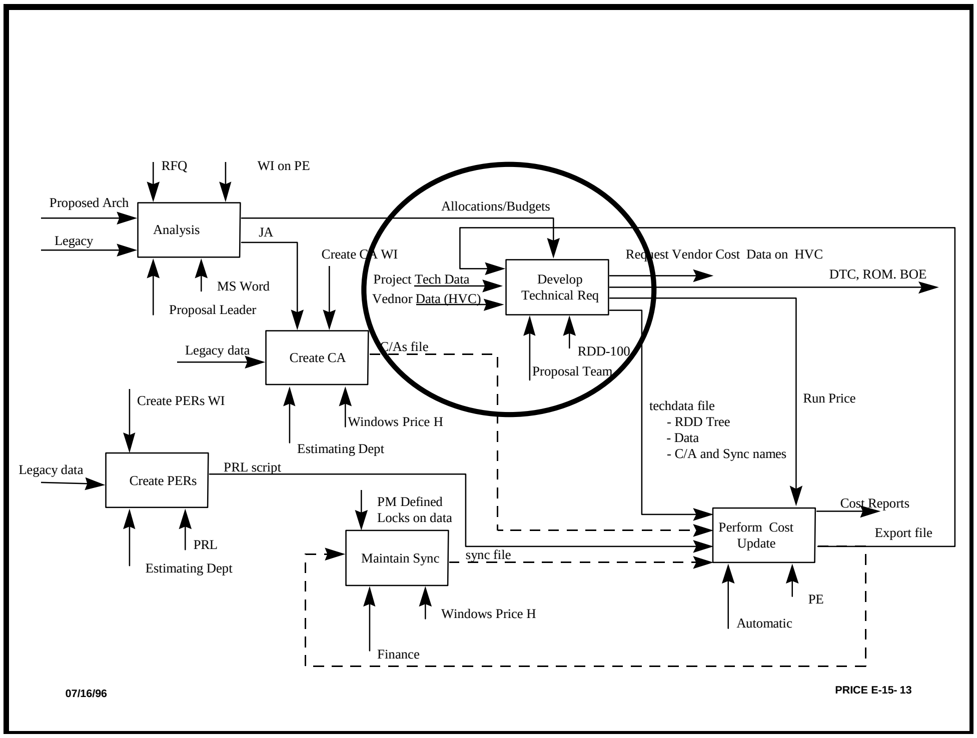


Figure 2-8 IDTC Process Overview Focusing on Development of Technical Requirements

Concurrently, the systems engineering group begins to (1) decompose the top level technical requirements down to single, testable units, (2) map those requirements onto a functional system model and (3) allocate those functions onto candidate component architectures.

2.3.4.1 Inputs

Technical program data obtained from the RFQ, the Job Analysis and the customer and vendor quotes for High Value Components (HVC) that will likely be subcontracted.

2.3.4.2 Controls

The Job Analysis and its budgetary allocations.

2.3.4.3 Resources

The systems engineering department, using a design tool like RDD-100.

2.3.4.4 Outputs

A candidate architecture which references the appropriate PRL script, CA File and Sync File. This architecture should be under CM(Configuration Management) control.

2.3.5 Obtain Cost Estimate

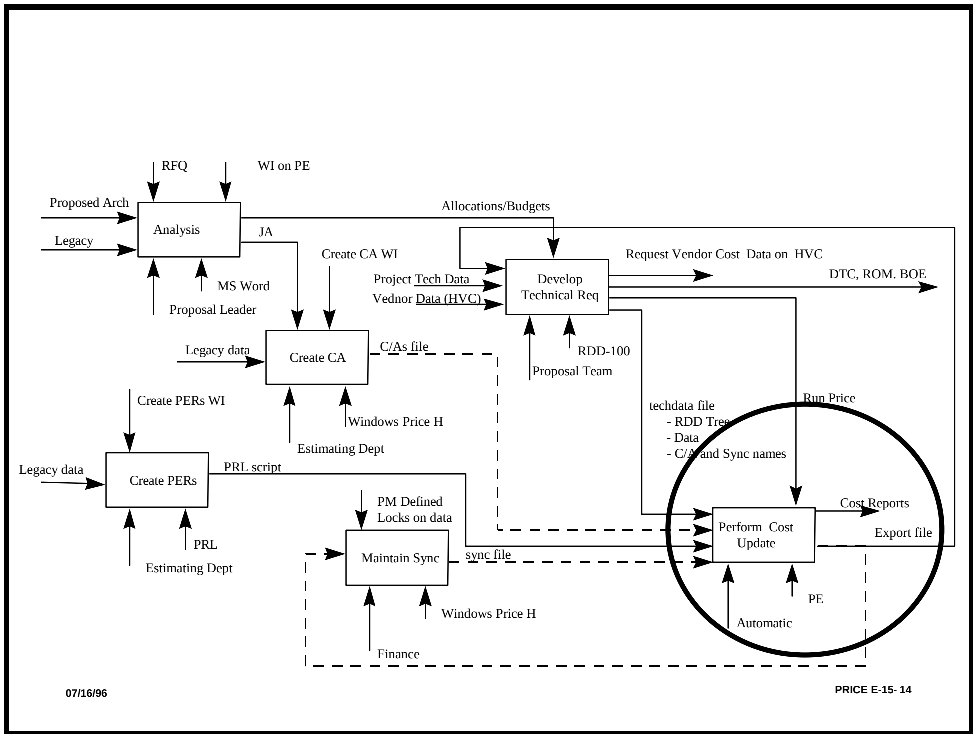


Figure 2-9 IDTC Process Overview Focusing on the Cost Estimate

At this point you have sufficient information to begin conducting integrated cost estimates. Certainly it is likely that previous estimates were done, either on the back of an envelope, parametrically, or bottom-up. But it is only now that a sufficient database of electronic information exists to support cross functional estimating in an automated fashion. It is at this point that the value of IDTC begins to reveal itself because engineers and estimators can initiate an estimate without first meeting

2.3.5.1 Inputs

2.3.5.1.1 Candidate Architecture, stored electronically in RDD that references a CAFile and PRL script.

2.3.5.1.2 CA File, stored in a PRICE “.hpr” format

2.3.5.1.3 Sync File, the Sync File is an output of the first (and perhaps later) iteration(s).

2.3.5.1.4 PRL Script, implementing the derived PERs that will drive the cost model to accurate estimates.

2.3.5.2 Controls

Either the estimator or engineer can initiate an estimate.

2.3.5.3 Resources

The process is automatic.

2.3.5.4 Outputs

Cost estimating reports, an input stream to back populate the systems engineering tool and, optionally, the Sync File.

2.3.6 Modify Sync File as needed

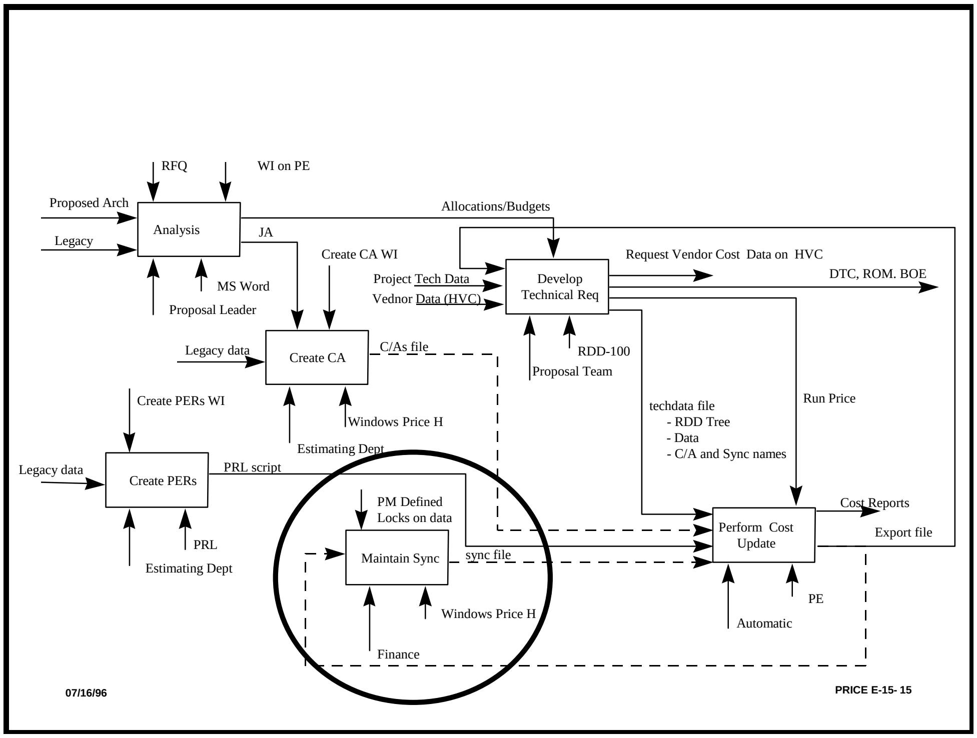


Figure 2-10 IDTC Process Overview Focusing on the Sync File

Beginning with the first estimate, a “Sync File,” potentially containing override information can be exported from the estimating process for use on subsequent estimates. Override data is based upon empirical legacy data or heuristic estimating relationships which may be imputed into future estimates. This allows you to extend the IDTC estimating capability beyond the domains for which the PRL script was designed. Off-line, the PRL can later be expanded to incorporate the relationships that are manually supplied via the Sync File.

2.3.6.1 Inputs

The primary input to a Sync File is an “.hpr” project file saved from an estimate. As a result, all of the inputs to an estimate should be considered as secondary inputs.

2.3.6.2 Controls

Work instructions regarding ownership of data attributes.

2.3.6.3 Resources

Primarily the estimator using the PRICE models.

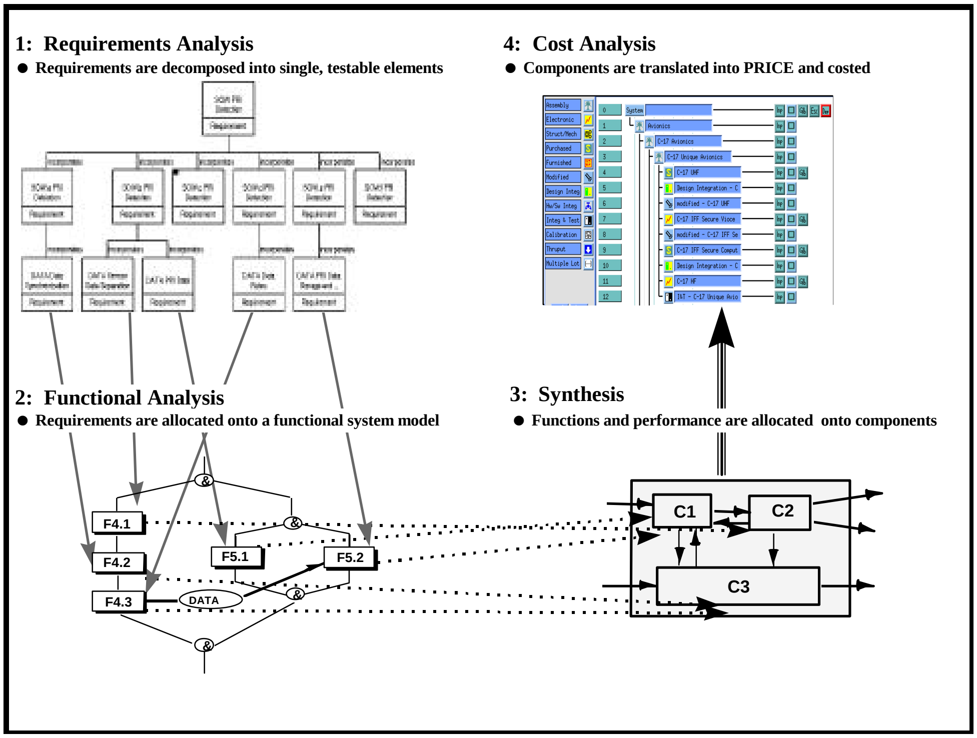
2.3.6.4 Outputs

An updated Sync File.

3. PRICE Enterprise Process Details

3.1 Creating a component architecture in RDD

Within RDD-100, requirements are decomposed until a functional model can be specified. At that point the functions are allocated onto a component hierarchy. This process is portrayed graphically in Figure 3-1.



1: Requirements Analysis

- Requirements are decomposed into single, testable elements

4: Cost Analysis

- Components are translated into PRICE and costed

2: Functional Analysis

- Requirements are allocated onto a functional system model

3: Synthesis

- Functions and performance are allocated onto components

Figure 3-1 IDTC Traceability

From the perspective of cost, the intermediate output required for costing is a component hierarchy like the one shown as in Figure 3-2.

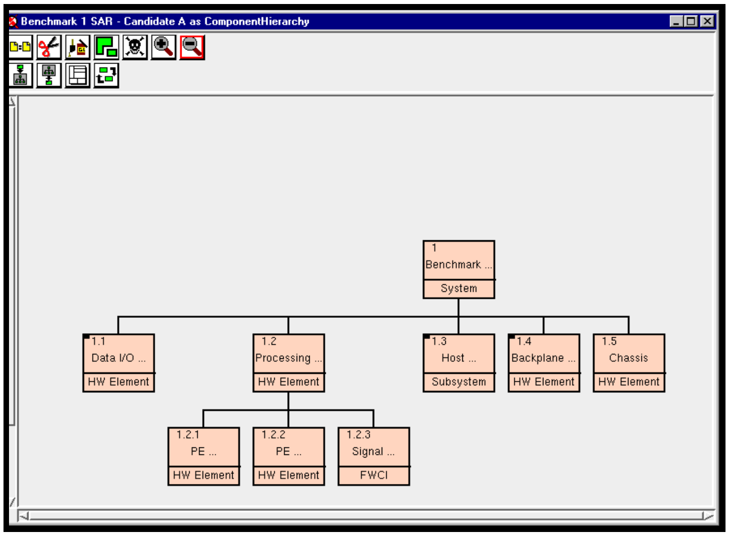


Figure 3-2 Component Hierarchy

For details on how a component hierarchy is built and what tools are available to help a systems engineer do this see the accompanying RDD IDTC documentation. For our purposes, let's consider that it has been correctly completed and an estimate is now desired. How do you export the component hierarchy out of RDD and into PRICE?

3.2 Exporting a component architecture

3.2.1 Select the "Internal Report" sub-item from the "Print" menu item.

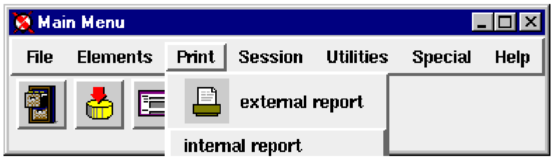
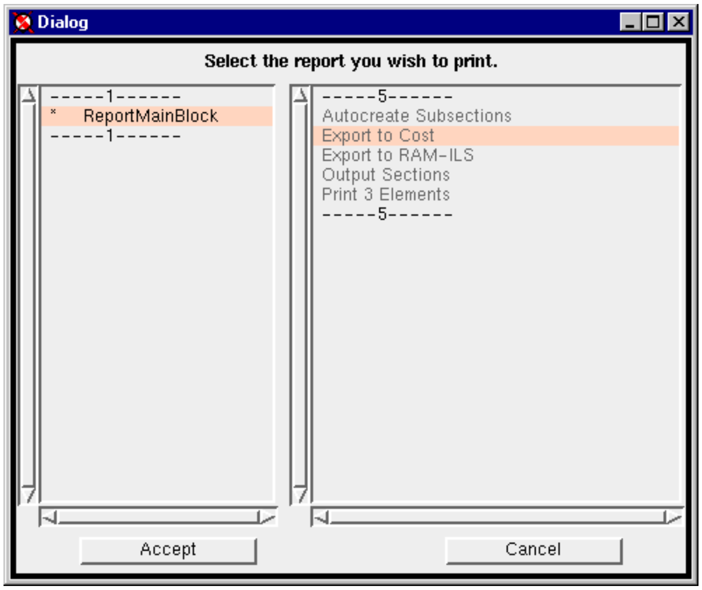


Figure 3-3 RDD-100 Main Menu

3.2.2 Select the "Export to Cost" item.

The window that appears in response to the above menu selection contains a list of the internal reports available to you. Select the "Export to Cost" report.



Select the report you wish to print.

```
-----1------
*   ReportMainBlock
-----1------
```

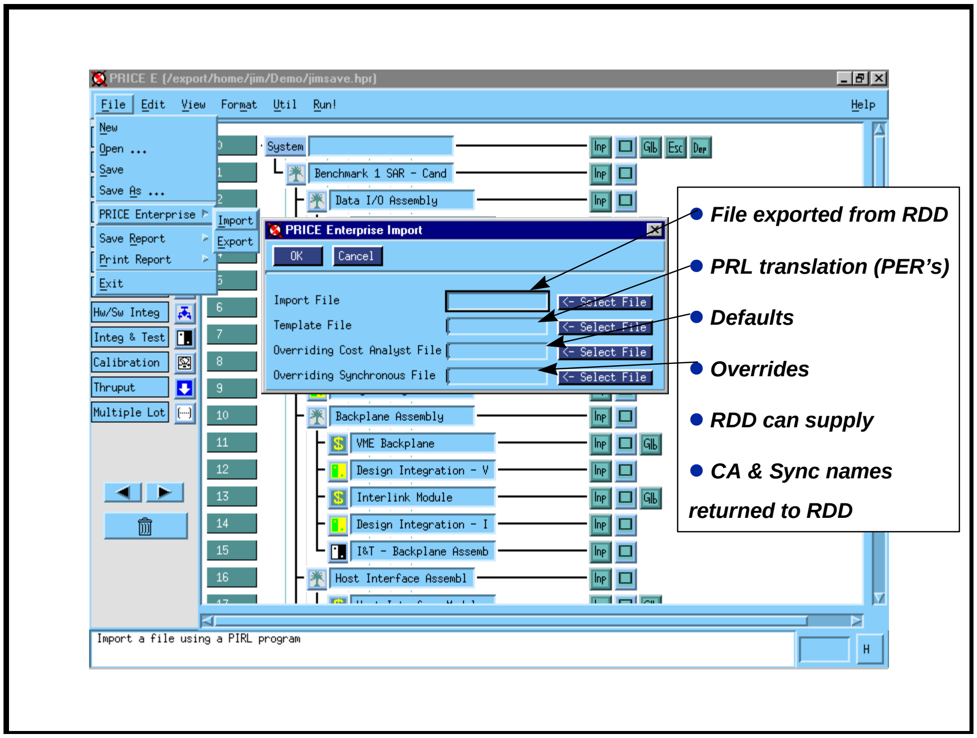
```
-----5------
Autocreate Subsections
Export to Cost
Export to RAM-ILS
Output Sections
Print 3 Elements
-----5------
```

Figure 3-4 Internal Report Dialog

This creates a file in the format that PRICE Enterprise's IDTC translator expects. The name and path is determined by the user who should follow the organizational CM convention detailed in your IDTC work instructions.

3.3 Importing a component architecture into PRICE

Once the file is created, go to the PRICE Enterprise tool and select "File," "PRICE Enterprise," and then "Import" in succession. This presents the following dialog box shown in Figure 3-5. Only the names of the RDD "Export to Cost" file and the PRL translation script need to be supplied (the remaining information is actually contained within the RDD export). You may optionally enter the names of the CA File and Sync File if you desire to override the files requested from the RDD export file.

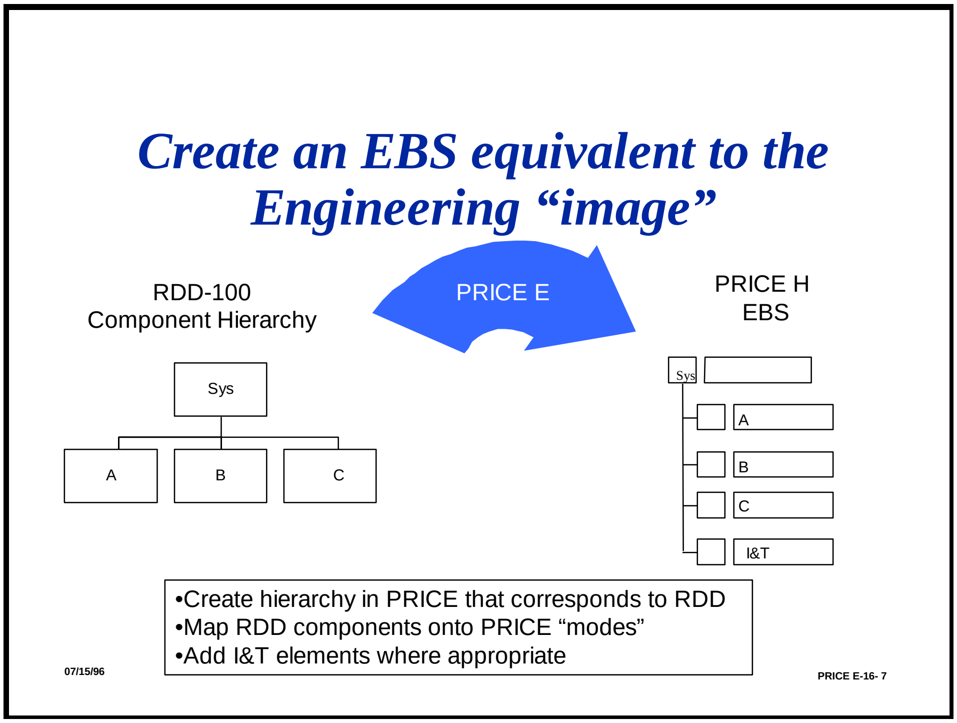


- ***File exported from RDD***
- ***PRL translation (PER's)***
- ***Defaults***
- ***Overrides***
- ***RDD can supply***
- ***CA & Sync names returned to RDD***

Figure 3-5 PRICE Enterprise Import Dialog

After clicking on the "OK" button, the translation process begins. The details of the translation process follow.

3.3.1 Create Estimating Breakdown Structure (EBS)



Create an EBS equivalent to the Engineering "image"

- Create hierarchy in PRICE that corresponds to RDD
- Map RDD components onto PRICE "modes"
- Add I&T elements where appropriate

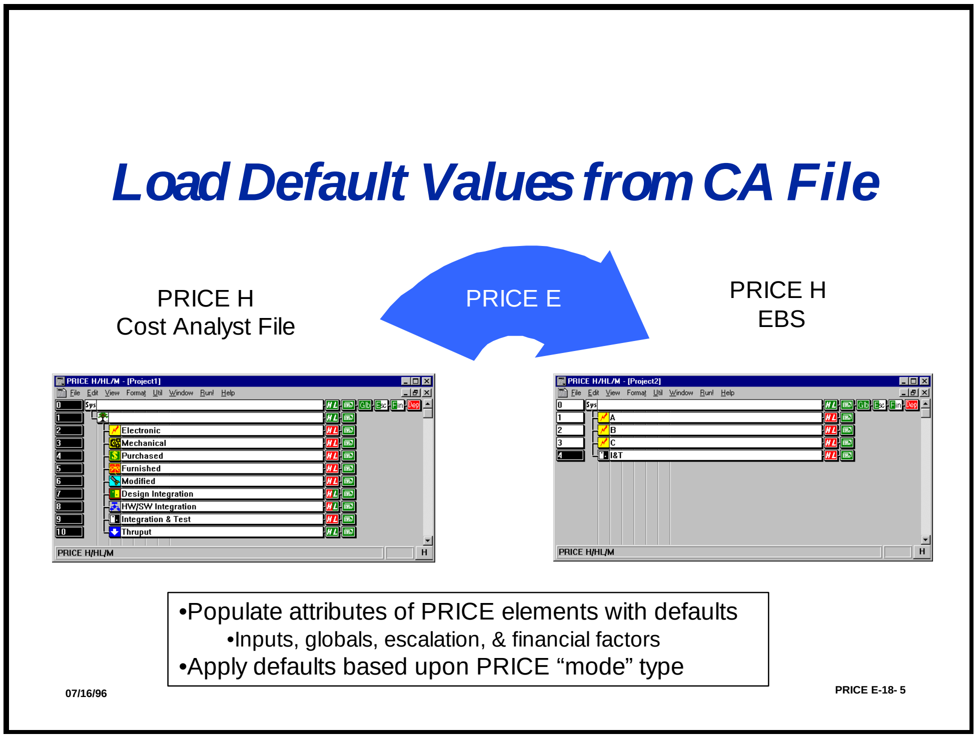
07/15/96

PRICE E-16- 7

Figure 3-6 Create EBS from Component Hierarchy

The first step in the translation process is the creation of a “target EBS” in PRICE that matches the RDD component hierarchy. In the simple example shown above, a system composed of “A”, “B”, & “C” is created in PRICE to match an equivalent structure in RDD. It is the PRL script that translates the component hierarchy into PRICE. An integration and test element was added automatically during the translation by the PRL script which recognized a decomposed element (in this case the system element). The generic PRL script also automatically adds hardware/software integration and test and design integration elements when the structure implies a requirement. The rules that guide the creation of PRICE elements are contained within the PRL script and therefore are under the control of your company.

3.3.2 Apply the CA File



Load Default Values from CA File

PRICE H
Cost Analyst File

PRICE E

PRICE H
EBS

- •Populate attributes of PRICE elements with defaults
  - •Inputs, globals, escalation, & financial factors
- •Apply defaults based upon PRICE “mode” type

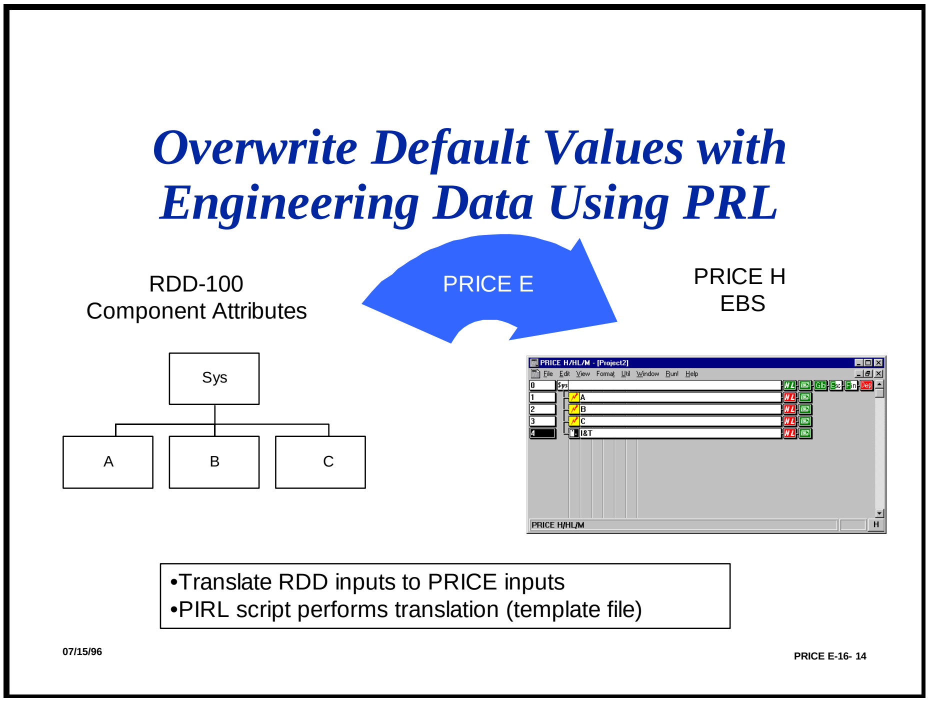
07/16/96

PRICE E-18- 5

Figure 3-7 Apply the CAFile

After the EBS has been created default values are determined through interrogation of the CA File. For instance, if electronic items exist in the target EBS, the values from the electronic item in the CA File are copied into the target EBS electronic items. If globals or other data types are attached to the electronic item in the CA File then they are also added to the target EBS electronic items.

3.3.3 Create PRICE inputs from RDD data using PRL



Overwrite Default Values with Engineering Data Using PRL

RDD-100
Component Attributes

PRICE E

PRICE H
EBS

Sys
A
B
C
PRICE H/HL/M - [Project2]
File Edit View Format Util Window Run! Help
Sys
A
B
C
I&T
PRICE H/HL/M

- Translate RDD inputs to PRICE inputs
- PIRL script performs translation (template file)

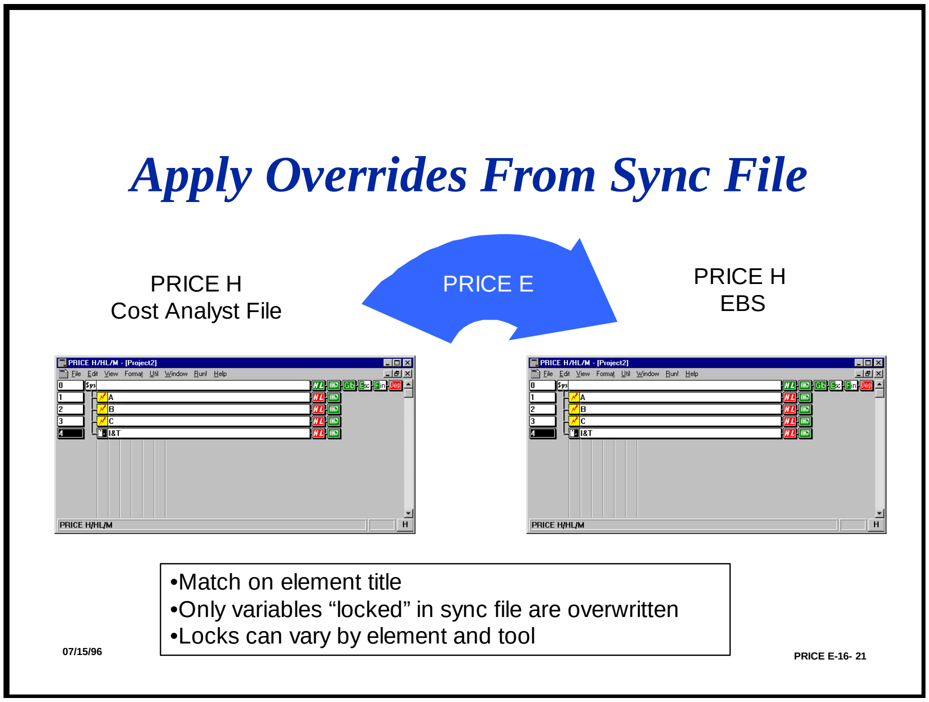
07/15/96

PRICE E-16- 14

Figure 3-8 Create PRICE Inputs from RDD Data

Next, the mapping functions in the PRL script are executed. These functions are contained within the PRL mapping section (detailed in the PRL documentation section). These functions are used to establish the rules for translating the attributes of an RDD component into the attributes of a PRICE element. If no mapping function exists for a PRICE attribute, then the default value, taken from the CA File, will remain in the target EBS for that element.

3.3.4 Apply the SyncFile overrides



Apply Overrides From Sync File

PRICE H
Cost Analyst File

PRICE E

PRICE H
EBS

PRICE H/HL/M - [Project2]
File Edit View Format Util Window Run! Help
Sys
A
B
C
I&T
PRICE H/HL/M
PRICE H/HL/M - [Project2]
File Edit View Format Util Window Run! Help
Sys
A
B
C
I&T
PRICE H/HL/M

- Match on element title
- Only variables “locked” in sync file are overwritten
- Locks can vary by element and tool

07/15/96

PRICE E-16- 21

Figure 3-9 Apply the SyncFile

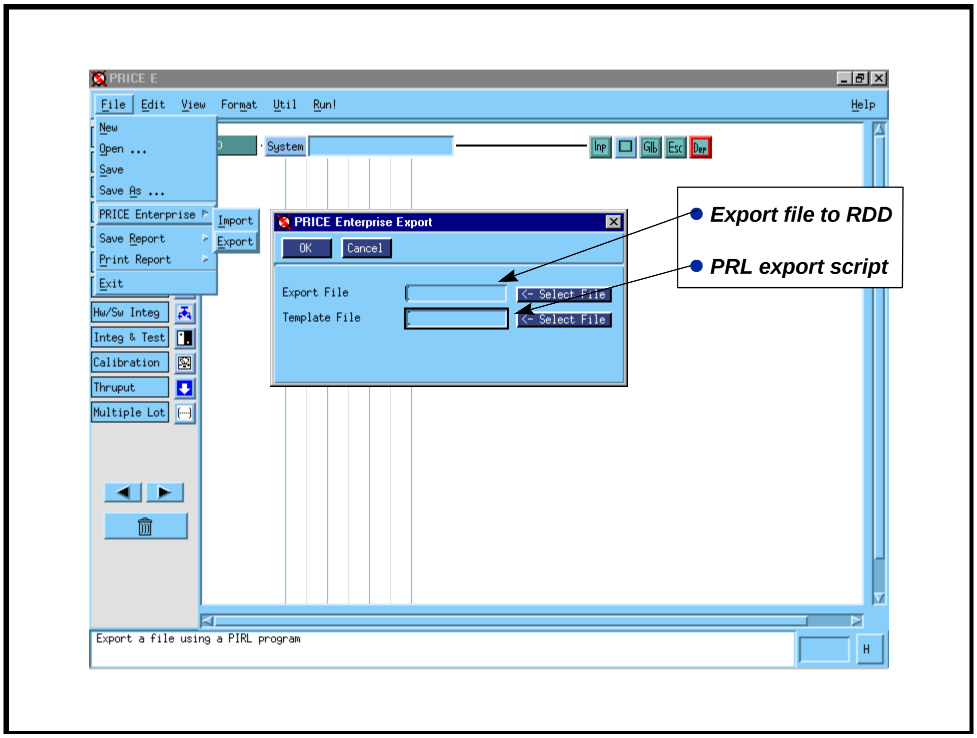
Finally, the Sync File is interrogated to determine if any overrides exist for PRICE attributes. The Sync File generally has the same structure and components as the target EBS. Inside the Sync File, individual attributes can be “locked.” When an attribute is locked, the PRL mapping function is said to be overridden. When the Sync File is processed, locked attributes are written from the Sync File into the target EBS. If an element from the Sync File has the same name as an element in the target EBS then the individual attributes of the Sync File element are examined for locks. If an attribute is found to be locked, then the attribute value from the Sync File is copied over the attribute value in the target EBS. The rules for creating Sync Files are explained in Section 4.

3.4 Running PRICE

At this point, a valid PRICE EBS has been created and populated by merging inputs from the engineer and the Cost estimator. If desired, you may work with the PRICE model using the graphical user interface described in the PRICE H and HL documentation that accompanied those products.

3.5 Exporting an PRICE cost file

To back-populate the cost data from PRICE into RDD choose the “File”, “Enterprise”, “Export” menu succession. The dialog box shown in Figure 3-10 will be presented. This dialog asks for two filenames. The first is the name of the output file to be created. This is the file that will be read into RDD. The second is the name of the PRL script that will extract the required cost information from PRICE and format the file for RDD.

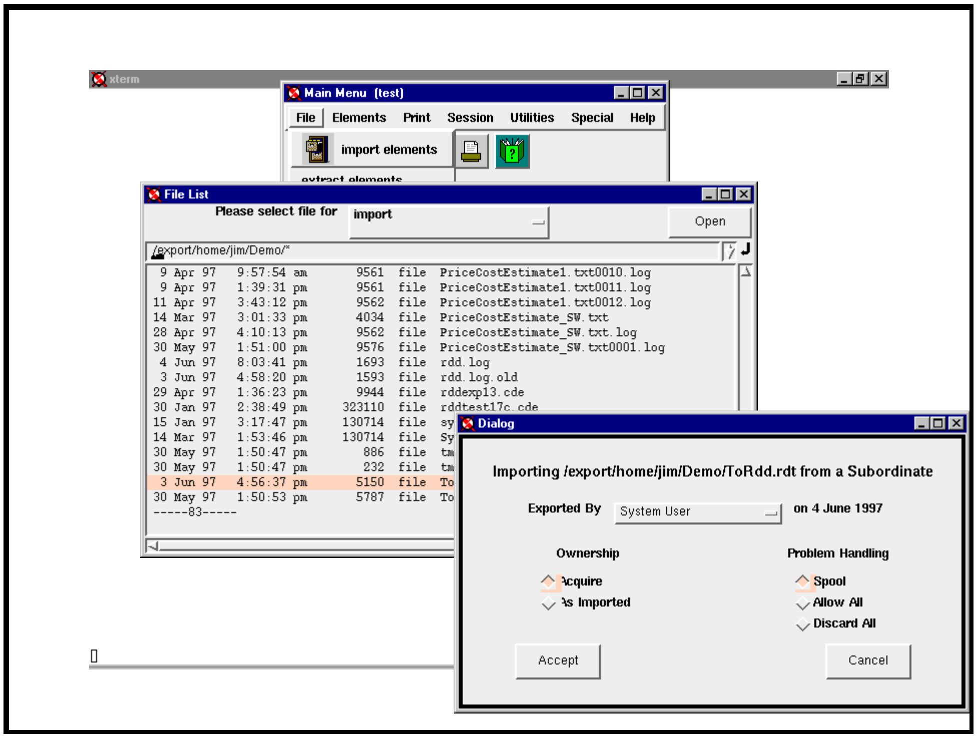


- **Export file to RDD**
- **PRL export script**

```
Export a file using a PIRL program
```

Figure 3-10 Exporting a PRICE Cost File

3.6 Importing a PRICE cost file into RDD



Please select file for: import

```
/export/home/jim/Demo/*
 9 Apr 97   9:57:54 am      9561  file  PriceCostEstimate1.txt0010.log
 9 Apr 97   1:39:31 pm      9561  file  PriceCostEstimate1.txt0011.log
11 Apr 97   3:43:12 pm      9562  file  PriceCostEstimate1.txt0012.log
14 Mar 97   3:01:33 pm      4034  file  PriceCostEstimate_SW.txt
28 Apr 97   4:10:13 pm      9562  file  PriceCostEstimate_SW.txt.log
30 May 97   1:51:00 pm      9576  file  PriceCostEstimate_SW.txt0001.log
 4 Jun 97   8:03:41 pm      1693  file  rdd.log
 3 Jun 97   4:58:20 pm      1593  file  rdd.log.old
29 Apr 97   1:36:23 pm      9944  file  rddexp13.cde
30 Jan 97   2:38:49 pm    323110  file  rddtest17c.cde
15 Jan 97   3:17:47 pm    130714  file  sy
14 Mar 97   1:53:46 pm    130714  file  Sy
30 May 97   1:50:47 pm       886  file  tm
30 May 97   1:50:47 pm       232  file  tm
 3 Jun 97   4:56:37 pm      5150  file  To
30 May 97   1:50:53 pm      5787  file  To
-----83-----
```

Importing /export/home/jim/Demo/ToRdd.rdt from a Subordinate

Exported By: System User on 4 June 1997

Ownership
- ● Acquire
- ○ As Imported

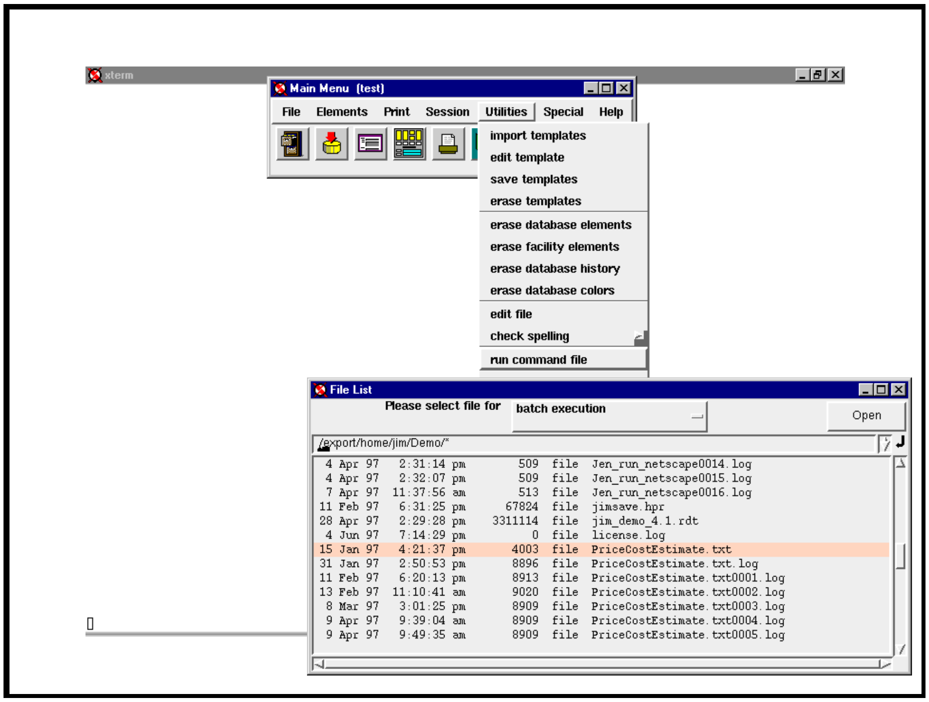
Problem Handling
- ● Spool
- ○ Allow All
- ○ Discard All

Figure 3-11 Importing a PRICE Cost File into RDD

To import the file that was exported from PRICE into RDD choose “File,” then “Import Elements” in succession and choose the name of the exported file from the dialog box that appears. A confirmation dialog box will appear. Select the “Acquire” mode and press the “Accept” button. This will cause the cost data for the component hierarchy to be captured and aligned with the hierarchy. This action populates the predicted cost attributes of the RDD cost elements. At this point, the cost exercise has been completed.

3.6.1.1.1.1 Automated Cost Estimate Operation

The operation described above exchanges data electronically, provides complete visibility of the process and access to the tools involved. Another option exists that insulates the engineer from the cost tool.



Please select file for batch execution

/export/home/jim/Demo/*

```
 4 Apr 97   2:31:14 pm      509  file  Jen_run_netscape0014.log
 4 Apr 97   2:32:07 pm      509  file  Jen_run_netscape0015.log
 7 Apr 97  11:37:56 am      513  file  Jen_run_netscape0016.log
11 Feb 97   6:31:25 pm    67824  file  jimsave.hpr
28 Apr 97   2:29:28 pm  3311114  file  jim_demo_4.1.rdt
 4 Jun 97   7:14:29 pm        0  file  license.log
15 Jan 97   4:21:37 pm     4003  file  PriceCostEstimate.txt
31 Jan 97   2:50:53 pm     8896  file  PriceCostEstimate.txt.log
11 Feb 97   6:20:13 pm     8913  file  PriceCostEstimate.txt0001.log
13 Feb 97  11:10:41 am     9020  file  PriceCostEstimate.txt0002.log
 8 Mar 97   3:01:25 pm     8909  file  PriceCostEstimate.txt0003.log
 9 Apr 97   9:39:04 am     8909  file  PriceCostEstimate.txt0004.log
 9 Apr 97   9:49:35 am     8909  file  PriceCostEstimate.txt0005.log
```

Figure 3-12 RDD-100 Automated Cost Estimate Operation

An automation script may be employed. To activate an automated cost estimate, choose the “Utilities,” then “Run Command File” in succession and select the “PriceCostEstimate.txt” file. This file will export the RDD file, import it into PRICE, run the model, export the cost file, and import it back into RDD where it will align with the exported component hierarchy. When using this method, the PRICE model executes without its graphical user interface and terminates on its own.

4. Creating and Maintaining the CA File and Sync File

4.1 Creating a CA File

The Cost Analyst or CA File may contain any of the following information:

- Default inputs for attributes not generated by the RDD/PRICE PRL template
- Labor and material splits, learning curves, AMS/OEM percentages
- Globals: organizational characteristics, economic basis of output, etc.
- Escalation: inflation rates and conversion factors
- Financial factors: labor rates and burdenings
- Deployment: number of units, time period

The CA File is used to populate the attributes that are not populated by the PRL template. This is considered to be almost exclusively information that resides in the functional domain of the estimator. The file should be created using the Windows™ product and uses the traditional ".hpr" file format. To create one, simply open a new project, add one element of each type from the tool box to the structure, modify the inputs as desired, and add global, financial factor, escalation and deployment tables where desired. A completed CA File will look something like this:

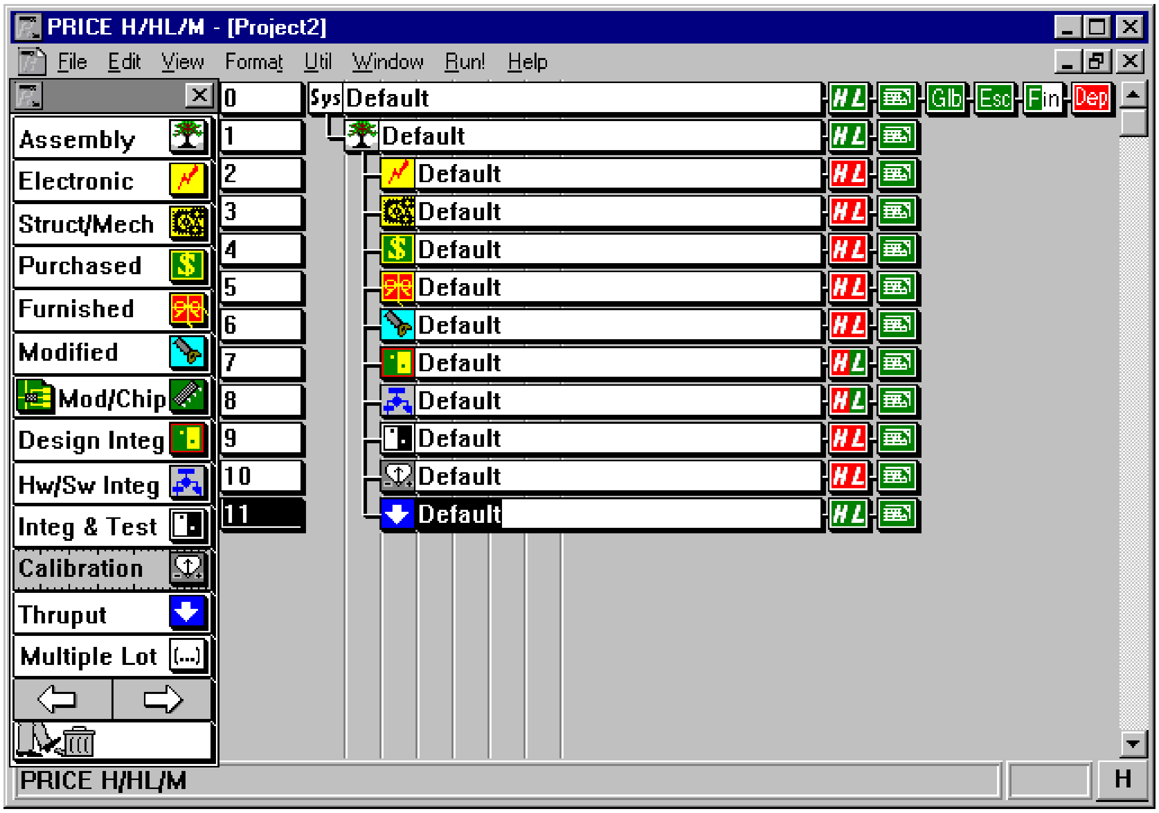


Figure 4-1 CAFile Example

The file does not need to be validated, although the individual parameters that are populated must each be within the valid range. This file will be used as the source of default data and be applied on an element-type basis. That means that during the creation of an estimating breakdown structure for an IDTC run, the PRL script will look into the CA File, find the element whose type matches that which you are creating, and populate it with the attributes from the CA File's matching element.

4.2 Creating a Sync File

To create a Sync File, simply save the contents of an Enterprise import as a normal PRICE ".hpr" file using the "File," "Save," or "Save As" menu selections. The name and location of the file (ideally defined in your IDTC work instructions) must be entered into the RDD image if future cost iterations are to include the Sync File automatically. The purpose of the Sync File is to override or augment the PRL translation and mapping functions for a PRICE attribute.

4.3 Override Screen

To access the override screen, open any input, global, escalation, financial factors or deployment screen; locate the "Override" button, and select it. The override screen, shown below, will appear. The override screen is organized into rows of attributes and columns of tools. That is because PRICE Enterprise is designed to interface with many non-cost and cost tools. For instance, you may use PRL scripts to interface PRICE with systems, electronics and mechanical design automation tools or spreadsheets, databases or proposal pricing systems. For each PRL script that you use, a corresponding tool column should exist in the Sync File. That way, multiple external tools can interact with a single Sync File that defines a powerful lattice of overlapping data sources for your estimates.

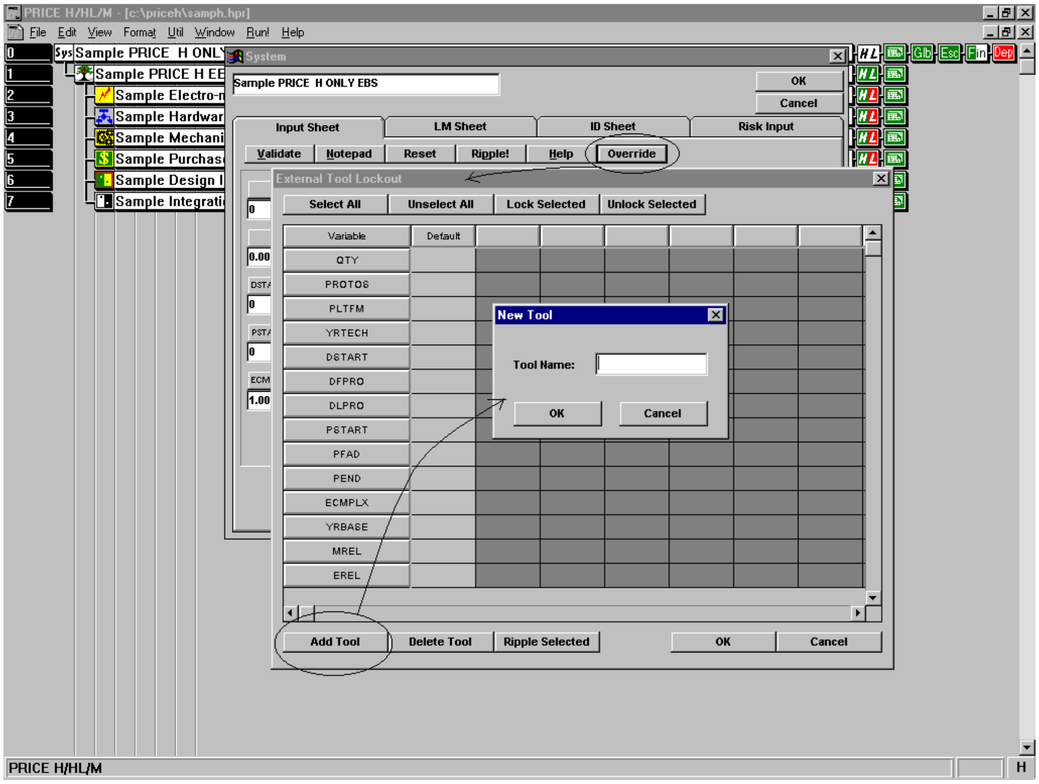
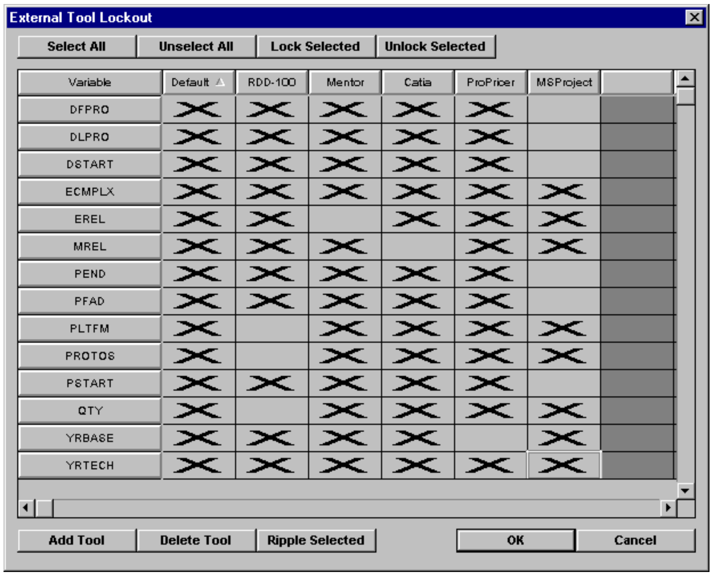


Figure 4-2 Override Screen

To add a tool column, select the "Add Tool" button on the override screen as shown above. Then, enter the identifying name for that tool. The same name will be entered into the RDD image. The RDD image allows you to identify the tool column in PRICE that contains the lock mask you wish to use.

In the example shown below, tool columns have been created for RDD, Mentor (an electronic CAD tool), Catia (a mechanical CAD tool), ProPricer (a proposal pricing tool), and MS Project. To lock a parameter against update from a tool,click at the intersection of the appropriate row and column. An "X" will appear at that location, indicating that the variable is now "locked."



Variable	Default	RDD-100	Mentor	Catia	ProPricer	MSProject	
DFPRO	X	X	X	X	X		
DLPRO	X	X	X	X	X		
DSTART	X	X	X	X	X		
ECMPLX	X	X	X	X	X	X	
EREL	X	X		X	X	X	
MREL	X	X	X		X	X	
PEND	X	X	X	X	X		
PFAD	X	X	X	X	X		
PLTFM	X		X	X	X	X	
PROTOS	X		X	X	X	X	
PSTART	X	X	X	X	X		
QTY	X		X	X	X	X	
YRBASE	X	X	X	X		X	
YRTECH	X	X	X	X	X	X	

Figure 4-3 Example of Lock Mask

In the example above, the lock mask allows the scheduling tool to modify the schedule dates; the CAD tools to modify their associated MTBF multipliers (electronic or mechanical, as the case may be); the system tool (RDD) to modify the PLTFM, WTY, and PROTOS parameters; the Pricing tool to control the economic basis (YRBASE), and the estimator to control the YRTECH parameter (via the CA File/Sync File combination). Once a tool column is added within a Sync File, it appears at each element and for each input type (inputs, globals etc.)

5. PRICE Rule Language (PRL)

5.1 PRL Overview

PRL is a proprietary interpreted language used for importing and exporting data to and from the PRICE models.

5.1.1 Import

A PRL import involves reading data from an external tool, making relationships between the data and the PRICE Models, and building an EBS.

5.1.1.1 Header Section.

This section includes the `CDE_Import_Template` key, user comments, a description of the data to import, and CA and Sync Files.

5.1.1.2 Input Names Section.

This section lists the available inputs in the foreign file including the input name, data type, and alias. Because some of the input names can have more than one word it is easier to have an alias for later references. For example, if a foreign file has an input name called "Component Name," it would be simpler to call it CompName as an alias.

5.1.1.3 Mapping Section.

This section contains all of the functions that will map one or more foreign inputs to a PRICE input.

5.1.1.4 Node Creation Section.

This is an optional section. It adds and modifies nodes.

5.1.2 Export

An export PRL program has a simpler syntax than an import PRL program. One header section and 10 format sections comprise an export PRL program. The header section is used for generating comments in the report The format sections describe how the report should be generated.

5.2 PRL for Importing

5.2.1 Header Section.

The first five lines of the PRL program make the header section.

5.2.1.1 Example Header Section.

The following example shows the header section of an Import PRL program.

```
CDE_Import_Template Version 1
1 2 0
/project/cafile/proj025.hpr
/project/sync/proj025.hpr
5
```

5.2.1.2 Line 1: Header key and Comments.

The first line of a PRL import file must start with the string **`CDE_Import_Template`**. The rest of the line is available for optional user comments.

5.2.1.3 Line 2: Data Read Settings.

The second line contains three values for describing the form of the data being imported from an external tool.

5.2.1.3.1 Node Options.

When PRICE Enterprise starts to read an input file it reads a chunk of information that will lead to creating an EBS node in the PRICE models. Node Options tell PRICE Enterprise how to read the input file. Three node operations are specified for PRICE Enterprise, however, only Node_By_Row is implemented.

5.2.1.3.1.1 0: Node_By_Column.

This is not implemented.

5.2.1.3.1.2 1: Node_By_Row.

Import RDD-100 files and other tab-delimited files in the following format.

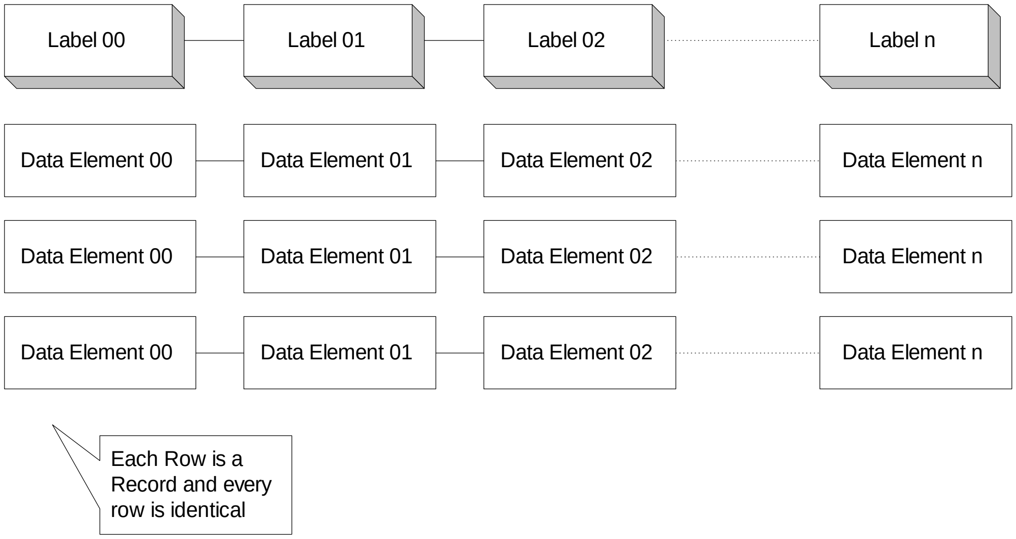


Figure 5-1: Node By Row File Format.

5.2.1.3.1.3 2: Node_By_Block.

This is not implemented.

5.2.1.3.2 Number of Lines Ignored.

If the Node Option is Node_By_Row, this input instructs PRICE Enterprise to ignore the number of lines in the input file. This would typically be comments.

5.2.1.3.3 Number of Columns Ignored.

If the Node Option has a value Node_By_Column, this input instructs PRICE Enterprise to ignore this number of columns.

5.2.1.4 Line 3: Cost Analyst File Name.

If a number rather than a file name is found on this line, PRICE E will get the file name from the import data file using that number as the line number in the data file. If a file name is found, it will be used for the Cost Analyst File. Note that the PRICE Enterprise import dialog could specify a Cost Analyst File and override both of these cases.

5.2.1.5 Line 4: Sync File Name.

This line is similar in operation to the previous line for a Cost Analyst File name. If a number rather than a file name is found on this line, PRICE E will get the file name from the import data file using that number as the line number in the data file. If a file name is found, it will be used for the sync file. Note that the PRICE Enterprise import dialog could specify a Cost Analyst file and override both of these cases.

5.2.1.6 Line 5: Lock ID.

This instructs PRICE Enterprise as to which lock ID to use when synchronizing the import data with the Sync File. If a number rather than a lock ID name is found on this line, PRICE E will get the lock ID from the import data file using that number as the line number in the data file. If a lock ID is found, it will be used for the sync file.

5.2.2 Input Names Section.

This section lists all the relevant inputs in the input file. Every line in this section starts with the word “Inputs” followed by a colon and a list of input names. In front of each input name comes a type of string or double inside a pair of parenthesis. The input name is a quoted text. An optional variable followed by a colon can be put between the type and the input name as an alias. Any C Language acceptable variable name may be used for the alias. This alias is helpful in the Mapping Section when it becomes necessary to reference an input whose name consists of more than one word. There is no limit on the number of lines in this section.

```
Inputs: (string) CompName: "Component Name"
Inputs: (string) CompType:"Component Type"
Inputs: (double) length: "Component Length"
```

With an alias length for "Component Length," it would be simpler in the mapping section to have a statement like

```
VOL = length * width * depth;
```

instead of

```
VOL = Component Length * Component Width * Component Depth;
```

In the case that the import file does not provide input names, the input names section should use a dollar sign ($) followed by the field number to describe the input names. For example, with the following input names section in a PRL program, it indicates that the first column (for a Node_By_Row file) is the Component Name, the second column is the Component Type, and the third column has the length input.

```
Inputs: (string) CompName: $1
Inputs: (string) CompType: $2
Inputs: (double) length: $3
```

An alias is mandatory for an input like this because the dollar sign and the number together do not compose a valid variable later in the mapping and node creation sections.

5.2.3 Mapping Section.

All the mapping functions reside in this section. The user can create as many functions as needed. Every mapping function starts with a line that has a form like the following line

```
Start_Map: variable
```

and ends with a line End_Map. A variable that identifies a PRICE input should be specified on the Start_Map line so that when PRICE Enterprise finishes this mapping function it knows where to put the result. A return line should be used right before the End_Map line.

The result of the expression after return will then be copied to the PRICE variable specified on the Start_Map line. Between the Start_Map and End_Map lines are PRL statements that look very much like C Language syntax.

PRICE Enterprise goes through all the mapping functions *each time* it reads a piece of information from the import file. This could be a row, a column or a block. By going through the mapping functions PRICE Enterprise will create one PRICE node on the EBS screen. A valid mapping function has the following syntax:

```
Start_Map: variable
   pirl-statement1
   pirl-statement2
   …
   return expression;
End_Map
```

5.2.4 Node Creation Section.

When importing a file, PRL allows the user not only to map the information in the foreign file to the PRICE variables, but also to create nodes. As soon as PRICE Enterprise finishes the mapping section it creates a PRICE EBS structure based on the input information and the mapping functions.

If for any reason the user feels the information from the input file is not sufficient, the user can add or change nodes in the EBS that has been created. For example, if the input file does not supply integration information and the user wants the PRICE model to estimate integration cost as well, the EBS can be changed by the user.

The user can create as many node creation functions as necessary. For every node creation function in the PRL program PRICE Enterprise will automatically try to add a node right after each node in the already created EBS. The user may choose to direct PRICE Enterprise to create a node only if a specific condition holds.

For example, the user may direct PRICE Enterprise to add a hardware-software integration node only after any electronic nodes with an HSINT value greater than 0 or to add an integration-and-test node after the assembly nodes. The only mandatory value that PRICE Enterprise needs to complete a node creation request is the Mode Type. Please see Appendix A for PRICE Mode Types. Thus, there should always be an assignment statement in the node creation function to assign a mode value as in the next line.

```
PRICE<-Mode = 52;
```

A node creation function starts with a line Start_Node and ends with a line End_Node. Any PRL statements described in the mapping section can be used inside the node creation function. When PRICE Enterprise finds a node creation function it starts from the first node on the already created EBS and executes each and every node. Thus, for every node on the EBS structure it will execute the node creation function once. If it finds that a valid mode is given after executing the function it adds a node after the current node.

If the function gives an indent number PRICE Enterprise will build a parent-child relationship, but only when the indent number is equal to or one level greater than the current node. So, if the indent number is not given or invalid, PRICE Enterprise will add the node as a sibling of the current node. If the indent number is one level greater than the current node and the current node is an assembly node, PRICE Enterprise will add a node as a child of the current node.

The following node creation function asks PRICE Enterprise to add a hardware/software integration (Mode 52) after a Mode 1 (electronic node), if the HSINT value in the electronic node is greater than 0.

```
Start_Node
   if (PRICE->Mode == 1 && PRICE->HSINT > 0.)
       PRICE<-Mode = 52;
   endif
End_Node
```

If a node creation function is created to add an integration and test node (Mode 5), a design integration node (Mode 51), or a hardware software integration node (Mode 52), PRICE Enterprise will automatically add the new node to the assembly node as the last child.

As a final note please be warned that you could create an endless PRL program by doing something like the following:

```
Start_Node
    // Do not do this.
    if (PRICE->Mode == 1)
        // Create another node.
        PRICE<-Mode = 1;
    endif
End_Node
```

This example is a reminder to make sure that you write code that produces a finite number of nodes.

5.2.5 Cost Analyst File

The role of the Cost Analyst File is to provide default cost estimation data for an EBS. This file fills input values if they are not specified by the PRL file or Sync File. The Cost Analyst File is searched by mode. The first instance of a mode will be used to set the default values.

5.2.6 Sync File

The Sync File locks data and overrides values set by the PRL file and the Cost Analyst File. You can override the Sync File if you modify a value in the node creation section. The Sync File is searched by title and mode. The first instance of a matching mode will be used to set the override. If global tables are attached to the match in the Sync File, then an attempt will be made to override the global tables in addition to the node. *Please be aware that this only works for values set in the mapping section!*

5.2.7 PRL Syntax

Much of the syntax of PRL is based on the C Language. Beware of cases where the implementation of PRL differs.

5.2.7.1 PRL Types.

There essentially are two types in PRL: STRING_TYPE and VALUE_TYPE. PRL uses implicit typing for variables. VALUE_TYPE numbers are interpreted as floating point. The exception is the integer PRICE Enterprise variables.

5.2.7.2 PRL Variables.

As with any high level programming language you can use variables in PRL to store values, but unlike most you do not have to define or declare the variables except for the Input Variables. PRICE Enterprise uses an implicit way of interpreting the types of variables. The first time that a variable shows up usually allows PRICE Enterprise to determine the type. If the variable month shows up for the first time in a statement like the following

```
month = 12;
```

PRICE Enterprise will set a double type for the variable month. There will be a type mismatch error if later on in the PRL program another statement like the following shows up.

```
strcpy(month, "December");
```

If a variable shows up for the first time and PRICE Enterprise is unable to determine the type, you will get a type mismatch error as well. This is illustrated in the following PRL statements.

```
A = 50.0 + NoTypeVariable;
```

5.2.7.2.1 Input Variables.

These variables are used to store the data imported from the foreign file. All the input variables must be defined in the input names section. When PRICE Enterprise reads a node of information from the import file it automatically puts the values into these variables. You can use the values by referencing the input variables.

```
Inputs: (char) title: "Project Name"
Inputs: (double) weight: "Component Weight"
```

PRICE Enterprise will treat title and weight as input variables. You can use them in your mapping functions. The following lines are valid PRL statements:

```
strcpy (myvariable, title);
metricWt = weight * 0.4536;
```

As described above you should not try to change the value of an input variable. The following statements would be considered a syntax error. This prevents a user from accidentally using the same variable name as an input name.

```
strcpy(title, "New Name"); // Copy a string to an input var. -> syntax error
weight = weight * 0.4536;  // Assign a value to an input variable. -> syntax error
```

5.2.7.2.2 User Variables.

These are user created variables. The user can use any legal variable name in the PRL program for calculation purposes or for string operations.

```
vol = length * width * height;
```

In the example above, “vol” is a user variable while length, width, and height could be input variables if they are defined in the input names section.

5.2.7.2.3 PRICE Variables.

PRICE Variables refer to the existing EBS nodes. When you import a file PRICE Enterprise will build an EBS structure based on the information in the import file and the mapping functions. PRICE Enterprise will then check if the node creation section exists in the PRL program. If it finds a node creation section, PRICE Enterprise will loop through the entire EBS nodes for each node creation function. PRICE Variables refers to the node that PRICE Enterprise is currently on. You can use the values stored in these variables for calculation or condition checking.

Any variable from the PRICE Variables list prefixed by "PRICE->" is a PRICE Variable.

A special form of PRICE Variables, called PRICE Parent Variables is also available, which provides a vehicle to access a parent node. The syntax of such variables is the same as the regular PRICE Variables except that the prefix is "PRICE->PARENT->" instead. The following are valid PRICE Parent Variables:

```
PRICE->PARENT->QTY, PRICE->PARENT->QTYNHA
```

PRICE Parent Variables provide a convenient way to store the information of a child node to its parent especially when adding an integration node. For example, you may want to store the source lines of code of a software node in its parent node so that later when you add a hardware/software integration node you will be able to use this information. The following two node creation functions demonstrate this capability.

```
// Store hsint information in the parent nodes
Start_Node
   if (PRICE->HIN_Mode == 62) // if it is an assembly node
      PRICE->PX_Import_Int1 = 0;      // use PX_Import_Int1 to store SLOC
   else if (PRICE->HIN_Mode == 80)   // if it is a software node
      PRICE->PARENT->PX_Import_Int1 += PRICE->SIN_SLOC_1;
   endif
End_Node

// Add a Hardware/Software Integration Node to assembly nodes
Start_Node
   // if it is an assembly node and has SLOC
   if (PRICE->HIN_Mode == 62 && PRICE->PX_Import_Int1)
      strcpy(title, "HW/SW Int -");
      strcat(title, PRICE->HIN_Title);
      PRICE<-HIN_Mode = 52;          // add a HW/SW integration node
      PRICE<- SLOC = PRICE->PX_Import_Int1;  // use the stored information
      PRICE<-FRAC = 0.2;
      PRICE<-APPL = 5.0;      // should get it from software children
      PRICE<-CPLXM = 1.0;
      PRICE<-HIN_LANG = 21;   // should get this from software children
   endif
End_Node
```

5.2.7.2.4 Node Creation Variables.

In the node creation function the user must provide data for the node to be created by PRICE Enterprise. Node creation variables are used to store this data before PRICE Enterprise actually adds it to the EBS. Any variable from the PRICE Variables list prefixed by "PRICE<-" is a node creation variable.

The following node creation function will add a hardware/software integration node after every electronic or purchased node if its HSINT value is greater than 0.

```
Start_Node
   if ( (PRICE->HIN_Mode==1 || PRICE->HIN_Mode==3) && \
   PRICE->HIN_HSINT > 0.0)
      print  "Added H/S int";
      PRICE<-HIN_Mode = 52;
      PRICE<-HIN_SLOC = PRICE->HIN_HSINT  *  100000;
      // ALGOL -- This is a comment
      PRICE<-HIN_LANG = 2;    // language id, 2 is for ALGOL
   endif
End_Node
```

Normally the user would use PRICE variables and node creation variables in the node creation section only.

5.2.7.2.5 PRICE Variables versus Node Creation Variables.

The difference between a PRICE variable and a node creation variable is that a PRICE variable has a value in the current EBS node that PRICE Enterprise is on while a node creation variable lets the user store a value in the node which will be added to the EBS by PRICE Enterprise. If you have the following node creation function:

```
Start_Node
   if (PRICE->Mode == 1)
      PRICE<-Mode = 51;
   endif
End_Node
```

PRICE Enterprise will add a node after every Mode 1 (electronics) node. By default PRICE Enterprise adds a node at the same level as the node that it is currently on. So PRICE Enterprise adds a sibling unless the node creation function uses a PRICE<- indenture variable to enforce it otherwise. But if you have the following node creation function,

```
Start_Node
   if (PRICE->HIN_Mode == 1)  // if Electronic node
      PRICE->HIN_Mode = 3;    // Change it to Purchased Node
   endif
End_Node
```

PRICE Enterprise will convert every electronic node to a purchased node. Thus, whenever you assign a value to a PRICE variable you actually change the input data on the existing EBS. You can not change the output data of an existing node. Output data will always be generated by the model. So the following node creation function has no effect on the EBS.

```
Start_Node
   if (PRICE->HIN_Mode == 1)  // if Electronic node
      PRICE->HOUT_Drafting_TOT *= 2.0; // double drafting cost – won’t work
   endif
End_Node
```

The user does not have to use a node creation function to create a node. Users can also use a node creation function to modify the input data.

5.2.7.2.6 PRICE Enterprise Variables

There are 30 PRICE Enterprise Variables that are not used for calculation. Some import files contain information that the PRICE Model does not need. The user can put the information in these PRICE Import Variables and use it when exporting to another system. These variables are:

```
PX_Import_Str1, PX_Import_Str2, …, PX_Import_Str10
PX_Import_Int1, PX_Import_Int2, …, PX_Import_Int10
PX_Import_Flt1, PX_Import_Flt2, …, and PX_Import_Flt10
```

As you can see from their names, the PX_Import_Str group is used to store strings, the PX_Import_Int group stores integers and the PX_Import_Flt group stores floating point values.

The first string variable, PX_Import_Str1, is reserved for PRICE Enterprise to store the parent name of a node. Thus, you should always have a mapping function that puts the parent name of a node into Import_Str1, otherwise PRICE Enterprise would create a single level EBS with every node created as a child of the system node.

The following is how the PRICE Enterprise Variables are used for RASSP RDD-100 importing:

5.2.7.2.6.1 Strings.

PX_Import_Str1 - Parent of a component
PX_Import_Str2 - Duplicate component flag (Yes or No)

PX_Import_Str3 - Modified design source flag (yes or no)
PX_Import_Str4 - Technology maturity (state of the art, leading edge, mature, obsolete)
PX_Import_Str5 - Life cycle parameter name
PX_Import_Str6 - RMA element name
PX_Import_Str7 - Cost element name
PX_Import_Str8 - External tool file name
PX_Import_Str9 -
PX_Import_Str10-

5.2.7.2.6.2 Integers.

PX_Import_Int1 - Total production quantity for all common duplicate components
PX_Import_Int2 - Actual production quantity for a particular duplicate component
PX_Import_Int3 - Software flag (0 if not software, 1 if software)
PX_Import_Int4 - Source Lines of Code (SLOC) for software components
PX_Import_Int5 - Parent Need I&T node ? (0: No, 1: Yes)
PX_Import_Int6 -
PX_Import_Int7 -
PX_Import_Int8 -
PX_Import_Int9 -
PX_Import_Int10 -

5.2.7.2.6.3 Floating Point.

PX_Import_Flt1 - Actual prototype quantity for a particular duplicate component (is really a float)
PX_Import_Flt2 - Software application difficulty * Source Lines of Code (SLOC)
PX_Import_Flt3 - Quantity next higher assembly
PX_Import_Flt4 -
PX_Import_Flt5 -
PX_Import_Flt6 -
PX_Import_Flt7 -
PX_Import_Flt8 -
PX_Import_Flt9 -
PX_Import_Flt10 -

5.2.7.2.7 Arrays.

Arrays are not implemented in PRL.

5.2.7.3 Assignment Statements.

An assignment statement is used to set a variable's value. The left hand side of an assignment statement must be the variable being set. The right hand side of the equal sign is an expression, which could be a variable, an arithmetic expression, or a function. The following are all valid assignment statements:

```
     a = b + 25 / (c + d);
     x = sin(3.1415926) - 0.35;
     day = (month == 4 || month == 6 || month == 9 || month == 11) ? 30 : 31 ;
```

5.2.7.4 If Statements.

If statements support conditional decisions. It starts with an *if* line and ends with *endif* line. An optional *else* line or else if line can be used in conjunction with the *if* statement. Any valid PRL statement can be used to create a conditional decision. Valid *if* statements would be like the following:

```
if ( month == 2 )
     if (leapYear)
          day = 29;
     else
```

```
          day = 28;
     endif
else if (month == 4 || month == 6 || month == 9 || month == 11)
     day = 30;
else
     day = 31;
endif
```

5.2.7.5 String Functions.

Four types of string operation are currently supported: *strlen*, *strcmp*, *strcpy*, and *strcat*. These string functions work the same way as in the C Language. PRICE Enterprise allocates memory for string variables. If the size of a string variable changes, PRICE Enterprise will automatically reallocate memory accordingly.

5.2.7.5.1 Strlen().

This function calculates the length of a string;

5.2.7.5.2 Strcmp().

This function compares two strings.

5.2.7.5.3 Strcpy().

This function copies a string.

5.2.7.5.4 Strcat().

This functions concatenates a string with another string.

5.2.7.5.5 Examples.

```
length = strlen(name);
strcpy (internetaddress, name);
strcat (internataddress, "@pricesys.com");
```

5.2.7.6 Print Statements.

A print statement can be used in PRL to write the result of an expression to standard output. The user can use it for debugging a PRL program. Some examples are:

```
print a+b;
print "mapping started";
```

5.2.7.7 Exit Statements.

The exit statement will stop the execution of a PRL file. This is helpful for error conditions where it would be better to not complete the processing.

```
// Error condition found.
exit;
```

5.2.7.8 Math Functions.

Many of the math functions found in the C Language are supported in PRL.

5.2.7.8.1 sin(double a).

Returns radian trigonometric sine.

5.2.7.8.2 cos(double a).

Returns radian trigonometric cosine.

5.2.7.8.3 tan(double a).

Returns radian trigonometric tangent

5.2.7.8.4 asin(double a).

Returns radian arc sine.

5.2.7.8.5 acos(double a).

Returns radian arc cosine.

5.2.7.8.6 atan(double a).

Returns radian arc tangent.

5.2.7.8.7 sinh(double a).

Returns radian hyperbolic sine.

5.2.7.8.8 cosh(double a).

Returns radian hyperbolic cosine.

5.2.7.8.9 tanh(double a).

Returns radian hyperbolic tangent.

5.2.7.8.10 exp(double a).

Returns e^a.

5.2.7.8.11 log(double a).

Returns the natural logarithm of a.

5.2.7.8.12 log10(double a).

Returns the base 10 logarithm of a.

5.2.7.8.13 pow(double a, double b)

Returns a^b.

5.2.7.9 Comments.

Comments can only be used in the mapping section and the node creation section. They cannot be used in the header or input names sections. Anything following a double-slash (//) is ignored by PRICE Enterprise and is considered a comment.

5.2.8 System Functions.

There are some special system functions you can use in PRL. All the special system functions start with an @ sign and generally do not require any parameters. As an example @date is a system function that would generate a string such as May-31-1996.

```
strcpy (PRICE<-title, "Created on - ");
strcat (PRICE<-title, @date);
```

5.2.8.1 @date.

Returns a date string.

5.2.8.2 @time.

Returns a time string.

5.2.8.3 @LockID.

Returns the Lock ID string used.

5.2.8.4 @CAFile.

Returns a CA File name string.

5.2.8.5 @Sync File

Returns a Sync File name string.

5.2.8.6 @FillFromH.

Generates HL data using H data and returns 1 on success, 0 on failure.

5.2.8.7 @PropagateDeploymentFirstYear.

Propagates ED and OTF values on the pricehl deployment table from the first year to the rest of the deployment years.

5.2.8.8 @InitializeFunctionParameters.

Initializes function parameters to zero.

5.2.8.9 @MCPLXEGenerator.

Calls the MCPLXE generator. Note that @InitializeFunctionParameters needs to be initiated prior to using this special function. The requirements for using the PRICE MCPLXE generator are to set up a 7x 7 matrix of values as seen on the MCPLXE generator input sheet, as well as Quality Adjustment, Platform, and Density Adjustment. PRICE E provides a list of variables for the user to assign values. These values will be used when you initiate a system function such as the @MCPLXEGenerator, which requires the following variables to be assigned a value:

PRICE->FUNC_Param1,
PRICE->FUNC_Param2,
...
PRICE->FUNC_Param52

5.2.9 Mode Morphing

PRL has a great deal of power to build EBSs. This power can cause problems for the careless user. Coupling this with minimum error checking it is possible to unintentionally build EBSs that were never meant to exist. You might set input variables that do not belong to a particular node. PRL will not challenge this, but you will get unpredictable results. For example you could incorrectly set WT for a purchased item. One area of confusion related to this problem is the ability to morph a node from one node to another in the node creation section. You can get into trouble if you are not careful. If you adhere to the following instructions you should be all right.

- ❑ Copy all input data to temporary variables, if you want to use the data after the morph.
- ❑ Copy all output data to temporary variables, if you want to use the data after the morph.
- ❑ You do not have to backup the PRICE Enterprise variables. This is autonmatic on a mode morph.
- ❑ Change the mode. PRICE->HIN_Mode = SomeMode;
- ❑ Set the backed up data, if it makes sense for the current mode.

5.2.10 Software Languages

For software language data to be set up correctly you must start by assigning the SIN_Language_1 variable first and *then* set the appropriate values. Proceed in order with the appropriate languages if required. The order is essential for this data to be set correctly. This is a limitation of the language. Note that only four software languages are available per software node.

5.2.11 Import File Text Self Access

Occasionally, a PRL program needs access to a line in its own program. An example would be when retrieving a Cost Analyst or Sync File name to pass back to the host tool. You can access any line in the import file using the following expression:

```
#<Field Number> @ #<Line Number>
```

or

```
#<Line Number>
```

If the second line of an import file is

```
price template - rddtest.cde
```

the following PRL code will copy the string rddtest.cde to the variable szFileName

```
strcpy(szFileName,#4@#2);
```

and the following code will copy the entire line to the variable szSecondLine

```
strcpy(szSecondLine,#2);
```

5.2.12 Import Example

The following is a complete example of importing a file into PRICE E. The import file is a Microsoft Excel file saved as a tab-delimited text file. The following PRL program can be used to import the file.

```
CDE_Import_Template
1      1    0
/project/cafile/proj025.hpr
/project/sync/proj025.hpr
RDD-100
Inputs: (string)CompName:"Component Name"
Inputs: (string)CompType: "Component Type"
Inputs: (double) Quantity: "Component Quantity"
Inputs: (double) "Technology"        (string)"Owner"
Inputs: (double)"Weight"   (double)"Length"   (double)"Width"
Inputs: (double)"Height"

// Mapping Section
Start_Map: HIN_Title
   return (CompName);
End_Map

Start_Map: HIN_Mode
   val = -1;   // if none of  the following
   if (!strcmp(CompType, "Assembly"))
      val = 62;
   else if (!strcmp(CompType, "Electronic"))
      val = 1;
   else if (!strcmp(CompType, "Mechanical"))
      val = 2;
   else if (!strcmp(CompType, "Purchased"))
      val = 3;
   else if (!strcmp(CompType, "Furnished"))
      val = 4;
   else if (!strcmp(CompType, "Modified"))
      val = 6;
   endif
   return (val);
End_Map

Start_Map: HIN_QTY
   return Quantity;
End_Map

Start_Map: HIN_YEARTECH
   return Technology;
End_Map

Start_Map: PX_Import_Str1     // always store parent’s title
   return Owner;
End_Map

Start_Map: HIN_WT
   return Weight;
End_Map

Start_Map: HIN_VOL
   return (Length * Width * Height);
End_Map
```

The following EBS will be created by PRICE Enterprise after it executes the PRL program with the import file as its input.

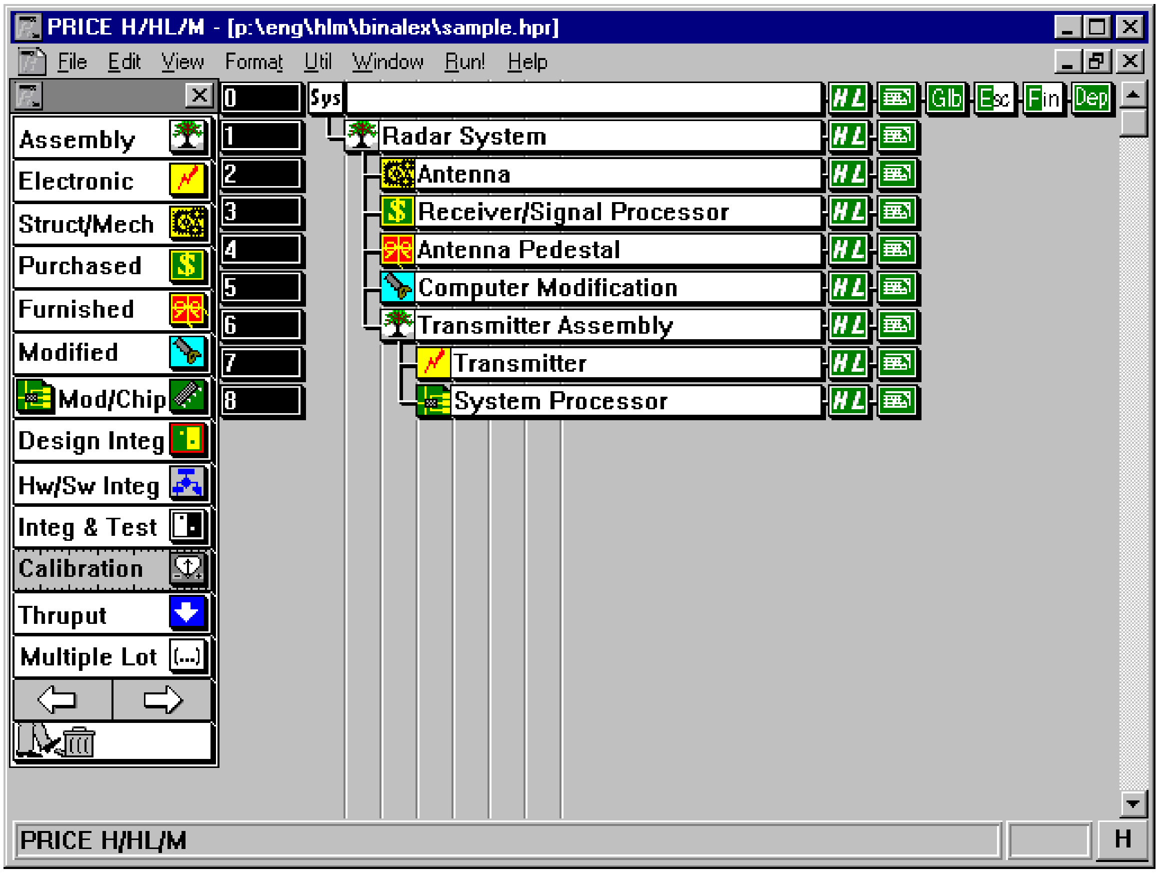


Figure 5-2 EBS After Import

5.3 PRL for Exporting

The syntax of a PRL program for exporting is quite different. Because the main purpose of an export PRL is to generate an output file. PRICE Enterprise automatically print the results of any expressions it finds in the PRL program. The print statement described in the import PRL is not necessary. So, if you have an expression like

```
2+3+5
```

PRICE Enterprise will print out 10 as the result of the expression.

5.3.1 Header Section

The header section must begin at the first line of the export template and must start with the line *Header_Start* and end with the line *Header_End*. Anything between these two lines will be output to the file as is, except the special system functions such as *@date* which will be replaced by the string that the special function generates.

5.3.2 Format Section

You can use up to 10 format sections in an export PRL program. A format section starts with the line Section_Start and ends with the line Section_End. Every line between these two lines is examined by PRICE Enterprise. Any expressions found by PRICE Enterprise will be executed and the result will be printed. For every format section, PRICE Enterprise loops through every node that is tagged to execute the format section. An expression can be a quoted text, a math expression, a math function, or a special system function such as @date. If you have the following format section,

```
Section_Start
if (PRICE->HIN_Mode == 1)
       " Electronic Item" PRICE->Title
else if (PRICE->HIN_Mode == 2)
       "Mechanical Item" PRICE->HIN_Title
endif
Section_End
```

PRICE Enterprise will go through every tagged node and print the node type and its title if it is a Mode 1 (Electronic) or Mode 2 (Mechanical) node. You can concatenate expressions in one line such as

```
"Created on" @date "at" @time "by John PRICE."
```

but you should put it in the header section if you want a line to be printed just once and on the top of the output file. However, since anything in the header section will be printed the way it is (except the special functions), the line should be written like the following example instead.

```
Created on @date at @time by John PRICE.
```

You can use PRICE variables in an expression such as the following:

```
"Electronic Weight: " PRICE->WT - PRICE->WS
```

But, there should be no node creation variables in an export PRL program. You can have an assignment statement in your format section which PRICE Enterprise will execute, but no value will be printed. So the user can insert user variables in an assignment statement without PRICE Enterprise printing anything. The following line will not cause PRICE Enterprise to print anything.

```
A = pow (2.0, 3.0);
```

but the following line (the single letter *A*)

```
A
```

will cause PRICE Enterprise to print the value of *A*. You can put a format after an expression to control the output position such as %d, %10.2f, %12s.

5.3.3 Export Example

With an EBS as in page 12 the following export PRL program can be used to generate an output file.

```
Header_Start
From PRICE H on @date at @time
Header_End
Section_Start
printcost = 1;
if (PRICE->HIN_Mode == 1)
   strcpy(nodeType, "Electronic");
else if(PRICE->HIN_Mode == 2)
   strcpy(nodeType, "Mechanical");
else if (PRICE->HIN_Mode == 3)
   printcost = 0;
   strcpy(nodeType, "Purchased");
else if(PRICE->HIN_Mode == 4)
   printcost = 0;
   strcpy(nodeType, "Furnished");
else if (PRICE->HIN_Mode == 5)
   strcpy(nodeType, "Integration and Test");
else if (PRICE->HIN_Mode == 6)
   strcpy(nodeType, "Modified");
else if (PRICE->HIN_Mode == 7)
   strcpy(nodeType, "Calibration");
else if (PRICE->HIN_Mode == 8)
   strcpy(nodeType, "Thruput");
else if (PRICE->HIN_Mode == 9)
   strcpy(nodeType, "Multiple Lot");
else if (PRICE->HIN_Mode == 33)
   strcpy(nodeType, "Detailed Cost");
else if (PRICE->HIN_Mode == 51)
   strcpy(nodeType, "Design Integration");
else if (PRICE->HIN_Mode == 52)
   strcpy(nodeType, "Hardware/Software Integration");
else if (PRICE->HIN_Mode == 60)
   strcpy(nodeType, "System");
else if (PRICE->HIN_Mode == 62)
   strcpy(nodeType, "Assembly");
else
   printcost = 0;
   strcpy(nodeType, "Unknown Type");
endif

PRICE->HIN_Title ": " nodeType
if (printcost)
   "Program Cost            Development             Production                    Total
Cost"
   "Engineering"
   "  Drafting"%-20s       PRICE->HOUT_Drafting_DEV%15.2f  PRICE->HOUT_Drafting_PROD%15.2f
\
  PRICE->HOUT_Drafting_TOT%15.2f
   "   Design"%-20s       PRICE->HOUT_Design_DEV%15.2f     PRICE->HOUT_Design_PROD%15.2f
\
  PRICE->HOUT_Design_TOT%15.2f
    " "
endif
Section_End
```

After PRICE executes this pirl program it creates an output file like the following

```
From PRICE H on Wed, 13-Mar-96 at 14:43
Radar System : Assembly
Program Cost         Development          Production          Total Cost
Engineering
```

```
    Drafting                      1504.69                  290.08             1794.78
    Design                       1848.41                  352.72             2201.13

Antenna : Mechanical
Program Cost         Development          Production          Total Cost
Engineering
    Drafting                        47.86                        0.72                48.58
    Design                       137.28                         1.90             139.18

Receiver/Signal Processor : Purchased

Antenna Pedestal : Furnished

Computer Modification : Modified
Program Cost         Development          Production          Total Cost
Engineering
    Drafting                            2.30                      4.23
6.54
    Design                             5.40                      11.78
17.17

Transmitter Assembly : Assembly
Program Cost         Development          Production          Total Cost
Engineering
    Drafting                       904.69                   120.08             1024.78
    Design                       1548.41                  252.72             1801.13

          Transmitter : Electronic
Program Cost         Development          Production          Total Cost
Engineering
    Drafting                       454.53                     85.13               539.66
    Design                       1405.73                    239.05            1644.78

System Processor : Unknown Type
```

5.4 Appendix A: PRICE EBS Node Modes

Mode	Description
1	Electronic Item
2	Mechanical Item
3	Purchased Item
4	Furnished Item
5	Integration and Test
6	Modify Item
7	Calibration
8	Throughput
9	Multiple Lot
33	Detailed Cost item
51	Design Integration
52	Hardware/Software Integration
60	System
62	Assembly
80	Software

5.5 Appendix B: PRL Variable List

The following is a list of variables recognized by PRICE Enterprise. To use them as PRICE variables as described in 4.0.3, prefix them with PRICE->; for node creation variables, prefix them with PRICE<-.

5.5.1 H Inputs

HIN_Title	STRING_TYPE
HIN_Mode	VALUE_TYPE
HIN_Indenture	VALUE_TYPE
HIN_WBS_Number	VALUE_TYPE
HIN_APPL	VALUE_TYPE
HIN_AUCOST	VALUE_TYPE
HIN_CATGRY	VALUE_TYPE
HIN_COST	VALUE_TYPE
HIN_CPLXM	VALUE_TYPE
HIN_DCOST	VALUE_TYPE
HIN_DDACST	VALUE_TYPE
HIN_DDECST	VALUE_TYPE
HIN_DDRCST	VALUE_TYPE
HIN_DEND	VALUE_TYPE
HIN_DESRPE	VALUE_TYPE
HIN_DESRPS	VALUE_TYPE
HIN_DFPRO	VALUE_TYPE
HIN_DLEVE	VALUE_TYPE
HIN_DLEVS	VALUE_TYPE
HIN_DLPRO	VALUE_TYPE
HIN_DMULT	VALUE_TYPE
HIN_DPJCST	VALUE_TYPE
HIN_DPRCST	VALUE_TYPE
HIN_DSTART	VALUE_TYPE
HIN_DSYCST	VALUE_TYPE
HIN_DTCOST	VALUE_TYPE
HIN_DTLGTS	VALUE_TYPE
HIN_DTTCST	VALUE_TYPE
HIN_ECMPLX	VALUE_TYPE
HIN_EPLANS	VALUE_TYPE
HIN_EREL	VALUE_TYPE
HIN_COST_TYPE	VALUE_TYPE
HIN_FRAC	VALUE_TYPE
HIN_HSINT	VALUE_TYPE
HIN_INTEGE	VALUE_TYPE
HIN_INTEGS	VALUE_TYPE
HIN_LANG	VALUE_TYPE
HIN_LOT	VALUE_TYPE

HIN_MCPLXE	VALUE_TYPE
HIN_MCPLXS	VALUE_TYPE
HIN_MREL	VALUE_TYPE
HIN_NEWEL	VALUE_TYPE
HIN_NEWST	VALUE_TYPE
HIN_PCOST	VALUE_TYPE
HIN_PDACST	VALUE_TYPE
HIN_PDECST	VALUE_TYPE
HIN_PDRCST	VALUE_TYPE
HIN_PEND	VALUE_TYPE
HIN_PFAD	VALUE_TYPE
HIN_MPI	VALUE_TYPE
HIN_PIF	VALUE_TYPE
HIN_PLTFM	VALUE_TYPE
HIN_PMULT	VALUE_TYPE
HIN_PPJCST	VALUE_TYPE
HIN_PPRCST	VALUE_TYPE
HIN_PRCOST	VALUE_TYPE
HIN_PROSUP	VALUE_TYPE
HIN_PROTOS	VALUE_TYPE
HIN_PSTART	VALUE_TYPE
HIN_PTCOST	VALUE_TYPE
HIN_PTLGTS	VALUE_TYPE
HIN_PTTCST	VALUE_TYPE
HIN_QTY	VALUE_TYPE
HIN_QTYNHA	VALUE_TYPE
HIN_RATOOL	VALUE_TYPE
HIN_SLOC	VALUE_TYPE
HIN_SPLANS	VALUE_TYPE
HIN_TCOST	VALUE_TYPE
HIN_VOL	VALUE_TYPE
HIN_WECF_USEVO	VALUE_TYPE
HIN_WS	VALUE_TYPE
HIN_WT	VALUE_TYPE
HIN_YRBASE	VALUE_TYPE
HIN_YRTECH	VALUE_TYPE
HIN_NOTE	STRING_TYPE

5.5.2 H Outputs

HOUT_Title	STRING_TYPE
HOUT_Drafting_DEV	VALUE_TYPE
HOUT_Drafting_PROD	VALUE_TYPE
HOUT_Drafting_TOT	VALUE_TYPE
HOUT_Design_DEV	VALUE_TYPE
HOUT_Design_PROD	VALUE_TYPE
HOUT_Design_TOT	VALUE_TYPE
HOUT_Systems_DEV	VALUE_TYPE
HOUT_Systems_TOT	VALUE_TYPE
HOUT_ProgMgmt_DEV	VALUE_TYPE
HOUT_ProgMgmt_PRO	VALUE_TYPE
HOUT_ProgMgmt_TOT	VALUE_TYPE
HOUT_Data_DEV	VALUE_TYPE
HOUT_Data_PROD	VALUE_TYPE
HOUT_Data_TOT	VALUE_TYPE
HOUT_Engineering_DEV	VALUE_TYPE
HOUT_Engineering_PROD	VALUE_TYPE
HOUT_Engineering_TOT	VALUE_TYPE
HOUT_Production_PROD	VALUE_TYPE
HOUT_Production_TOT	VALUE_TYPE
HOUT_Prototype_DEV	VALUE_TYPE
HOUT_Prototype_TOT	VALUE_TYPE
HOUT_ToolTest_DEV	VALUE_TYPE
HOUT_ToolTest_PROD	VALUE_TYPE
HOUT_ToolTest_TOT	VALUE_TYPE
HOUT_Manufacturing_DEV	VALUE_TYPE
HOUT_Manufacturing_PROD	VALUE_TYPE
HOUT_Manufacturing_TOT	VALUE_TYPE
HOUT_Total_DEV	VALUE_TYPE
HOUT_Total_PROD	VALUE_TYPE
HOUT_Total_TOT	VALUE_TYPE
HOUT_Cal_MCPLXS	VALUE_TYPE
HOUT_Cal_MCPLXE	VALUE_TYPE
HOUT_Purch_Cost	VALUE_TYPE
HOUT_Avg_Unit_Cost	VALUE_TYPE
HOUT_DSTART	VALUE_TYPE
HOUT_DFPRO	VALUE_TYPE
HOUT_DLPRO	VALUE_TYPE

HOUT_PSTART	VALUE_TYPE
HOUT_PFAD	VALUE_TYPE
HOUT_PEND	VALUE_TYPE
HOUT_MTBF	VALUE_TYPE

5.5.3 L Inputs

LIN_Title	STRING_TYPE
LIN_Equipment_Config	VALUE_TYPE
LIN_Indenture	VALUE_TYPE
LIN_WBS_Number	VALUE_TYPE
LIN_MTBF	VALUE_TYPE
LIN_TF	VALUE_TYPE
LIN_TI	VALUE_TYPE
LIN_TD	VALUE_TYPE
LIN_TMO	VALUE_TYPE
LIN_TMI	VALUE_TYPE
LIN_TMD	VALUE_TYPE
LIN_EE	VALUE_TYPE
LIN_FN	VALUE_TYPE
LIN_CEND	VALUE_TYPE
LIN_CPE	VALUE_TYPE
LIN_CUR	VALUE_TYPE
LIN_CMR	VALUE_TYPE
LIN_TRE	VALUE_TYPE
LIN_P	VALUE_TYPE
LIN_PP	VALUE_TYPE
LIN_FNSP	VALUE_TYPE
LIN_CPPE	VALUE_TYPE
LIN_CFIM	VALUE_TYPE
LIN_CFIP	VALUE_TYPE
LIN_FTSQF	VALUE_TYPE
LIN_FTSQP	VALUE_TYPE
LIN_TC	VALUE_TYPE
LIN_CCOU	VALUE_TYPE
LIN_FTSQC	VALUE_TYPE
LIN_ProdCost_LRU	VALUE_TYPE
LIN_ProdCost_MOD	VALUE_TYPE
LIN_ProdCost_PART	VALUE_TYPE
LIN_LCurve_LRU	VALUE_TYPE

LIN_LCurve_MOD	VALUE_TYPE
LIN_LCurve_PART	VALUE_TYPE
LIN_RefQty_LRU	VALUE_TYPE
LIN_RefQty_MOD	VALUE_TYPE
LIN_RefQty_PART	VALUE_TYPE
LIN_WT_LRU	VALUE_TYPE
LIN_WT_MOD	VALUE_TYPE
LIN_WT_PART	VALUE_TYPE
LIN_VOL_LRU	VALUE_TYPE
LIN_VOL_MOD	VALUE_TYPE
LIN_VOL_PART	VALUE_TYPE
LIN_Thru_CATGRY	VALUE_TYPE
LIN_Thru_DCOST	VALUE_TYPE
LIN_Thru_PCOST	VALUE_TYPE
LIN_Thru_SCOST	VALUE_TYPE
LIN_Thru_TCOST	VALUE_TYPE
LIN_Concept1	VALUE_TYPE
LIN_Concept2	VALUE_TYPE
LIN_Concept3	VALUE_TYPE
LIN_Concept4	VALUE_TYPE
LIN_Concept5	VALUE_TYPE
LIN_Concept6	VALUE_TYPE
LIN_Concept7	VALUE_TYPE
LIN_Concept8	VALUE_TYPE
LIN_Concept9	VALUE_TYPE
LIN_Concept10	VALUE_TYPE
LIN_Concept11	VALUE_TYPE
LIN_Concept12	VALUE_TYPE
LIN_Concept13	VALUE_TYPE
LIN_Concept14	VALUE_TYPE
LIN_Concept15	VALUE_TYPE
LIN_Concept16	VALUE_TYPE
LIN_Concept17	VALUE_TYPE
LIN_Concept18	VALUE_TYPE
LIN_Concept19	VALUE_TYPE
LIN_Concept20	VALUE_TYPE
LIN_Concept21	VALUE_TYPE
LIN_Concept22	VALUE_TYPE
LIN_Concept23	VALUE_TYPE

LIN_Concept24	VALUE_TYPE
LIN_Concept25	VALUE_TYPE
LIN_Concept26	VALUE_TYPE
LIN_Concept27	VALUE_TYPE
LIN_Concept28	VALUE_TYPE

5.5.4 L Outputs

LOUT_Title	STRING_TYPE
LOUT_MissionEqp_DEV	VALUE_TYPE
LOUT_MissionEqp_PROD	VALUE_TYPE
LOUT_MissionEqp_TOT	VALUE_TYPE
LOUT_SupportEqp_PROD	VALUE_TYPE
LOUT_SupportEqp_SUP	VALUE_TYPE
LOUT_SupportEqp_TOT	VALUE_TYPE
LOUT_Supply_PROD	VALUE_TYPE
LOUT_Supply_SUP	VALUE_TYPE
LOUT_Supply_TOT	VALUE_TYPE
LOUT_SupAdmin_PROD	VALUE_TYPE
LOUT_SupAdmin_SUP	VALUE_TYPE
LOUT_SupAdmin_TOT	VALUE_TYPE
LOUT_Labor_SUP	VALUE_TYPE
LOUT_Labor_TOT	VALUE_TYPE
LOUT_Contract_SUP	VALUE_TYPE
LOUT_Contract_TOT	VALUE_TYPE
LOUT_Other_SUP	VALUE_TYPE
LOUT_Other_TOT	VALUE_TYPE
LOUT_Total_DEV	VALUE_TYPE
LOUT_Total_PROD	VALUE_TYPE
LOUT_Total_SUP	VALUE_TYPE
LOUT_Total_TOT	VALUE_TYPE
LOUT_Field_Sup_Thru_DEV	VALUE_TYPE
LOUT_Field_Sup_Thru_PROD	VALUE_TYPE
LOUT_Field_Sup_Thru_SUP	VALUE_TYPE
LOUT_Field_Sup_Thru_TOT	VALUE_TYPE
LOUT_Field_Test_Thru_DEV	VALUE_TYPE
LOUT_Field_Test_Thru_PROD	VALUE_TYPE
LOUT_Field_Test_Thru_SUP	VALUE_TYPE
LOUT_Field_Test_Thru_TOT	VALUE_TYPE
LOUT_Software_Thru_DEV	VALUE_TYPE

LOUT_Software_Thru_PROD	VALUE_TYPE
LOUT_Software_Thru_SUP	VALUE_TYPE
LOUT_Software_Thru_TOT	VALUE_TYPE
LOUT_Other_Thru_DEV	VALUE_TYPE
LOUT_Other_Thru_PROD	VALUE_TYPE
LOUT_Other_Thru_SUP	VALUE_TYPE
LOUT_Other_Thru_TOT	VALUE_TYPE
LOUT_Total_Thru_DEV	VALUE_TYPE
LOUT_Total_Thru_PROD	VALUE_TYPE
LOUT_Total_Thru_SUP	VALUE_TYPE
LOUT_Total_Thru_TOT	VALUE_TYPE
LOUT_Availability	VALUE_TYPE
LOUT_Readiness	VALUE_TYPE
LOUT_Reliability	VALUE_TYPE
LOUT_TestSets_Org	VALUE_TYPE
LOUT_TestSets_Intermed	VALUE_TYPE
LOUT_TestSets_Depot	VALUE_TYPE
LOUT_UtilFac_Org	VALUE_TYPE
LOUT_UtilFac_Intermed	VALUE_TYPE
LOUT_UtilFac_Depot	VALUE_TYPE
LOUT_LoadFac_org	VALUE_TYPE
LOUT_LoadFac_Intermed	VALUE_TYPE
LOUT_LoadFac_Depot	VALUE_TYPE
LOUT_Initial_LRUs	VALUE_TYPE
LOUT_Initial_Modules	VALUE_TYPE
LOUT_Initial_Parts	VALUE_TYPE
LOUT_Replenishment_LRUs	VALUE_TYPE
LOUT_Replenishment_Modules	VALUE_TYPE
LOUT_Replenishment_Parts	VALUE_TYPE
LOUT_Maintenance_Concept	VALUE_TYPE

5.5.5 H Globals

HG_GLOBAL_TTILE	VALUE_TYPE
HG_DDATA	VALUE_TYPE
HG_DDRAFT	VALUE_TYPE
HG_DDSIGN	VALUE_TYPE
HG_Global_DMULT	VALUE_TYPE
HG_DPROJ	VALUE_TYPE
HG_ECNE	VALUE_TYPE

HG_ECNS	VALUE_TYPE
HG_ETLG1	VALUE_TYPE
HG_ETLG2	VALUE_TYPE
HG_GAPFAC	VALUE_TYPE
HG_GDTLGT	VALUE_TYPE
HG_GPTLGT	VALUE_TYPE
HG_LOTFAC	VALUE_TYPE
HG_NFACS	VALUE_TYPE
HG_NSHIFT	VALUE_TYPE
HG_PDATA	VALUE_TYPE
HG_PDRAFT	VALUE_TYPE
HG_PDSIGN	VALUE_TYPE
HG_Global_PMULT	VALUE_TYPE
HG_PPROJ	VALUE_TYPE
HG_PRMULT	VALUE_TYPE
HG_PSF	VALUE_TYPE
HG_SMODS	VALUE_TYPE
HG_STLG1	VALUE_TYPE
HG_STLG2	VALUE_TYPE
HG_SYSTEM	VALUE_TYPE
HG_TCALD	VALUE_TYPE
HG_TCALP	VALUE_TYPE
HG_TECDEL	VALUE_TYPE
HG_UNITLC	VALUE_TYPE
HG_ZTECH	VALUE_TYPE

5.5.6 HL Globals

LG_HL_Global_Title	STRING_TYPE
LG_CKUE	VALUE_TYPE
LG_CKUO	VALUE_TYPE
LG_CKUI	VALUE_TYPE
LG_CKUD	VALUE_TYPE
LG_CKME	VALUE_TYPE
LG_CKMO	VALUE_TYPE
LG_CKMI	VALUE_TYPE
LG_CKMD	VALUE_TYPE
LG_CKPE	VALUE_TYPE
LG_CKPO	VALUE_TYPE
LG_CKPI	VALUE_TYPE

LG_CKPD	VALUE_TYPE
LG_ZUE	VALUE_TYPE
LG_ZUO	VALUE_TYPE
LG_ZUI	VALUE_TYPE
LG_ZUD	VALUE_TYPE
LG_ZME	VALUE_TYPE
LG_ZMO	VALUE_TYPE
LG_ZMI	VALUE_TYPE
LG_ZMD	VALUE_TYPE
LG_ZPE	VALUE_TYPE
LG_ZPO	VALUE_TYPE
LG_ZPI	VALUE_TYPE
LG_ZPD	VALUE_TYPE
LG_HPU	VALUE_TYPE
LG_HPM	VALUE_TYPE
LG_HPP	VALUE_TYPE
LG_WOX	VALUE_TYPE
LG_WI	VALUE_TYPE
LG_WD	VALUE_TYPE
LG_CUE	VALUE_TYPE
LG_CUO	VALUE_TYPE
LG_CUI	VALUE_TYPE
LG_CUD	VALUE_TYPE
LG_FD21	VALUE_TYPE
LG_FD31	VALUE_TYPE
LG_AFSA	VALUE_TYPE
LG_SYR	VALUE_TYPE
LG_ANPR	VALUE_TYPE
LG_AOFF	VALUE_TYPE
LG_CEN	VALUE_TYPE
LG_CAD	VALUE_TYPE
LG_CLRUPG	VALUE_TYPE
LG_CMODPG	VALUE_TYPE
LG_EV	VALUE_TYPE
LG_FPE	VALUE_TYPE
LG_FNGF	VALUE_TYPE
LG_PCTS	VALUE_TYPE
LG_PODF	VALUE_TYPE
LG_REPEAT	VALUE_TYPE

LG_SMF	VALUE_TYPE
LG_YAT	VALUE_TYPE
LG_ASC	VALUE_TYPE
LG_DCEND	VALUE_TYPE
LG_DPEC	VALUE_TYPE
LG_DPSEC	VALUE_TYPE
LG_DPSC	VALUE_TYPE
LG_ACCOU	VALUE_TYPE
LG_ACFIM	VALUE_TYPE
LG_ACFIP	VALUE_TYPE
LG_ACM	VALUE_TYPE
LG_ACMP	VALUE_TYPE
LG_ACMR	VALUE_TYPE
LG_ACP	VALUE_TYPE
LG_ACPP	VALUE_TYPE
LG_ATC	VALUE_TYPE
LG_ACU	VALUE_TYPE
LG_ACUR	VALUE_TYPE
LG_AFTC	VALUE_TYPE
LG_AFTF	VALUE_TYPE
LG_AFTP	VALUE_TYPE
LG_AMTBF	VALUE_TYPE
LG_AP	VALUE_TYPE
LG_APP	VALUE_TYPE
LG_ATF	VALUE_TYPE
LG_ATMO	VALUE_TYPE
LG_AWM	VALUE_TYPE
LG_AWP	VALUE_TYPE
LG_AWU	VALUE_TYPE
LG_ACCPE	VALUE_TYPE
LG_HE_th1	VALUE_TYPE
LG_HE_th2	VALUE_TYPE
LG_HE_th3	VALUE_TYPE
LG_DOSE_th1	VALUE_TYPE
LG_DOSE_th2	VALUE_TYPE
LG_DOSE_th3	VALUE_TYPE
LG_DOSOC_th1	VALUE_TYPE
LG_DOSOC_th2	VALUE_TYPE
LG_DOSOC_th3	VALUE_TYPE

LG_DOSOR_th1	VALUE_TYPE
LG_DOSOR_th2	VALUE_TYPE
LG_DOSOR_th3	VALUE_TYPE
LG_DOSIC_th1	VALUE_TYPE
LG_DOSIC_th2	VALUE_TYPE
LG_DOSIC_th3	VALUE_TYPE
LG_DOSIR_th1	VALUE_TYPE
LG_DOSIR_th2	VALUE_TYPE
LG_DOSIR_th3	VALUE_TYPE
LG_DOSDR_th1	VALUE_TYPE
LG_DOSDR_th2	VALUE_TYPE
LG_DOSDR_th3	VALUE_TYPE
LG_CDOSIC_th1	VALUE_TYPE
LG_CDOSIC_th2	VALUE_TYPE
LG_CDOSIC_th3	VALUE_TYPE
LG_CDOSIR_th1	VALUE_TYPE
LG_CDOSIR_th2	VALUE_TYPE
LG_CDOSIR_th3	VALUE_TYPE
LG_CDOSDR_th1	VALUE_TYPE
LG_CDOSDR_th2	VALUE_TYPE
LG_CDOSDR_th3	VALUE_TYPE
LG_DOSDCU_th1	VALUE_TYPE
LG_DOSDCU_th2	VALUE_TYPE
LG_DOSDCU_th3	VALUE_TYPE
LG_DOSDCM_th1	VALUE_TYPE
LG_DOSDCM_th2	VALUE_TYPE
LG_DOSDCM_th3	VALUE_TYPE
LG_DOSDCP_th1	VALUE_TYPE
LG_DOSDCP_th2	VALUE_TYPE
LG_DOSDCP_th3	VALUE_TYPE
LG_CDEO_th1	VALUE_TYPE
LG_CDEO_th2	VALUE_TYPE
LG_CDEO_th3	VALUE_TYPE
LG_CDOE_th1	VALUE_TYPE
LG_CDOE_th2	VALUE_TYPE
LG_CDOE_th3	VALUE_TYPE
LG_CDOI_th1	VALUE_TYPE
LG_CDOI_th2	VALUE_TYPE
LG_CDOI_th3	VALUE_TYPE

LG_CDIO_th1	VALUE_TYPE
LG_CDIO_th2	VALUE_TYPE
LG_CDIO_th3	VALUE_TYPE
LG_CDID_th1	VALUE_TYPE
LG_CDID_th2	VALUE_TYPE
LG_CDID_th3	VALUE_TYPE
LG_CDDI_th1	VALUE_TYPE
LG_CDDI_th2	VALUE_TYPE
LG_CDDI_th3	VALUE_TYPE
LG_CDFD_th1	VALUE_TYPE
LG_CDFD_th2	VALUE_TYPE
LG_CDFD_th3	VALUE_TYPE
LG_SUE_th1	VALUE_TYPE
LG_SUE_th2	VALUE_TYPE
LG_SUE_th3	VALUE_TYPE
LG_SUO_th1	VALUE_TYPE
LG_SUO_th2	VALUE_TYPE
LG_SUO_th3	VALUE_TYPE
LG_SUI_th1	VALUE_TYPE
LG_SUI_th2	VALUE_TYPE
LG_SUI_th3	VALUE_TYPE
LG_SUD_th1	VALUE_TYPE
LG_SUD_th2	VALUE_TYPE
LG_SUD_th3	VALUE_TYPE
LG_SME_th1	VALUE_TYPE
LG_SME_th2	VALUE_TYPE
LG_SME_th3	VALUE_TYPE
LG_SMO_th1	VALUE_TYPE
LG_SMO_th2	VALUE_TYPE
LG_SMO_th3	VALUE_TYPE
LG_SMI_th1	VALUE_TYPE
LG_SMI_th2	VALUE_TYPE
LG_SMI_th3	VALUE_TYPE
LG_SMD_th1	VALUE_TYPE
LG_SMD_th2	VALUE_TYPE
LG_SMD_th3	VALUE_TYPE
LG_FUE_th1	VALUE_TYPE
LG_FUE_th2	VALUE_TYPE
LG_FUE_th3	VALUE_TYPE

LG_FUO_th1	VALUE_TYPE
LG_FUO_th2	VALUE_TYPE
LG_FUO_th3	VALUE_TYPE
LG_FUI_th1	VALUE_TYPE
LG_FUI_th2	VALUE_TYPE
LG_FUI_th3	VALUE_TYPE
LG_FUD_th1	VALUE_TYPE
LG_FUD_th2	VALUE_TYPE
LG_FUD_th3	VALUE_TYPE
LG_FMO_th1	VALUE_TYPE
LG_FMO_th2	VALUE_TYPE
LG_FMO_th3	VALUE_TYPE
LG_FMI_th1	VALUE_TYPE
LG_FMI_th2	VALUE_TYPE
LG_FMI_th3	VALUE_TYPE
LG_FMD_th1	VALUE_TYPE
LG_FMD_th2	VALUE_TYPE
LG_FMD_th3	VALUE_TYPE
LG_CFTO2_th1	VALUE_TYPE
LG_CFTO2_th2	VALUE_TYPE
LG_CFTO2_th3	VALUE_TYPE
LG_CFTI2_th1	VALUE_TYPE
LG_CFTI2_th2	VALUE_TYPE
LG_CFTI2_th3	VALUE_TYPE
LG_CFTD2_th1	VALUE_TYPE
LG_CFTD2_th2	VALUE_TYPE
LG_CFTD2_th3	VALUE_TYPE
LG_CFTO3_th1	VALUE_TYPE
LG_CFTO3_th2	VALUE_TYPE
LG_CFTO3_th3	VALUE_TYPE
LG_CFTI3_th1	VALUE_TYPE
LG_CFTI3_th2	VALUE_TYPE
LG_CFTI3_th3	VALUE_TYPE
LG_CFTD3_th1	VALUE_TYPE
LG_CFTD3_th2	VALUE_TYPE
LG_CFTD3_th3	VALUE_TYPE
LG_RATIO_th1	VALUE_TYPE
LG_RATIO_th2	VALUE_TYPE
LG_RATIO_th3	VALUE_TYPE

5.5.7 HL Deployment Table

LD_Deployment_Title	STRING_TYPE
LD_Number_Of_Years	VALUE_TYPE
LD_Number_Of_Theaters	VALUE_TYPE
LD_Mission_Period_th1	VALUE_TYPE
LD_Mission_Period_th2	VALUE_TYPE
LD_Mission_Period_th3	VALUE_TYPE
LD_OD_1	VALUE_TYPE
LD_OD_2	VALUE_TYPE
LD_OD_3	VALUE_TYPE
LD_DI_1	VALUE_TYPE
LD_DI_2	VALUE_TYPE
LD_DI_3	VALUE_TYPE
LD_DD_1	VALUE_TYPE
LD_DD_2	VALUE_TYPE
LD_DD_3	VALUE_TYPE
LD_EDS_1	VALUE_TYPE
LD_EDS_2	VALUE_TYPE
LD_EDS_3	VALUE_TYPE
LD_ODS_1	VALUE_TYPE
LD_ODS_2	VALUE_TYPE
LD_ODS_3	VALUE_TYPE
LD_DIS_1	VALUE_TYPE
LD_DIS_2	VALUE_TYPE
LD_DIS_3	VALUE_TYPE
LD_DDS_1	VALUE_TYPE
LD_DDS_2	VALUE_TYPE
LD_DDS_3	VALUE_TYPE
LD_ED_Th1_Yr1	VALUE_TYPE
LD_ED_Th2_Yr1	VALUE_TYPE
LD_ED_Th3_Yr1	VALUE_TYPE
LD_OTF_Th1_Yr1	VALUE_TYPE
LD_OTF_Th2_Yr1	VALUE_TYPE
LD_OTF_Th3_Yr1	VALUE_TYPE
LD_ED_Th1_Yr2	VALUE_TYPE
LD_ED_Th2_Yr2	VALUE_TYPE
LD_ED_Th3_Yr2	VALUE_TYPE
LD_OTF_Th1_Yr2	VALUE_TYPE
LD_OTF_Th2_Yr2	VALUE_TYPE

LD_OTF_Th3_Yr2	VALUE_TYPE
LD_ED_Th1_Yr3	VALUE_TYPE
LD_ED_Th2_Yr3	VALUE_TYPE
LD_ED_Th3_Yr3	VALUE_TYPE
LD_OTF_Th1_Yr3	VALUE_TYPE
LD_OTF_Th2_Yr3	VALUE_TYPE
LD_OTF_Th3_Yr3	VALUE_TYPE
LD_ED_Th1_Yr4	VALUE_TYPE
LD_ED_Th2_Yr4	VALUE_TYPE
LD_ED_Th3_Yr4	VALUE_TYPE
LD_OTF_Th1_Yr4	VALUE_TYPE
LD_OTF_Th2_Yr4	VALUE_TYPE
LD_OTF_Th3_Yr4	VALUE_TYPE
LD_ED_Th1_Yr5	VALUE_TYPE
LD_ED_Th2_Yr5	VALUE_TYPE
LD_ED_Th3_Yr5	VALUE_TYPE
LD_OTF_Th1_Yr5	VALUE_TYPE
LD_OTF_Th2_Yr5	VALUE_TYPE
LD_OTF_Th3_Yr5	VALUE_TYPE
LD_ED_Th1_Yr6	VALUE_TYPE
LD_ED_Th2_Yr6	VALUE_TYPE
LD_ED_Th3_Yr6	VALUE_TYPE
LD_OTF_Th1_Yr6	VALUE_TYPE
LD_OTF_Th2_Yr6	VALUE_TYPE
LD_OTF_Th3_Yr6	VALUE_TYPE
LD_ED_Th1_Yr7	VALUE_TYPE
LD_ED_Th2_Yr7	VALUE_TYPE
LD_ED_Th3_Yr7	VALUE_TYPE
LD_OTF_Th1_Yr7	VALUE_TYPE
LD_OTF_Th2_Yr7	VALUE_TYPE
LD_OTF_Th3_Yr7	VALUE_TYPE
LD_ED_Th1_Yr8	VALUE_TYPE
LD_ED_Th2_Yr8	VALUE_TYPE
LD_ED_Th3_Yr8	VALUE_TYPE
LD_OTF_Th1_Yr8	VALUE_TYPE
LD_OTF_Th2_Yr8	VALUE_TYPE
LD_OTF_Th3_Yr8	VALUE_TYPE
LD_ED_Th1_Yr9	VALUE_TYPE
LD_ED_Th2_Yr9	VALUE_TYPE

LD_ED_Th3_Yr9	VALUE_TYPE
LD_OTF_Th1_Yr9	VALUE_TYPE
LD_OTF_Th2_Yr9	VALUE_TYPE
LD_OTF_Th3_Yr9	VALUE_TYPE
LD_ED_Th1_Yr10	VALUE_TYPE
LD_ED_Th2_Yr10	VALUE_TYPE
LD_ED_Th3_Yr10	VALUE_TYPE
LD_OTF_Th1_Yr10	VALUE_TYPE
LD_OTF_Th2_Yr10	VALUE_TYPE
LD_OTF_Th3_Yr10	VALUE_TYPE
LD_ED_Th1_Yr11	VALUE_TYPE
LD_ED_Th2_Yr11	VALUE_TYPE
LD_ED_Th3_Yr11	VALUE_TYPE
LD_OTF_Th1_Yr11	VALUE_TYPE
LD_OTF_Th2_Yr11	VALUE_TYPE
LD_OTF_Th3_Yr11	VALUE_TYPE
LD_ED_Th1_Yr12	VALUE_TYPE
LD_ED_Th2_Yr12	VALUE_TYPE
LD_ED_Th3_Yr12	VALUE_TYPE
LD_OTF_Th1_Yr12	VALUE_TYPE
LD_OTF_Th2_Yr12	VALUE_TYPE
LD_OTF_Th3_Yr12	VALUE_TYPE
LD_ED_Th1_Yr13	VALUE_TYPE
LD_ED_Th2_Yr13	VALUE_TYPE
LD_ED_Th3_Yr13	VALUE_TYPE
LD_OTF_Th1_Yr13	VALUE_TYPE
LD_OTF_Th2_Yr13	VALUE_TYPE
LD_OTF_Th3_Yr13	VALUE_TYPE
LD_ED_Th1_Yr14	VALUE_TYPE
LD_ED_Th2_Yr14	VALUE_TYPE
LD_ED_Th3_Yr14	VALUE_TYPE
LD_OTF_Th1_Yr14	VALUE_TYPE
LD_OTF_Th2_Yr14	VALUE_TYPE
LD_OTF_Th3_Yr14	VALUE_TYPE
LD_ED_Th1_Yr15	VALUE_TYPE
LD_ED_Th2_Yr15	VALUE_TYPE
LD_ED_Th3_Yr15	VALUE_TYPE
LD_OTF_Th1_Yr15	VALUE_TYPE
LD_OTF_Th2_Yr15	VALUE_TYPE

LD_OTF_Th3_Yr15	VALUE_TYPE
LD_ED_Th1_Yr16	VALUE_TYPE
LD_ED_Th2_Yr16	VALUE_TYPE
LD_ED_Th3_Yr16	VALUE_TYPE
LD_OTF_Th1_Yr16	VALUE_TYPE
LD_OTF_Th2_Yr16	VALUE_TYPE
LD_OTF_Th3_Yr16	VALUE_TYPE
LD_ED_Th1_Yr17	VALUE_TYPE
LD_ED_Th2_Yr17	VALUE_TYPE
LD_ED_Th3_Yr17	VALUE_TYPE
LD_OTF_Th1_Yr17	VALUE_TYPE
LD_OTF_Th2_Yr17	VALUE_TYPE
LD_OTF_Th3_Yr17	VALUE_TYPE
LD_ED_Th1_Yr18	VALUE_TYPE
LD_ED_Th2_Yr18	VALUE_TYPE
LD_ED_Th3_Yr18	VALUE_TYPE
LD_OTF_Th1_Yr18	VALUE_TYPE
LD_OTF_Th2_Yr18	VALUE_TYPE
LD_OTF_Th3_Yr18	VALUE_TYPE
LD_ED_Th1_Yr19	VALUE_TYPE
LD_ED_Th2_Yr19	VALUE_TYPE
LD_ED_Th3_Yr19	VALUE_TYPE
LD_OTF_Th1_Yr19	VALUE_TYPE
LD_OTF_Th2_Yr19	VALUE_TYPE
LD_OTF_Th3_Yr19	VALUE_TYPE
LD_ED_Th1_Yr20	VALUE_TYPE
LD_ED_Th2_Yr20	VALUE_TYPE
LD_ED_Th3_Yr20	VALUE_TYPE
LD_OTF_Th1_Yr20	VALUE_TYPE
LD_OTF_Th2_Yr20	VALUE_TYPE
LD_OTF_Th3_Yr20	VALUE_TYPE
LD_ED_Th1_Yr21	VALUE_TYPE
LD_ED_Th2_Yr21	VALUE_TYPE
LD_ED_Th3_Yr21	VALUE_TYPE
LD_OTF_Th1_Yr21	VALUE_TYPE
LD_OTF_Th2_Yr21	VALUE_TYPE
LD_OTF_Th3_Yr21	VALUE_TYPE
LD_ED_Th1_Yr22	VALUE_TYPE
LD_ED_Th2_Yr22	VALUE_TYPE

LD_ED_Th3_Yr22	VALUE_TYPE
LD_OTF_Th1_Yr22	VALUE_TYPE
LD_OTF_Th2_Yr22	VALUE_TYPE
LD_OTF_Th3_Yr22	VALUE_TYPE
LD_ED_Th1_Yr23	VALUE_TYPE
LD_ED_Th2_Yr23	VALUE_TYPE
LD_ED_Th3_Yr23	VALUE_TYPE
LD_OTF_Th1_Yr23	VALUE_TYPE
LD_OTF_Th2_Yr23	VALUE_TYPE
LD_OTF_Th3_Yr23	VALUE_TYPE
LD_ED_Th1_Yr24	VALUE_TYPE
LD_ED_Th2_Yr24	VALUE_TYPE
LD_ED_Th3_Yr24	VALUE_TYPE
LD_OTF_Th1_Yr24	VALUE_TYPE
LD_OTF_Th2_Yr24	VALUE_TYPE
LD_OTF_Th3_Yr24	VALUE_TYPE
LD_ED_Th1_Yr25	VALUE_TYPE
LD_ED_Th2_Yr25	VALUE_TYPE
LD_ED_Th3_Yr25	VALUE_TYPE
LD_OTF_Th1_Yr25	VALUE_TYPE
LD_OTF_Th2_Yr25	VALUE_TYPE
LD_OTF_Th3_Yr25	VALUE_TYPE
LD_ED_Th1_Yr26	VALUE_TYPE
LD_ED_Th2_Yr26	VALUE_TYPE
LD_ED_Th3_Yr26	VALUE_TYPE
LD_OTF_Th1_Yr26	VALUE_TYPE
LD_OTF_Th2_Yr26	VALUE_TYPE
LD_OTF_Th3_Yr26	VALUE_TYPE
LD_ED_Th1_Yr27	VALUE_TYPE
LD_ED_Th2_Yr27	VALUE_TYPE
LD_ED_Th3_Yr27	VALUE_TYPE
LD_OTF_Th1_Yr27	VALUE_TYPE
LD_OTF_Th2_Yr27	VALUE_TYPE
LD_OTF_Th3_Yr27	VALUE_TYPE
LD_ED_Th1_Yr28	VALUE_TYPE
LD_ED_Th2_Yr28	VALUE_TYPE
LD_ED_Th3_Yr28	VALUE_TYPE
LD_OTF_Th1_Yr28	VALUE_TYPE
LD_OTF_Th2_Yr28	VALUE_TYPE

LD_OTF_Th3_Yr28	VALUE_TYPE
LD_ED_Th1_Yr29	VALUE_TYPE
LD_ED_Th2_Yr29	VALUE_TYPE
LD_ED_Th3_Yr29	VALUE_TYPE
LD_OTF_Th1_Yr29	VALUE_TYPE
LD_OTF_Th2_Yr29	VALUE_TYPE
LD_OTF_Th3_Yr29	VALUE_TYPE
LD_ED_Th1_Yr30	VALUE_TYPE
LD_ED_Th2_Yr30	VALUE_TYPE
LD_ED_Th3_Yr30	VALUE_TYPE
LD_OTF_Th1_Yr30	VALUE_TYPE
LD_OTF_Th2_Yr30	VALUE_TYPE
LD_OTF_Th3_Yr30	VALUE_TYPE
LD_ED_Th1_Yr31	VALUE_TYPE
LD_ED_Th2_Yr31	VALUE_TYPE
LD_ED_Th3_Yr31	VALUE_TYPE
LD_OTF_Th1_Yr31	VALUE_TYPE
LD_OTF_Th2_Yr31	VALUE_TYPE
LD_OTF_Th3_Yr31	VALUE_TYPE
LD_ED_Th1_Yr32	VALUE_TYPE
LD_ED_Th2_Yr32	VALUE_TYPE
LD_ED_Th3_Yr32	VALUE_TYPE
LD_OTF_Th1_Yr32	VALUE_TYPE
LD_OTF_Th2_Yr32	VALUE_TYPE
LD_OTF_Th3_Yr32	VALUE_TYPE
LD_ED_Th1_Yr33	VALUE_TYPE
LD_ED_Th2_Yr33	VALUE_TYPE
LD_ED_Th3_Yr33	VALUE_TYPE
LD_OTF_Th1_Yr33	VALUE_TYPE
LD_OTF_Th2_Yr33	VALUE_TYPE
LD_OTF_Th3_Yr33	VALUE_TYPE
LD_ED_Th1_Yr34	VALUE_TYPE
LD_ED_Th2_Yr34	VALUE_TYPE
LD_ED_Th3_Yr34	VALUE_TYPE
LD_OTF_Th1_Yr34	VALUE_TYPE
LD_OTF_Th2_Yr34	VALUE_TYPE
LD_OTF_Th3_Yr34	VALUE_TYPE
LD_ED_Th1_Yr35	VALUE_TYPE
LD_ED_Th2_Yr35	VALUE_TYPE

LD_ED_Th3_Yr35	VALUE_TYPE
LD_OTF_Th1_Yr35	VALUE_TYPE
LD_OTF_Th2_Yr35	VALUE_TYPE
LD_OTF_Th3_Yr35	VALUE_TYPE
LD_ED_Th1_Yr36	VALUE_TYPE
LD_ED_Th2_Yr36	VALUE_TYPE
LD_ED_Th3_Yr36	VALUE_TYPE
LD_OTF_Th1_Yr36	VALUE_TYPE
LD_OTF_Th2_Yr36	VALUE_TYPE
LD_OTF_Th3_Yr36	VALUE_TYPE
LD_ED_Th1_Yr37	VALUE_TYPE
LD_ED_Th2_Yr37	VALUE_TYPE
LD_ED_Th3_Yr37	VALUE_TYPE
LD_OTF_Th1_Yr37	VALUE_TYPE
LD_OTF_Th2_Yr37	VALUE_TYPE
LD_OTF_Th3_Yr37	VALUE_TYPE
LD_ED_Th1_Yr38	VALUE_TYPE
LD_ED_Th2_Yr38	VALUE_TYPE
LD_ED_Th3_Yr38	VALUE_TYPE
LD_OTF_Th1_Yr38	VALUE_TYPE
LD_OTF_Th2_Yr38	VALUE_TYPE
LD_OTF_Th3_Yr38	VALUE_TYPE
LD_ED_Th1_Yr39	VALUE_TYPE
LD_ED_Th2_Yr39	VALUE_TYPE
LD_ED_Th3_Yr39	VALUE_TYPE
LD_OTF_Th1_Yr39	VALUE_TYPE
LD_OTF_Th2_Yr39	VALUE_TYPE
LD_OTF_Th3_Yr39	VALUE_TYPE
LD_ED_Th1_Yr40	VALUE_TYPE
LD_ED_Th2_Yr40	VALUE_TYPE
LD_ED_Th3_Yr40	VALUE_TYPE
LD_OTF_Th1_Yr40	VALUE_TYPE
LD_OTF_Th2_Yr40	VALUE_TYPE
LD_OTF_Th3_Yr40	VALUE_TYPE
LD_ED_Th1_Yr41	VALUE_TYPE
LD_ED_Th2_Yr41	VALUE_TYPE
LD_ED_Th3_Yr41	VALUE_TYPE
LD_OTF_Th1_Yr41	VALUE_TYPE
LD_OTF_Th2_Yr41	VALUE_TYPE

LD_OTF_Th3_Yr41	VALUE_TYPE
LD_ED_Th1_Yr42	VALUE_TYPE
LD_ED_Th2_Yr42	VALUE_TYPE
LD_ED_Th3_Yr42	VALUE_TYPE
LD_OTF_Th1_Yr42	VALUE_TYPE
LD_OTF_Th2_Yr42	VALUE_TYPE
LD_OTF_Th3_Yr42	VALUE_TYPE
LD_ED_Th1_Yr43	VALUE_TYPE
LD_ED_Th2_Yr43	VALUE_TYPE
LD_ED_Th3_Yr43	VALUE_TYPE
LD_OTF_Th1_Yr43	VALUE_TYPE
LD_OTF_Th2_Yr43	VALUE_TYPE
LD_OTF_Th3_Yr43	VALUE_TYPE
LD_ED_Th1_Yr44	VALUE_TYPE
LD_ED_Th2_Yr44	VALUE_TYPE
LD_ED_Th3_Yr44	VALUE_TYPE
LD_OTF_Th1_Yr44	VALUE_TYPE
LD_OTF_Th2_Yr44	VALUE_TYPE
LD_OTF_Th3_Yr44	VALUE_TYPE
LD_ED_Th1_Yr45	VALUE_TYPE
LD_ED_Th2_Yr45	VALUE_TYPE
LD_ED_Th3_Yr45	VALUE_TYPE
LD_OTF_Th1_Yr45	VALUE_TYPE
LD_OTF_Th2_Yr45	VALUE_TYPE
LD_OTF_Th3_Yr45	VALUE_TYPE
LD_ED_Th1_Yr46	VALUE_TYPE
LD_ED_Th2_Yr46	VALUE_TYPE
LD_ED_Th3_Yr46	VALUE_TYPE
LD_OTF_Th1_Yr46	VALUE_TYPE
LD_OTF_Th2_Yr46	VALUE_TYPE
LD_OTF_Th3_Yr46	VALUE_TYPE
LD_ED_Th1_Yr47	VALUE_TYPE
LD_ED_Th2_Yr47	VALUE_TYPE
LD_ED_Th3_Yr47	VALUE_TYPE
LD_OTF_Th1_Yr47	VALUE_TYPE
LD_OTF_Th2_Yr47	VALUE_TYPE
LD_OTF_Th3_Yr47	VALUE_TYPE
LD_ED_Th1_Yr48	VALUE_TYPE
LD_ED_Th2_Yr48	VALUE_TYPE

LD_ED_Th3_Yr48	VALUE_TYPE
LD_OTF_Th1_Yr48	VALUE_TYPE
LD_OTF_Th2_Yr48	VALUE_TYPE
LD_OTF_Th3_Yr48	VALUE_TYPE
LD_ED_Th1_Yr49	VALUE_TYPE
LD_ED_Th2_Yr49	VALUE_TYPE
LD_ED_Th3_Yr49	VALUE_TYPE
LD_OTF_Th1_Yr49	VALUE_TYPE
LD_OTF_Th2_Yr49	VALUE_TYPE
LD_OTF_Th3_Yr49	VALUE_TYPE
LD_ED_Th1_Yr50	VALUE_TYPE
LD_ED_Th2_Yr50	VALUE_TYPE
LD_ED_Th3_Yr50	VALUE_TYPE
LD_OTF_Th1_Yr50	VALUE_TYPE
LD_OTF_Th2_Yr50	VALUE_TYPE
LD_OTF_Th3_Yr50	VALUE_TYPE
LD_ED_Th1_Yr51	VALUE_TYPE
LD_ED_Th2_Yr51	VALUE_TYPE
LD_ED_Th3_Yr51	VALUE_TYPE
LD_OTF_Th1_Yr51	VALUE_TYPE
LD_OTF_Th2_Yr51	VALUE_TYPE
LD_OTF_Th3_Yr51	VALUE_TYPE
LD_ED_Th1_Yr52	VALUE_TYPE
LD_ED_Th2_Yr52	VALUE_TYPE
LD_ED_Th3_Yr52	VALUE_TYPE
LD_OTF_Th1_Yr52	VALUE_TYPE
LD_OTF_Th2_Yr52	VALUE_TYPE
LD_OTF_Th3_Yr52	VALUE_TYPE
LD_ED_Th1_Yr53	VALUE_TYPE
LD_ED_Th2_Yr53	VALUE_TYPE
LD_ED_Th3_Yr53	VALUE_TYPE
LD_OTF_Th1_Yr53	VALUE_TYPE
LD_OTF_Th2_Yr53	VALUE_TYPE
LD_OTF_Th3_Yr53	VALUE_TYPE
LD_ED_Th1_Yr54	VALUE_TYPE
LD_ED_Th2_Yr54	VALUE_TYPE
LD_ED_Th3_Yr54	VALUE_TYPE
LD_OTF_Th1_Yr54	VALUE_TYPE
LD_OTF_Th2_Yr54	VALUE_TYPE

LD_OTF_Th3_Yr54	VALUE_TYPE
LD_ED_Th1_Yr55	VALUE_TYPE
LD_ED_Th2_Yr55	VALUE_TYPE
LD_ED_Th3_Yr55	VALUE_TYPE
LD_OTF_Th1_Yr55	VALUE_TYPE
LD_OTF_Th2_Yr55	VALUE_TYPE
LD_OTF_Th3_Yr55	VALUE_TYPE
LD_ED_Th1_Yr56	VALUE_TYPE
LD_ED_Th2_Yr56	VALUE_TYPE
LD_ED_Th3_Yr56	VALUE_TYPE
LD_OTF_Th1_Yr56	VALUE_TYPE
LD_OTF_Th2_Yr56	VALUE_TYPE
LD_OTF_Th3_Yr56	VALUE_TYPE
LD_ED_Th1_Yr57	VALUE_TYPE
LD_ED_Th2_Yr57	VALUE_TYPE
LD_ED_Th3_Yr57	VALUE_TYPE
LD_OTF_Th1_Yr57	VALUE_TYPE
LD_OTF_Th2_Yr57	VALUE_TYPE
LD_OTF_Th3_Yr57	VALUE_TYPE
LD_ED_Th1_Yr58	VALUE_TYPE
LD_ED_Th2_Yr58	VALUE_TYPE
LD_ED_Th3_Yr58	VALUE_TYPE
LD_OTF_Th1_Yr58	VALUE_TYPE
LD_OTF_Th2_Yr58	VALUE_TYPE
LD_OTF_Th3_Yr58	VALUE_TYPE
LD_ED_Th1_Yr59	VALUE_TYPE
LD_ED_Th2_Yr59	VALUE_TYPE
LD_ED_Th3_Yr59	VALUE_TYPE
LD_OTF_Th1_Yr59	VALUE_TYPE
LD_OTF_Th2_Yr59	VALUE_TYPE
LD_OTF_Th3_Yr59	VALUE_TYPE
LD_ED_Th1_Yr60	VALUE_TYPE
LD_ED_Th2_Yr60	VALUE_TYPE
LD_ED_Th3_Yr60	VALUE_TYPE
LD_OTF_Th1_Yr60	VALUE_TYPE
LD_OTF_Th2_Yr60	VALUE_TYPE
LD_OTF_Th3_Yr60	VALUE_TYPE

5.5.8 System Globals

SG_YRECON	VALUE_TYPE
SG_COSTU	VALUE_TYPE
SG_CurrSym	VALUE_TYPE
SG_Cost_Type	VALUE_TYPE
SG_Metric	VALUE_TYPE

5.5.9 PRICE Enterprise Variables

5.5.9.1 Strings

PX_Import_Str1	STRING_TYPE
PX_Import_Str2	STRING_TYPE
PX_Import_Str3	STRING_TYPE
PX_Import_Str4	STRING_TYPE
PX_Import_Str5	STRING_TYPE
PX_Import_Str6	STRING_TYPE
PX_Import_Str7	STRING_TYPE
PX_Import_Str8	STRING_TYPE
PX_Import_Str9	STRING_TYPE
PX_Import_Str10	STRING_TYPE

5.5.9.2 Integers

PX_Import_Int1	VALUE_TYPE
PX_Import_Int2	VALUE_TYPE
PX_Import_Int3	VALUE_TYPE
PX_Import_Int4	VALUE_TYPE
PX_Import_Int5	VALUE_TYPE
PX_Import_Int6	VALUE_TYPE
PX_Import_Int7	VALUE_TYPE
PX_Import_Int8	VALUE_TYPE
PX_Import_Int9	VALUE_TYPE
PX_Import_Int10	VALUE_TYPE

5.5.9.3 Floating Point

PX_Import_Flt1	VALUE_TYPE
PX_Import_Flt2	VALUE_TYPE
PX_Import_Flt3	VALUE_TYPE
PX_Import_Flt4	VALUE_TYPE
PX_Import_Flt5	VALUE_TYPE

PX_Import_Flt6	VALUE_TYPE
PX_Import_Flt7	VALUE_TYPE
PX_Import_Flt8	VALUE_TYPE
PX_Import_Flt9	VALUE_TYPE
PX_Import_Flt10	VALUE_TYPE

5.5.10 PRICE S Inputs

SIN_Language_1	STRING_TYPE
SIN_SLOC_1	VALUE_TYPE
SIN_FRAC_1	VALUE_TYPE
SIN_CPLX1_1	VALUE_TYPE
SIN_CPLX2_1	VALUE_TYPE
SIN_PROFAC_1	VALUE_TYPE
SIN_APPL_1	VALUE_TYPE
SIN_NEWD_1	VALUE_TYPE
SIN_NEWC_1	VALUE_TYPE
SIN_Language_2	STRING_TYPE
SIN_SLOC_2	VALUE_TYPE
SIN_FRAC_2	VALUE_TYPE
SIN_CPLX1_2	VALUE_TYPE
SIN_CPLX2_2	VALUE_TYPE
SIN_PROFAC_2	VALUE_TYPE
SIN_APPL_2	VALUE_TYPE
SIN_NEWD_2	VALUE_TYPE
SIN_NEWC_2	VALUE_TYPE
SIN_Language_3	STRING_TYPE
SIN_SLOC_3	VALUE_TYPE
SIN_FRAC_3	VALUE_TYPE
SIN_CPLX1_3	VALUE_TYPE
SIN_CPLX2_3	VALUE_TYPE
SIN_PROFAC_3	VALUE_TYPE
SIN_APPL_3	VALUE_TYPE
SIN_NEWD_3	VALUE_TYPE
SIN_NEWC_3	VALUE_TYPE
SIN_Language_4	STRING_TYPE
SIN_SLOC_4	VALUE_TYPE
SIN_FRAC_4	VALUE_TYPE
SIN_CPLX1_4	VALUE_TYPE
SIN_CPLX2_4	VALUE_TYPE

SIN_PROFAC_4	VALUE_TYPE
SIN_APPL_4	VALUE_TYPE
SIN_NEWD_4	VALUE_TYPE
SIN_NEWC_4	VALUE_TYPE
SIN_SCON	VALUE_TYPE
SIN_SRR	VALUE_TYPE
SIN_SDR	VALUE_TYPE
SIN_SSR	VALUE_TYPE
SIN_PDR	VALUE_TYPE
SIN_CDR	VALUE_TYPE
SIN_TRR	VALUE_TYPE
SIN_FCA	VALUE_TYPE
SIN_PCA	VALUE_TYPE
SIN_FQR	VALUE_TYPE
SIN_OTE	VALUE_TYPE
SIN_PLTFM	VALUE_TYPE
SIN_CPLXM	VALUE_TYPE
SIN_INTEGI	VALUE_TYPE
SIN_INTEGE	VALUE_TYPE
SIN_UTIL	VALUE_TYPE

5.5.11 PRICE S Outputs

SOUT_SysCon_Des	VALUE_TYPE
SOUT_SysCon_Prog	VALUE_TYPE
SOUT_SysCon_Data	VALUE_TYPE
SOUT_SysCon_SPM	VALUE_TYPE
SOUT_SysCon_QA	VALUE_TYPE
SOUT_SysCon_Config	VALUE_TYPE
SOUT_SysCon_Total	VALUE_TYPE
SOUT_SysReq_Des	VALUE_TYPE
SOUT_SysReq_Prog	VALUE_TYPE
SOUT_SysReq_Data	VALUE_TYPE
SOUT_SysReq_SPM	VALUE_TYPE
SOUT_SysReq_QA	VALUE_TYPE
SOUT_SysReq_Config	VALUE_TYPE
SOUT_SysReq_Total	VALUE_TYPE
SOUT_SwReq_Des	VALUE_TYPE
SOUT_SwReq_Prog	VALUE_TYPE
SOUT_SwReq_Data	VALUE_TYPE

SOUT_SwReq_SPM	VALUE_TYPE
SOUT_SwReq_QA	VALUE_TYPE
SOUT_SwReq_Config	VALUE_TYPE
SOUT_SwReq_Total	VALUE_TYPE
SOUT_PreDes_Des	VALUE_TYPE
SOUT_PreDes_Prog	VALUE_TYPE
SOUT_PreDes_Data	VALUE_TYPE
SOUT_PreDes_SPM	VALUE_TYPE
SOUT_PreDes_QA	VALUE_TYPE
SOUT_PreDes_Config	VALUE_TYPE
SOUT_PreDes_Total	VALUE_TYPE
SOUT_DetDes_Des	VALUE_TYPE
SOUT_DetDes_Prog	VALUE_TYPE
SOUT_DetDes_Data	VALUE_TYPE
SOUT_DetDes_SPM	VALUE_TYPE
SOUT_DetDes_QA	VALUE_TYPE
SOUT_DetDes_Config	VALUE_TYPE
SOUT_DetDes_Total	VALUE_TYPE
SOUT_CodeTest_Des	VALUE_TYPE
SOUT_CodeTest_Prog	VALUE_TYPE
SOUT_CodeTest_Data	VALUE_TYPE
SOUT_CodeTest_SPM	VALUE_TYPE
SOUT_CodeTest_QA	VALUE_TYPE
SOUT_CodeTest_Config	VALUE_TYPE
SOUT_CodeTest_Total	VALUE_TYPE
SOUT_CSCITest_Des	VALUE_TYPE
SOUT_CSCITest_Prog	VALUE_TYPE
SOUT_CSCITest_Data	VALUE_TYPE
SOUT_CSCITest_SPM	VALUE_TYPE
SOUT_CSCITest_QA	VALUE_TYPE
SOUT_CSCITest_Config	VALUE_TYPE
SOUT_CSCITest_Total	VALUE_TYPE
SOUT_SysTest_Des	VALUE_TYPE
SOUT_SysTest_Prog	VALUE_TYPE
SOUT_SysTest_Data	VALUE_TYPE
SOUT_SysTest_SPM	VALUE_TYPE
SOUT_SysTest_QA	VALUE_TYPE
SOUT_SysTest_Config	VALUE_TYPE
SOUT_SysTest_Total	VALUE_TYPE

SOUT_OperTE_Des	VALUE_TYPE
SOUT_OperTE_Prog	VALUE_TYPE
SOUT_OperTE_Data	VALUE_TYPE
SOUT_OperTE_SPM	VALUE_TYPE
SOUT_OperTE_QA	VALUE_TYPE
SOUT_OperTE_Config	VALUE_TYPE
SOUT_OperTE_Total	VALUE_TYPE
SOUT_Total_Des	VALUE_TYPE
SOUT_Total_Prog	VALUE_TYPE
SOUT_Total_Data	VALUE_TYPE
SOUT_Total_SPM	VALUE_TYPE
SOUT_Total_QA	VALUE_TYPE
SOUT_Total_Config	VALUE_TYPE
SOUT_Total_Total	VALUE_TYPE

5.5.12 PRICE SL Outputs

SLOUT_Des_Maint	VALUE_TYPE
SLOUT_Des_Enhan	VALUE_TYPE
SLOUT_Des_Grow	VALUE_TYPE
SLOUT_Des_Total	VALUE_TYPE
SLOUT_Prog_Maint	VALUE_TYPE
SLOUT_Prog_Enhan	VALUE_TYPE
SLOUT_Prog_Grow	VALUE_TYPE
SLOUT_Prog_Total	VALUE_TYPE
SLOUT_Data_Maint	VALUE_TYPE
SLOUT_Data_Enhan	VALUE_TYPE
SLOUT_Data_Grow	VALUE_TYPE
SLOUT_Data_Total	VALUE_TYPE
SLOUT_SysMgmt_Maint	VALUE_TYPE
SLOUT_SysMgmt_Enhan	VALUE_TYPE
SLOUT_SysMgmt_Grow	VALUE_TYPE
SLOUT_SysMgmt_Total	VALUE_TYPE
SLOUT_QA_Maint	VALUE_TYPE
SLOUT_QA_Enhan	VALUE_TYPE
SLOUT_QA_Grow	VALUE_TYPE
SLOUT_QA_Total	VALUE_TYPE
SLOUT_Config_Maint	VALUE_TYPE
SLOUT_Config_Enhan	VALUE_TYPE
SLOUT_Config_Grow	VALUE_TYPE

SLOUT_Config_Total	VALUE_TYPE
SLOUT_Total_Maint	VALUE_TYPE
SLOUT_Total_Enhan	VALUE_TYPE
SLOUT_Total_Grow	VALUE_TYPE
SLOUT_Total_Total	VALUE_TYPE

5.5.13 PRICE S Deployment

SD_Title	STRING_TYPE
SD_START	VALUE_TYPE
SD_END	VALUE_TYPE
SD_INSTALL	VALUE_TYPE
SD_GLEVEL	VALUE_TYPE
SD_ELEVEL	VALUE_TYPE
SD_QLEVEL	VALUE_TYPE
SD_MPROFAC	VALUE_TYPE
SD_EPROFAC	VALUE_TYPE
SD_GPROFAC	VALUE_TYPE

5.5.14 PRICE M Input

MIN_ID	STRING_TYPE
MIN_QTY	VALUE_TYPE
MIN_PROTOS	VALUE_TYPE
MIN_LENGTH	VALUE_TYPE
MIN_WIDTH	VALUE_TYPE
MIN_LAYERS	VALUE_TYPE
MIN_PLTFM	VALUE_TYPE
MIN_BTYPE	VALUE_TYPE
MIN_BSIDES	VALUE_TYPE
MIN_BWT	VALUE_TYPE
MIN_BCOST	VALUE_TYPE
MIN_PTYPE	VALUE_TYPE
MIN_PPINS	VALUE_TYPE
MIN_PWT	VALUE_TYPE
MIN_PKCOST	VALUE_TYPE
MIN_INTEGE	VALUE_TYPE
MIN_HSINT	VALUE_TYPE
MIN_WT	VALUE_TYPE
MIN_VOL	VALUE_TYPE
MIN_LOTQTY	VALUE_TYPE

MIN_ECMPLX	VALUE_TYPE
MIN_NEWDES	VALUE_TYPE
MIN_DESRPT	VALUE_TYPE
MIN_ACOST	VALUE_TYPE
MIN_TCOST	VALUE_TYPE
MIN_DSTART	VALUE_TYPE
MIN_DFPRO	VALUE_TYPE
MIN_DLPRO	VALUE_TYPE
MIN_DBINDX	VALUE_TYPE
MIN_PSTART	VALUE_TYPE
MIN_PFAD	VALUE_TYPE
MIN_PEND	VALUE_TYPE
MIN_MBINDX	VALUE_TYPE
MIN_MAUTO	VALUE_TYPE
MIN_MMAT	VALUE_TYPE
MIN_YRTECH	VALUE_TYPE
MIN_CostType	VALUE_TYPE
MIN_YRBASE	VALUE_TYPE
MIN_QTYNHA	VALUE_TYPE
MIN_CDFRAC	VALUE_TYPE
MIN_PINS	VALUE_TYPE
MIN_GATES	VALUE_TYPE
MIN_XSTRS	VALUE_TYPE
MIN_DPLTFM	VALUE_TYPE
MIN_SPLTFM	VALUE_TYPE
MIN_DINDEX	VALUE_TYPE
MIN_NEWCEL	VALUE_TYPE
MIN_CADFAC	VALUE_TYPE
MIN_ITERAT	VALUE_TYPE
MIN_PROFAC	VALUE_TYPE
MIN_MINDEX	VALUE_TYPE
MIN_PKGFAC	VALUE_TYPE
MIN_SUBFAC	VALUE_TYPE
MIN_WSIZE	VALUE_TYPE
MIN_FSIZE	VALUE_TYPE
MIN_CPYLD	VALUE_TYPE
MIN_ASMYLD	VALUE_TYPE
MIN_OVLYLD	VALUE_TYPE
MIN_MSKLVL	VALUE_TYPE

MIN_DEFDEN	VALUE_TYPE
MIN_PTEND	VALUE_TYPE
MIN_TSTEND	VALUE_TYPE
MIN_DEND	VALUE_TYPE
MIN_PPEND	VALUE_TYPE

5.5.15 Component Database

CDB_CNAME	STRING_TYPE
CDB_CELM	VALUE_TYPE
CDB_CTYPE	VALUE_TYPE
CDB_CPKG	VALUE_TYPE
CDB_CPINS	VALUE_TYPE
CDB_CWT	VALUE_TYPE
CDB_CCOST	VALUE_TYPE
CDB_CPERQ	VALUE_TYPE
CDB_PLTFM	VALUE_TYPE
CDB_YRBASE	VALUE_TYPE

5.5.16 MCPLXS Generator

CPXS_Title	STRING_TYPE
CPXS_Precision	VALUE_TYPE
CPXS_Maturity	VALUE_TYPE
CPXS_PLTFM	VALUE_TYPE
CPXS_PercOfTotal	VALUE_TYPE
CPXS_Machine	VALUE_TYPE
CPXS_NumParts	VALUE_TYPE
CPXS_CalFactor	VALUE_TYPE
CPXS_CalMCPLXS	VALUE_TYPE
CPXS_Distance	VALUE_TYPE
CPXS_Hogout	VALUE_TYPE
CPXS_SurfFin	VALUE_TYPE
CPXS_PercFin	VALUE_TYPE
CPXS_Weight	VALUE_TYPE

5.5.17 Function Parameters

FUNC_Param1	VALUE_TYPE
FUNC_Param2	VALUE_TYPE
FUNC_Param3	VALUE_TYPE
FUNC_Param4	VALUE_TYPE

FUNC_Param5	VALUE_TYPE
FUNC_Param6	VALUE_TYPE
FUNC_Param7	VALUE_TYPE
FUNC_Param8	VALUE_TYPE
FUNC_Param9	VALUE_TYPE
FUNC_Param10	VALUE_TYPE
FUNC_Param11	VALUE_TYPE
FUNC_Param12	VALUE_TYPE
FUNC_Param13	VALUE_TYPE
FUNC_Param14	VALUE_TYPE
FUNC_Param15	VALUE_TYPE
FUNC_Param16	VALUE_TYPE
FUNC_Param17	VALUE_TYPE
FUNC_Param18	VALUE_TYPE
FUNC_Param19	VALUE_TYPE
FUNC_Param20	VALUE_TYPE
FUNC_Param21	VALUE_TYPE
FUNC_Param22	VALUE_TYPE
FUNC_Param23	VALUE_TYPE
FUNC_Param24	VALUE_TYPE
FUNC_Param25	VALUE_TYPE
FUNC_Param26	VALUE_TYPE
FUNC_Param27	VALUE_TYPE
FUNC_Param28	VALUE_TYPE
FUNC_Param29	VALUE_TYPE
FUNC_Param30	VALUE_TYPE
FUNC_Param31	VALUE_TYPE
FUNC_Param32	VALUE_TYPE
FUNC_Param33	VALUE_TYPE
FUNC_Param34	VALUE_TYPE
FUNC_Param35	VALUE_TYPE
FUNC_Param36	VALUE_TYPE
FUNC_Param37	VALUE_TYPE
FUNC_Param38	VALUE_TYPE
FUNC_Param39	VALUE_TYPE
FUNC_Param40	VALUE_TYPE
FUNC_Param41	VALUE_TYPE
FUNC_Param42	VALUE_TYPE
FUNC_Param43	VALUE_TYPE

FUNC_Param44	VALUE_TYPE
FUNC_Param45	VALUE_TYPE
FUNC_Param46	VALUE_TYPE
FUNC_Param47	VALUE_TYPE
FUNC_Param48	VALUE_TYPE
FUNC_Param49	VALUE_TYPE
FUNC_Param50	VALUE_TYPE
FUNC_Param51	VALUE_TYPE
FUNC_Param52	VALUE_TYPE
FUNC_Param53	VALUE_TYPE
FUNC_Param54	VALUE_TYPE
FUNC_Param55	VALUE_TYPE
FUNC_Param56	VALUE_TYPE
FUNC_Param57	VALUE_TYPE
FUNC_Param58	VALUE_TYPE
FUNC_Param59	VALUE_TYPE
FUNC_Param60	VALUE_TYPE
FUNC_Param61	VALUE_TYPE
FUNC_Param62	VALUE_TYPE
FUNC_Param63	VALUE_TYPE
FUNC_Param64	VALUE_TYPE
FUNC_Param65	VALUE_TYPE
FUNC_Param66	VALUE_TYPE
FUNC_Param67	VALUE_TYPE
FUNC_Param68	VALUE_TYPE
FUNC_Param69	VALUE_TYPE
FUNC_Param70	VALUE_TYPE
FUNC_Param71	VALUE_TYPE
FUNC_Param72	VALUE_TYPE
FUNC_Param73	VALUE_TYPE
FUNC_Param74	VALUE_TYPE
FUNC_Param75	VALUE_TYPE
FUNC_Param76	VALUE_TYPE
FUNC_Param77	VALUE_TYPE
FUNC_Param78	VALUE_TYPE
FUNC_Param79	VALUE_TYPE
FUNC_Param80	VALUE_TYPE
FUNC_Param81	VALUE_TYPE
FUNC_Param82	VALUE_TYPE

FUNC_Param83	VALUE_TYPE
FUNC_Param84	VALUE_TYPE
FUNC_Param85	VALUE_TYPE
FUNC_Param86	VALUE_TYPE
FUNC_Param87	VALUE_TYPE
FUNC_Param88	VALUE_TYPE
FUNC_Param89	VALUE_TYPE
FUNC_Param90	VALUE_TYPE
FUNC_Param91	VALUE_TYPE
FUNC_Param92	VALUE_TYPE
FUNC_Param93	VALUE_TYPE
FUNC_Param94	VALUE_TYPE
FUNC_Param95	VALUE_TYPE
FUNC_Param96	VALUE_TYPE
FUNC_Param97	VALUE_TYPE
FUNC_Param98	VALUE_TYPE
FUNC_Param99	VALUE_TYPE
FUNC_Param100	VALUE_TYPE

5.5.18 Global Variables

GLB_Counter	VALUE_TYPE

Index