SAS® Inventory Optimization 1.3
User's Guide

The Power to Know®

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2006. *SAS® Inventory Optimization 1.3: User's Guide*. Cary, NC: SAS Institute Inc.

SAS® Inventory Optimization 1.3: User's Guide

ISBN-13: 978-1-59047-926-1
ISBN-10: 1-59047-926-2

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, April 2006

SAS Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Contents

Acknowledgments

Credits

Documentation

Writing	Tugrul Sanli, Jinxin Yi, Michelle Opp, Rob Pratt, Charles B. Kelly, Heidi Porter
Editing	Virginia Clark
Documentation Support	Tim Arnold, Michelle Opp
Review	Feng Chen, Radhika V. Kulkarni, Michelle Opp, Bengt Pederson, Rob Pratt

Software

SAS Inventory Optimization was implemented by the Operations Research and Development Department. Substantial support was given to the project by other members of the Analytical Solutions Division.

IRP Procedure	Tugrul Sanli
MIRP Procedure	Jinxin Yi

Support Groups

Software Testing	Hilmi Aydin, Bengt Pederson
Technical Support	Tonya Chapman

Acknowledgments

Many people have been instrumental in the development of SAS Inventory Optimization. We would like to thank Duke University Professor Paul Zipkin for his invaluable comments.

The final responsibility for the SAS System lies with SAS Institute alone. We hope that you will always let us know your opinions about the SAS System and its documentation. It is through your participation that SAS software is continuously improved.

What's New in SAS Inventory Optimization 1.3

Overview

The SAS Inventory Replenishment Planning product has a new name, SAS Inventory Optimization. SAS Inventory Optimization also has a new version numbering scheme. SAS Inventory Optimization 1.3 provides more features and greater functionality than SAS Inventory Replenishment Planning 1.2 and SAS 9.1.2 Inventory Replenishment Planning.

SAS Inventory Optimization enables you to calculate periodic-review inventory replenishment policies using information about demand, lead time, costs, and desired service measures.

SAS Inventory Replenishment Planning 9.1.2 included the experimental MIRP procedure for calculating inventory control parameters in multiechelon supply chains. The MIRP procedure is designed to help users better understand how network topologies, cost structures, and service level requirements, among other factors, impact the inventory investment and allocation across a supply chain.

SAS Inventory Replenishment Planning 1.2 included

- new options for the IRP procedure to give the user more control over calculating policies
- improvements in the IRP algorithms
- production-level MIRP procedure, which replaces the experimental MIRP procedure
- the new SAS Inventory Policy Studio solution, which provides a user-friendly interface to the IRP procedure

SAS Inventory Optimization 1.3 includes

- new features in the MIRP procedure
- the production-level version of SAS Inventory Policy Studio, which provides a user-friendly interface to the IRP procedure

PROC IRP

PROC IRP can calculate four types of replenishment policies. These policies are determined through a number of algorithms that are controlled by user-specified options. PROC IRP can accommodate both single-location and two-echelon distribution inventory systems.

The input data set to PROC IRP specifies information about lead time, demand, and costs, as well as options to control the policy. The output data set produced by PROC IRP gives the policy parameters for each item. In addition, estimates of measures such as fill rate, ready rate, and average inventory, among others, are included in the output data set.

With SAS Inventory Replenishment Planning 1.2 and SAS Inventory Optimization 1.3, PROC IRP includes new and improved algorithms. In addition, new options give the user more control over optimization of policies:

- ALGORITHM= option
- QGRID= option
- DIST= option

PROC MIRP

SAS Inventory Replenishment Planning 1.2 and higher includes the production-level MIRP procedure. PROC MIRP can calculate inventory control parameters for all stock-keeping locations in general supply chain networks, which may consist of any combination of serial, assembly, and distribution subnetworks. These parameters ensure that service requirements are satisfied at minimum inventory cost for every supply chain. PROC MIRP gives users flexibility in modeling their supply chain networks. These functionalities include:

- two replenishment policies: base-stock policy and min-max policy
- four service-level measures: ready rate, fill rate, backorder ratio, and waiting time probability
- order constraints: minimum size, maximum size, and fixed lot size
- multiple-period replenishment planning
- backlog and lost sales options
- intermittent demand

The input data sets to PROC MIRP specify information about lead times, holding costs, service levels, and demand of stock-keeping location in the networks, as well as information about the structure of the networks. The output data set produced by PROC MIRP gives the inventory control parameters, along with estimates of measures such as order quantities, backlogs, and service levels, for each stock-keeping location and each period in a planning horizon. PROC MIRP can also be used to evaluate the performance of users' own inventory policies.

In SAS Inventory Optimization 1.3, PROC MIRP includes support for random lead time. This new functionality enables users to capture uncertainties in lead times between locations within a supply chain network, and from external suppliers to any locations in the network. Users specify the minimum, average, and maximum values for lead times. PROC MIRP uses these values in a triangular distribution to model the lead time uncertainty.

SAS Inventory Policy Studio

SAS Inventory Optimization 1.2 included a preproduction version of SAS Inventory Policy Studio. SAS Inventory Optimization 1.3 includes production-level SAS Inventory Policy Studio, a graphical user interface designed to support the use of PROC IRP for users new to SAS or unfamiliar with the SAS language. In addition to supporting the optimization of inventory policies, Inventory Policy Studio

- facilitates the creation and management of projects, enabling users to plan inventory policies for specific categories of products
- enables users to create "what if" scenarios using different parameters or forecasts, and determine the impact on policies, customer service levels, and costs
- provides a graphical sensitivity analysis to assist users in quickly gauging the impact that changes in specific parameters will have on policies, customer service levels, and costs
- provides reporting capabilities
- provides the SAS Inventory Optimization Script Runner, which enables users to write and run or schedule batch scripts for certain SAS Inventory Policy Studio tasks

After creating scenarios, and calculating, evaluating, and promoting policies, the user can export the project table for use in other operational systems.

SAS Inventory Policy Studio does not support all of the capabilities of PROC IRP. For example, SAS Inventory Policy Studio

- does not support the use of backorder penalty costs as input to policy calculations
- does not support policy calculation for two-echelon systems

For information about SAS Inventory Policy Studio, see the *SAS Inventory Optimization 1.3: Inventory Policy Studio User's Guide*.

Chapter 1
Introduction to SAS Inventory Optimization

Chapter Contents

Chapter 1
Introduction to SAS Inventory Optimization

Welcome to SAS Inventory Optimization and SAS Inventory Policy Studio

The SAS Inventory Optimization product provides two ways of calculating inventory replenishment policies:

IRP and MIRP Procedures The IRP procedure and the MIRP procedure are procedures that provide the ability to transform raw demand transaction data and order lead time estimates into policies for managing product inventory levels. PROC IRP can calculate four types of replenishment policies. PROC MIRP can calculate optimal inventory investment policies using two types of replenishment policies for multi-echelon supply chains and four types of service level requirements.

The *SAS Inventory Optimization User's Guide* contains documentation about the IRP and MIRP procedures.

SAS Inventory Policy Studio SAS Inventory Policy Studio is a Java client application that provides a graphical user interface designed to support the use of PROC IRP for users new to SAS or unfamiliar with the SAS language. SAS Inventory Policy Studio enables users to analyze demand, cost, and lead time data, and calculate inventory replenishment policies for single-location inventory systems.

SAS Inventory Policy Studio uses the SAS Metadata Server and SAS Workspace Server of the SAS Intelligence Platform. SAS Inventory Policy Studio also uses SAS Analytics Platform. For information about how SAS Inventory Policy Studio works with the SAS Intelligence Platform and SAS Analytics Platform, refer to the *SAS Inventory Policy Studio Administrator's Guide*.

The *SAS Inventory Policy Studio User's Guide* contains documentation about SAS Inventory Policy Studio usage.

You can use PROC IRP, PROC MIRP, or SAS Inventory Policy Studio to calculate optimal replenishment policies that minimize the average costs associated with ordering, holding, and backordering products. PROC IRP, PROC MIRP, and SAS Inventory Policy Studio all calculate

- ordering information, such as when to order, how much to order, and ordering frequency

- service level information
- inventory levels and costs

For an overview of inventory optimization and summaries and comparisons of SAS Inventory Policy Studio, the IRP procedure, and the MIRP procedure functionality, see "Overview of Inventory, SAS Inventory Policy Studio, and SAS Inventory Optimization."

Overview of Inventory, SAS Inventory Policy Studio, and SAS Inventory Optimization

Impact of Inventory

In many industries, inventory is a large part of conducting business:

- **In the manufacturing industry**, it is necessary to coordinate both inventory-producing and inventory-consuming activities.
- **In the retail industry**, companies maintain large volumes of different items at various locations and must monitor quantities, estimate usage, and place orders for replenishment. Slow-moving items are discontinued, while new items are introduced.
- **In the service industry**, inventories are critical in providing the services that customers require. For instance, where would the hospital industry be without adequate supplies of surgical instruments and medicines? And how would a major package delivery company function without an inventory of trucks and spare parts?

The scope of inventory-dependent operations is large. Tracking inventory is essential to running a business efficiently, and managing inventory effectively can have a big impact on profitability:

- Customers often will not tolerate product unavailability or delays in delivery. In some cases, a shortage may be only a small inconvenience (such as selecting a different video at the rental store), while sometimes it may cause a severe problem (such as interrupting production-line activity at a computer manufacturer). On some occasions, sporadic shortages can be expected, but frequent shortages may ultimately erode a company's reputation and reduce their market share.
- Overabundant, slow-moving inventories can place a serious strain on a company's available capital and the company's ability to take advantage of financial opportunities.

Therefore, in order to compete effectively in today's business world, it is imperative that adequate inventories are maintained efficiently.

Function of Inventory

For many businesses, inventory is the foundation of conducting business. Businesses hold inventory in order to balance issues related to ordering, producing, and selling goods. Holding inventory enables businesses to

- create buffers against uncertainties of supply and demand (selling)
- take advantage of lower purchasing and transportation costs associated with high volumes (ordering)
- take advantage of economies of scales associated with manufacturing products in batches (producing)
- build up reserves for seasonal demand or promotional sales (selling)

However, uncertainty in supply and demand makes inventory decisions difficult due to the following issues:

- supply:
  - economies of scale (production and delivery)
  - capacity limits (production and delivery)
  - delays in replenishment (order lead time)
- demand:
  - steady or intermittent demand
  - variations in demand over time (trend, seasonality)
  - unpredictable demand variations (random)

In addition, product diversification increases the number of parts to control. When dealing with uncertainty, the traditional objective of inventory control models is to minimize expected costs. Consider some of the costs associated with most inventory control systems:

- fixed ordering cost (or replenishment cost)
  - cost of processing orders
  - cost independent of replenishment quantity
- holding cost
  - opportunity cost of capital invested in inventory
  - warehousing cost
  - handling and counting costs
  - other costs such as insurance and taxes
- stockout cost

  - cost of backordering
  - penalty cost for lost sales

Thus, both inventory shortages and inventory costs have a direct relationship to profitability. Accordingly, effective management of inventory can determine the success of a business.

Optimization of Inventory

To effectively manage inventory, businesses must do the following:

- improve product availability and reduce lost sales
- release working capital and improve inventory turns
- balance the need for customer service with the costs of maintaining inventory

To effectively manage inventory, businesses must also optimize the costs of buying, holding, producing, and selling inventory. Table 1.1 shows the business category, the cost name, the issue that must be weighed for optimization, and the policy outcome of that optimization.

Table 1.1. Costs and Issues for Inventory Policies

Category	Costs	Balancing Issue	Policy
Ordering Production	Fixed ordering/unit costs	Volume purchase Long production runs and large raw material inventory	When should orders be placed to restock inventory?
Holding	Holding cost	Low inventory and high turnover versus availability: uncertainty of supply and demand	How should inventory be replenished (how much should be ordered) to reduce costs and increase turns?
Sales	Stock-out*	Customer service	What will demand be? What is the projected customer service level?

***Note:** SAS Inventory Policy Studio does not currently use production or stock-out (penalty costs) to calculate inventory policy results and metrics (using the IRP procedure). Instead, you can specify a service measure that penalizes backorders in different ways.

You can use SAS Inventory Policy Studio (which uses the IRP procedure), or the IRP or MIRP procedures to optimize costs, and to provide answers to the corresponding questions that are used to effectively manage inventory.

Summary of SAS Inventory Policy Studio Functionality for Optimizing Inventory

To optimize costs, and provide policies for managing inventory, SAS Inventory Policy Studio uses the costs, historical or forecasted demand data, order lead times, and desired policy restrictions, along with the IRP procedure to calculate the appropriate inventory policies for single location inventory systems.

These calculations include answers to two fundamental inventory replenishment questions:

- When should orders be placed to restock inventory?
- How much should be ordered?

SAS Inventory Policy Studio also uses the IRP procedure to calculate average costs and project service levels. SAS Inventory Policy Studio uses a subset of the functionality of the IRP procedure to perform these calculations. SAS Inventory Policy Studio only supports policy calculation for single location inventory systems, and does not support the use of backorder penalty costs (stockout costs).

SAS Inventory Policy Studio provides a graphical user interface to enable easy viewing and calculation of inventory policies for single location inventory systems.

To enable users to easily analyze and optimize data, SAS Inventory Policy Studio provides the following features:

- the ability to import data from an external data set into the SAS Inventory Policy Studio application
- the ability to filter data into different scenarios for use in calculating policies for different subsets of data and parameters
- the ability to edit the parameters in the scenarios
- the ability to create a sensitivity analysis that enables you to see the effects of varying the value of a parameter
- table and detail views, to enable flexibility when viewing and editing data
- the ability to export SAS Inventory Policy Studio data for use in other operational systems
- reporting capabilities
- the SAS Inventory Optimization Script Runner, which enables users to write and run or schedule batch scripts for certain SAS Inventory Policy Studio tasks.

SAS Inventory Policy Studio uses the underlying IRP procedure to calculate policies using a number of algorithms that are controlled by policy restriction variables that you can specify. PROC IRP can calculate four types of replenishment policies. For more information about PROC IRP and replenishment policies, see Chapter 2, “The IRP Procedure.”

SAS Inventory Policy Studio uses the SAS Metadata Server and SAS Workspace Server of the SAS Intelligence Architecture. For details about the SAS Inventory Policy Studio architecture and SAS Inventory Policy Studio administration, refer to the *SAS Inventory Policy Studio Administrator's Guide*.

For details about setting up and using SAS Inventory Policy Studio, refer to Chapter 2, "Using SAS Inventory Policy Studio," (*SAS Inventory Optimization: Inventory Policy Studio User's Guide*) .

Summary of PROC IRP and PROC MIRP Functionality for Optimizing Inventory

To optimize inventory, the IRP and MIRP procedures provide essential aid to decision making by answering two fundamental questions:

- When should orders be placed to restock inventory?
- How much should be ordered?

The IRP and MIRP procedures provide the ability to transform raw demand transaction data and order lead time estimates into rules for managing product inventory levels:

- **For single-location or two-echelon inventory distribution systems**, the IRP procedure uses estimates of review-time demand and replenishment order lead time along with the associated inventory costs for ordering, holding, and stockouts, to calculate optimal policies. If the stockout penalty cost is unknown, one of several service measures can be substituted and the IRP procedure can calculate nearly optimal policies. In both cases, PROC IRP provides an estimate of service measures for the purpose of evaluating projected policy performance. For details, see Chapter 2, "The IRP Procedure."
- **For multi-echelon supply chains**, the MIRP procedure calculates optimal inventory investment decisions using two types of replenishment policies for multi-echelon supply chains and four types of service level requirements. In addition, PROC MIRP provides estimates of measures such as order quantities, backlogs, and inventory costs. For details, see Chapter 3, "The MIRP Procedure."

Comparison of SAS Inventory Policy Studio, PROC IRP, and PROC MIRP

To enable you to compare the functionality of SAS Inventory Policy Studio, and the IRP and MIRP Procedures, Table 1.2 shows the types of inventory systems, input parameters, and features supported by each product or procedure:

Table 1.2. Comparison of Functionality Differences for SAS Inventory Policy Studio, the IRP Procedure, and the MIRP Procedure

Feature	SAS Inventory Policy Studio	IRP Procedure	MIRP Procedure
Supports single location systems	X	X	X
Supports two-echelon systems		X	X
Supports multi-echelon systems (>2)			X
Can use backorder penalty (stockout) costs to calculate policy results and metrics		X	
Provides a user-friendly interface for easy filtering and analysis of data	X		
Requires SAS Intelligence and SAS Analytics Platforms	X		
Supports sensitivity analysis	X		
Provides reporting capabilities	X		
Provides batch script tool (SAS Inventory Optimization Script Runner)	X		
Provides evaluation of existing policies			X

Using This Documentation

Purpose

This guide provides information about how to use the IRP and MIRP procedures, including

- overview information about inventory processes, and about how the procedures might be used to solve a variety of problems
- reference information for procedures
- examples

Intended Audience

The User's Guide is for users who need to do either of the following:

- write code for the IRP and MIRP procedures
- understand the parameters or policy algorithms used by the IRP or MIRP procedure

Organization

The User's Guide is organized as follows:

Introduction provides a brief description of the guide and summarizes related SAS products and services. This section also discusses inventory issues and outlines the major functions and features of the software.

PROC IRP provides information for coding the IRP procedure and examples that implement the IRP procedure

PROC MIRP provides information for coding the MIRP procedure and examples that implement the MIRP procedure

Typographical Conventions

This document uses several type styles for presenting information. The following list explains the meaning of the typographical conventions used in this document:

roman
: is the standard type style used for most text.

UPPERCASE ROMAN
: is used for SAS statements, options, and other SAS language elements when they appear in the text. However, you can enter these elements in your own SAS programs in lowercase, uppercase, or a mixture of the two.

UPPERCASE BOLD
: is used in the "Syntax" sections' initial lists of SAS statements and options.

oblique
: is used for user-supplied values for options in the syntax definitions. In the text, these values are written in *italic*.

bold
: is used to refer to matrices and vectors.

italic
: is used for terms that are defined in the text, for emphasis, and for references to publications.

`monospace`
is used for names of variables, data sets, and example code when they appear in the text. In most cases, this book uses lowercase type for SAS code.

`monospace bold`
is used for URL and path names.

Where to Go for More Information

If you need more information about your software, then refer to the following sources of information:

- *SAS Inventory Optimization: Inventory Policy Studio Administrator's Guide*
- *SAS Inventory Optimization: Inventory Policy Studio User's Guide*

SAS Institute Technical Support Services

As with all SAS Institute products, the SAS Institute Technical Support staff is available to respond to problems and answer technical questions.

References

Graves, S. C., Rinnooy Kan, A. H. G., and Zipkin, P. H. (Editors) (1993), *Logistics of Production and Inventory*, *Handbooks in Operations Research and Management Science*, Vol. 4, Netherlands: North-Holland.

Schneider, H. (1978), "Methods for Determining the Re-order Point of an (s, S) Ordering Policy When a Service Level is Specified," *J. Opl. Res. Soc.*, 29 (12), 1181–1193.

Schneider, H. (1981), "Effects of Service-Levels on Order-Points or Order-Levels in Inventory Models," *Int. J. Prod. Res.*, 19 (6), 615–631.

Schneider, H. and Ringuest, J.L. (1990), "Power Approximation for Computing (s, S) Policies Using Service Level," *Management Science*, 36 (7), 822–834.

Silver, E. A., Pyke, D. F., and Peterson, R. (1998), *Inventory Management and Production Planning and Scheduling*, New York: John Wiley and Sons, Inc.

Tijms, H. and Groenevelt, H. (1984), "Simple Approximations for the Reorder Point in Periodic and Continuous Review (s, S) Inventory Systems with Service Level Constraints," *European Journal of Operations Research*, 17, 175–190.

Zheng, Y. and Chen, F. (1992), "Inventory Policies with Quantized Ordering," *Naval Research Logistics*, 39, 285–305.

Zheng, Y. and Federgruen, A. (1992), "Finding Optimal (s, S) Policies Is About as Simple as Evaluating a Single Policy," *Operations Research*, 39 (4), 654–665.

Zipkin, P. H. (2000), *Foundations of Inventory Management*, New York: McGraw-Hill.

Chapter 2
The IRP Procedure

Chapter Contents

Chapter 2
The IRP Procedure

Overview

The IRP procedure provides the ability to calculate periodic-review inventory replenishment policies for single-location and two-echelon distribution inventory systems. These policies are determined through a number of algorithms that are controlled by user-specified options.

PROC IRP can calculate four types of replenishment policies that are different types of (s, S) or (s, NQ) policies. For details, see the "Replenishment Policies" section on page 36.

An *optimal* policy is defined as a policy that minimizes the average cost — the total of ordering, holding, and backorder penalty costs. PROC IRP uses several heuristic algorithms to approximate optimal policies to meet the user-specified service constraints. If the penalty cost information is available, PROC IRP can also calculate optimal policies for single-location inventory systems.

Getting Started

Single-Location Inventory Systems

In a single-location inventory system, customers (or demand transactions) request a random amount of an item (SKU). Customer orders are filled from on-hand inventory. If insufficient inventory is available, the order is filled partially with available inventory and any unsatisfied portion is backlogged (or backordered). The *inventory position*, which is on-hand inventory plus inventory-on-order minus backorders, is monitored periodically. Based on the current inventory position, the replenishment policy will determine whether or not a replenishment order should be placed from an outside supplier.

Periodic review is the most common type of review process. Inventory is counted or evaluated periodically (say, monthly) at discrete points in time to determine if a replenishment order needs to be placed. Replenishment decisions can be made only at those points. The time between two review points is called the *review period*.

When a replenishment order is placed, there may be a delay between when the order is placed and when the order arrives. This delay is called the *lead time*, and is specified in the same units as the review period. For example, if the review period is one day (i.e., inventory is reviewed daily) and the lead time is one week, the lead time would be specified as seven days. The IRP procedure accounts for demand that occurs during the lead time.

The size of the demand that occurs during one review period is called the *review-time demand*. When demand is stationary (that is, demand stays relatively constant across review periods), PROC IRP requires only the mean and variance of review-time demand. For example, these values may be estimates calculated using a forecast engine prior to invoking PROC IRP. When demand is not stationary, information must be provided to PROC IRP about the lead-time demand rather than the review-time demand; see the "LEADTIMEDEMAND Statement" section on page 26 for more information about lead-time demand.

PROC IRP calculates inventory replenishment policies using this information — inventory position, lead time, and review-time demand — together with user-specified inventory-related costs and policy restrictions.

As a simple example, consider a single store that carries five different items (SKUs), which are ordered from an outside supplier. Calculation of demand forecasts and inventory review is done weekly. The manager wants to calculate (s, S) policies that will minimize expected holding and ordering costs and achieve a target fill rate of 95%. Table 2.1 summarizes the demand, lead time, and cost information for these items. Note that the lead times are expressed in terms of weeks (either one, two, or three weeks), because the review period is one week.

Table 2.1. Data Summary

Sku	Holding Cost	Ordering Cost	Lead Time	Mean of Demand	Variance of Demand
A	0.35	90	1	125.1	2170.8
B	0.05	50	2	140.3	1667.7
C	0.12	50	3	116.0	3213.4
D	0.10	75	1	291.8	5212.4
E	0.45	75	2	134.5	1980.5

This information is stored in a data set called skuInfo and displayed in Figure 2.1. The mean and variance of one-period demand are given by the RTDmean and RTDvar variables. The lead time is fixed (that is, it has zero variance) and is given by the LTmean variable. Similarly, holding and ordering costs are given by the holdingCost and fixedCost variables. Finally, the serviceLevel variable specifies the desired service level.

Input Data Set

Obs	sku	holding Cost	fixed Cost	LTmean	RTDmean	RTDvar	service Level
1	A	0.35	90	1	125.1	2170.8	0.95
2	B	0.05	50	2	140.3	1667.7	0.95
3	C	0.12	50	3	116.0	3213.4	0.95
4	D	0.10	75	1	291.8	5212.4	0.95
5	E	0.45	75	2	134.5	1980.5	0.95

Figure 2.1. Input Data Set skuInfo

The following IRP procedure call can be used to calculate the inventory policies.

```
proc irp data=skuInfo out=policy;
   itemid sku;
   holdingcost holdingCost;
   leadtime / mean=LTmean;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel;
run;
```

The REVIEWTIMEDEMAND statement specifies the variables that contain the mean and variance of review-time demand. Similarly the LEADTIME statement identifies the variable containing the lead time, and the SERVICE statement identifies the variable that specifies the desired service levels. Fill rate is the default service measure and (s, S) policies are the default policy type, so no extra options or statements are needed. The variables RTDmean, RTDvar, LTmean, fixedCost, holdingCost, and serviceLevel are all default variable names, so you do not need to specify them in any statements. Thus, the following IRP procedure call would have produced the same results:

```
proc irp data=skuInfo out=policy;
   itemid sku;
run;
```

The output data set policy is displayed in Figure 2.2.

Policy Data Set

Obs	sku	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost	inventory Ratio
1	A	211	463	133.739	6.0735	0.43646	86.0905	1.06906
2	B	335	842	229.279	6.8941	0.29107	26.0175	1.63420
3	C	470	792	216.028	6.0123	0.34361	43.1037	1.86231
4	D	432	1074	282.873	14.4130	0.42098	59.8611	0.96941
5	E	382	597	131.757	6.6193	0.50730	97.3379	0.97961

Obs	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	0.048550	0.93540	0.95155	0.87037	FR-SS-NO	SUCCESSFUL
2	0.049138	0.61192	0.95139	0.88256	FR-SS-NO	SUCCESSFUL
3	0.051830	0.53697	0.95122	0.90925	FR-SS-NO	SUCCESSFUL
4	0.049393	1.03156	0.95062	0.85239	FR-SS-NO	SUCCESSFUL
5	0.049214	1.02082	0.95109	0.86866	FR-SS-NO	SUCCESSFUL

Figure 2.2. Policy Output Data Set

The reorderLevel variable gives the reorder level, s, and the orderUpToLevel variable gives the order-up-to level, S. For example, for sku A, any time the inventory position is observed to be less than or equal to 211 at a review point, a replenishment order is placed to bring the inventory position up to 463. The _status_ variable indicates that the optimization was successful for all observations. The _algorithm_ variable gives information about the algorithm used; namely, a fill rate ('FR') service level heuristic was used to calculate (s, S) policies ('SS'), using a normal distribution ('NO') for lead-time demand and (lead time + review time)-demand. The remaining variables report several estimated inventory metrics to evaluate the performance of the underlying policy (see the "OUT= Data Set" section on page 32 for information about these variables).

Two-Echelon Distribution Inventory Systems

A two-echelon distribution inventory system consists of a single warehouse and multiple retail locations. The retail locations do not incur a fixed cost when ordering from the warehouse; therefore, the retail locations follow a base-stock policy. The warehouse, however, incurs a fixed cost when ordering from an outside supplier; the warehouse can therefore follow an (s, S) or (s, nQ) policy. PROC IRP can find nearly optimal policies for two-echelon distribution inventory systems with different service constraints on the retail locations.

Consider a warehouse-retailer distribution problem with two items. For sku A, the warehouse is in Raleigh, NC, and the retail locations are located in Atlanta, GA, Baltimore, MD, and Charleston, SC. For sku B, the warehouse is in Greensboro, NC, and the retail locations are in Atlanta, GA, and Charleston, SC. The demand, lead time, and cost information of each item is stored in a data set called skuInfo2, as shown in Figure 2.3.

Input Data Set

Obs	sku	warehouse	location	holdingCost	fixedCost	LTmean	RTDmean	RTDvar	serviceLevel
1	A	Raleigh, NC		0.35	90	1	125.1	2170.8	.
2	A	Raleigh, NC	Atlanta, GA	0.70	.	2	32.6	460.2	0.95
3	A	Raleigh, NC	Baltimore, MD	0.70	.	2	61.8	1133.5	0.95
4	A	Raleigh, NC	Charleston, SC	0.70	.	1	30.7	577.1	0.95
5	B	Greensboro, NC		0.05	50	2	140.3	1667.7	.
6	B	Greensboro, NC	Atlanta, GA	0.10	.	2	68.4	907.3	0.95
7	B	Greensboro, NC	Charleston, SC	0.10	.	1	71.9	760.4	0.95

Figure 2.3. Input Data Set skuInfo2

The location and serviceLevel variables have missing values when the observation corresponds to a warehouse. PROC IRP treats the current observation as a warehouse if the corresponding entry for the location variable is missing. Similarly, the fixedCost variable has missing values for the retail locations since the retail locations follow base-stock policies and do not incur ordering costs. Only the warehouses incur ordering costs because they replenish from an outside supplier.

The following IRP procedure call can be used to calculate inventory policies for the warehouses and the retail locations.

```
proc irp data=skuInfo2 out=policy2;
   itemid sku warehouse;
   location location;
   holdingcost holdingCost;
   leadtime / mean=LTmean;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel;
run;
```

The output data set policy2 is displayed in Figure 2.4. The reorderLevel variable gives the reorder level, s, and the orderUpToLevel variable gives the order-up-to level S. Note that for the retailers, the order-up-to level is one greater than the reorder level, since the retailers follow base-stock policies. The _status_ variable indicates that the optimization was successful for all observations. The _algorithm_ variable gives information about the algorithm used; namely, a fill rate ('FR') service level was used for the retailers, the warehouses follow (s, S) ('SS') policies and the retailers follow base-stock ('BS') policies, and the gamma distribution ('GA') was used for lead-time demand and (lead time + review time)-demand for all locations. The remaining variables report several estimated inventory metrics to evaluate the performance of the underlying policy (see the "OUT= Data Set" section on page 32 for information about these variables).

Policy Data Set

Obs	skuse	warehouse	location	Order Level	Order Level	avgInvBack	avgBackOrder	avgCost
1	A	Raleigh, NC		124	376	67.550	26.8844	0.43646	224.864
2	A	Raleigh, NC	Atlanta, GA	168	169	66.030	1.9299	0.99984	46.221
3	A	Raleigh, NC	Baltimore, MD	293	294	98.618	3.4774	1.00000	69.033
4	A	Raleigh, NC	Charleston, SC	132	133	66.694	1.7799	0.99820	46.686
5	B	Greensboro, NC		238	745	151.103	25.7180	0.29107	36.877
6	B	Greensboro, NC	Atlanta, GA	296	297	83.004	3.7423	1.00000	8.300
7	B	Greensboro, NC	Charleston, SC	217	218	64.683	3.6626	1.00000	6.468

Obs	invento ryRat io	backord erRati o	turnov er	filled Ratio	readyRatio	_alg or ithm_	_status_
1	0.53997	0.21490	1.85197	0.79626	0.64136	__-SS-GA	SUCCESSFUL
2	2.02547	0.05920	0.49371	0.94991	0.92813	FR-BS-GA	SUCCESSFUL
3	1.59577	0.05627	0.62666	0.95002	0.91638	FR-BS-GA	SUCCESSFUL
4	2.17244	0.05798	0.46031	0.95101	0.93722	FR-BS-GA	SUCCESSFUL
5	1.07700	0.18331	0.92851	0.83310	0.72480	__-SS-GA	SUCCESSFUL
6	1.21351	0.05471	0.82406	0.94927	0.89959	FR-BS-GA	SUCCESSFUL
7	0.89962	0.05094	1.11158	0.95155	0.88633	FR-BS-GA	SUCCESSFUL

Figure 2.4. Output Data Set policy2

Syntax

The following statements are used in PROC IRP:

PROC IRP *options* ;
HOLDINGCOST *variable* ;
ITEMID *variables* ;
LEADTIME */ lead time options* ;
LEADTIMEDEMAND */ lead-time demand options* ;
LOCATION *variable / location options* ;
PENALTY */ penalty options* ;
POLICYTYPE *variable* ;
REPLENISHMENT */ replenishment options* ;
REVIEWTIMEDEMAND */ review-time demand options* ;
SERVICE */ service options* ;

Functional Summary

The following tables outline the options available for the IRP procedure classified by function.

Table 2.2. Constraints and Policy Specifications

Description	Statement	Option
maximum ordering frequency *variable*	REPLENISHMENT	MAXFREQ=
minimum order size *variable*	REPLENISHMENT	MINSIZE=
base lot size *variable*	REPLENISHMENT	LOTSIZE=
policy type *variable*	POLICYTYPE	
service type *variable*	SERVICE	TYPE=
service level *variable*	SERVICE	LEVEL=

Table 2.3. Cost Specifications

Description	Statement	Option
fixed cost *variable*	REPLENISHMENT	FCOST=
holding cost *variable*	HOLDINGCOST	
penalty cost *variable*	PENALTY	COST=

Table 2.4. Data Set Specifications

Description	Statement	Option
input data set	PROC IRP	DATA=
output data set	PROC IRP	OUT=

Table 2.5. Identifier Variables

Description	Statement	Option
item id *variables*	ITEMID	
location *variable*	LOCATION	

Table 2.6. Lead Time Specifications

Description	Statement	Option
lead time mean *variable*	LEADTIME	MEAN=
lead time variance *variable*	LEADTIME	VARIANCE=
maximum allowed value of coefficient of variation for lead time	LEADTIME	MAXCOV=

Table 2.7. Lead-Time Demand Specifications

Description	Statement	Option
lead-time demand mean *variable*	LEADTIMEDEMAND	MEAN=
lead-time demand variance *variable*	LEADTIMEDEMAND	VARIANCE=
maximum allowed value of coefficient of variation for lead-time demand	LEADTIMEDEMAND	MAXCOV=

Table 2.8. Miscellaneous Options

Description	Statement	Option
maximum number of items for which input error messages are printed	PROC IRP	MAXMESSAGES=
estimate of the maximum number of retail locations	LOCATION	NLOCATIONS=

Table 2.9. Optimization Control Specifications

Description	Statement	Option
type of optimization algorithm	PROC IRP	ALGORITHM=
choice of probability distribution	PROC IRP	DIST=
maximum number of iterations	PROC IRP	MAXITER=
type of policy optimization	PROC IRP	METHOD=
specifies calculation of optimal policies	PENALTY	OPTIMAL
controls the scaling of demand and cost parameters	PENALTY	SCALE=
specifies the granularity of the inventory position distribution for (s, S) policies	REPLENISHMENT	QGRID=
criterion to determine $S - s$ or Q	REPLENISHMENT	DELTA=

Table 2.10. Review-Time Demand Specifications

Description	Statement	Option
review-time demand mean *variable*	REVIEWTIMEDEMAND	MEAN=
review-time demand variance *variable*	REVIEWTIMEDEMAND	VARIANCE=
maximum allowed value of coefficient of variation for review-time demand	REVIEWTIMEDEMAND	MAXCOV=

PROC IRP Statement

PROC IRP *options* **;**

The following options can appear in the PROC IRP statement.

ALGORITHM= 1 | 2
ALG= 1 | 2

specifies the type of optimization heuristic used for single-location systems. This option is ignored when the OPTIMAL option is specified. The default value is 1. See the "Policy Algorithm" section on page 39 for more information.

DATA=*SAS-data-set*

names the SAS data set that contains information about the items to be analyzed. Required information includes the mean and variance of review-time demand, mean replenishment order lead time, per unit holding cost, fixed replenishment cost, and the target service level or backorder penalty cost. Optional information may be supplied with other variables for use by the procedure. For single-location systems, every observation corresponds to an individual inventory item to be analyzed. For two-echelon distribution systems, every observation corresponds to an inventory item-location pair, and these pairs must be grouped together by item.

The DATA= input data set must be sorted by the variables specified with the ITEMID statement. See the "Input Data Set" section on page 30 for more information about the variables in this data set. If the DATA= option is omitted, the most recently created SAS data set is used.

DIST= AUTO | GAMMA

specifies the type of probability distribution used for approximating both the lead-time demand and the (lead time + review time)-demand distributions. If the selection is AUTO, normal distribution is used whenever appropriate and gamma distribution is used otherwise. If the selection is GAMMA, gamma distribution is used every time. This option is ignored when the OPTIMAL option is specified. The default value is AUTO. See the "Policy Algorithm" section on page 39 for more information.

MAXITER=*maxiter*

specifies the maximum number of iterations permitted for the heuristic algorithm to calculate inventory replenishment policies. The default value of *maxiter* is 100. This option is ignored when the OPTIMAL option is specified on the PENALTY statement.

MAXMESSAGES=*maxmessages*
MAXMSG=*maxmessages*
specifies the maximum number of different items in the DATA= input data set for which input error messages are printed to the SAS log. The default value of *maxmessages* is 100.

METHOD= SERVICE | PENALTY
specifies the optimization method used for calculating the inventory replenishment policies. If METHOD= is specified as PENALTY, PROC IRP uses backorder penalty costs to determine the replenishment policy. If METHOD= is SERVICE, then service level requirements are used to calculate the replenishment policy. The default value of METHOD= is SERVICE.

OUT=*SAS-data-set*
specifies a name for the output data set that contains inventory replenishment policies, service measures estimates, and other inventory metrics as determined by PROC IRP. This data set also contains all of the variables specified with the ITEMID statement. Every observation in the DATA= input data set has a corresponding observation in this output data set. See the "OUT= Data Set" section on page 32 for information about the variables in this data set. If the OUT= option is omitted, the SAS system creates a data set and names it according to the DATA*n* naming convention.

HOLDINGCOST Statement

HOLDINGCOST *variable* **;**
HCOST *variable* **;**

The HOLDINGCOST statement identifies the variable in the DATA= input data set that specifies the per-period per-unit holding cost of each item. Negative, zero, and missing values are not permitted. If this statement is not specified, PROC IRP looks for a default variable named HOLDINGCOST. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

ITEMID Statement

ITEMID *variables* **;**
ID *variables* **;**
SKUID *variables* **;**

The ITEMID statement specifies the variables in the DATA= input data set that identify individual inventory items. For a single-location system, the ITEMID variables are primarily used to identify unique items in the input data set. However, each observation is processed independently, regardless of whether or not the values of the ITEMID variables are unique. Thus, you can include any variables that may not necessarily pertain to the descriptions of the items in the list. All the variables specified by this statement are included in the output data set. Therefore, in addition to identifying inventory information (like SKU), the ITEMID statement can also be used in a

single-location system to specify variables that will be carried through from the input data set to the output data set. See Example 2.1 on page 47 for an illustration.

For a two-echelon system, the ITEMID statement specifies the variables in the DATA= input data set that are used in grouping the observations in the input data set. Each group identifies a single item that is shipped from a warehouse to one or more retailers; each individual observation within a group corresponds to a single warehouse or retailer. The observations within a group are used together to process the group. As in the single-location case, the variables specified by the ITEMID statement are included in the output data set; however, in this case, the variables are used to process observations in groups rather than independently. Thus, the ITEMID statement cannot be used (as in the single-location system) to simply copy variables from the input data set to the output data set. Rather, a simple DATA step can be performed after a call to PROC IRP to merge variables from the input and output data sets.

If the ITEMID statement is not specified, PROC IRP halts with an error. Furthermore, PROC IRP expects the DATA= input data set to be sorted by the variables specified by the ITEMID statement. The ITEMID statement behaves much like the BY statement; therefore, you can use options such as DESCENDING and NOTSORTED on the ITEMID statement. Refer to SAS System documentation for more information on the BY statement.

LEADTIME Statement

LEADTIME */ lead time options* **;**
LTIME */ lead time options* **;**

The LEADTIME statement identifies the variables in the DATA= input data set that contain the mean and variance of the replenishment order lead time. This information is used to calculate the mean and variance of lead-time demand. The replenishment order lead time should be specified using the same scale as the review periods. This statement is ignored if the LEADTIMEDEMAND statement is specified.

MEAN=*variable*
identifies the variable in the DATA= input data set that contains the mean of the replenishment order lead time. Negative, zero, and missing values are not permitted. If this option is omitted, PROC IRP looks for a default variable named LTMEAN. If this variable is not found in the DATA= data set, PROC IRP halts with an error.

VARIANCE=*variable*
VAR=*variable*
identifies the variable in the DATA= input data set that contains the variance of the replenishment order lead time. Negative and missing values are interpreted as 0. If this option is omitted, a value of 0 will be used for all observations.

MAXCOV=*maxcov*
specifies the maximum allowed value of the coefficient of variation for replenishment order lead time. Items with coefficient of variation (ratio of the standard deviation and mean) of lead time greater than *maxcov* are not processed. The default value of *maxcov* is 10.

LEADTIMEDEMAND Statement

LEADTIMEDEMAND */ lead-time demand options* **;**
LTDEMAND */ lead-time demand options* **;**

The LEADTIMEDEMAND statement identifies the variables in the DATA= input data set that contain the mean and variance of lead-time demand (that is, the amount of demand that occurs during the lead time). The IRP procedure uses the review-time demand and lead time information to calculate the parameters of lead-time demand. Instead of specifying the parameters of lead time, you can directly specify the mean and variance of lead-time demand with the LEADTIMEDEMAND statement. This feature is especially useful if lead time is greater than review time and demand is not stationary.

If this statement is specified, both the MEAN= and VARIANCE= options must be specified, and the LEADTIME statement is ignored. Since the inventory is periodically reviewed, the lead time in consideration should start after one review period. See Example 2.4 on page 56 for an illustration.

MEAN=*variable*
identifies the variable in the DATA= input data set that contains the mean of the demand during lead time. Negative, zero, and missing values are not permitted.

VARIANCE=*variable*
VAR=*variable*
identifies the variable in the DATA= input data set that contains the variance of the demand during lead time. Negative, zero, and missing values are not permitted.

MAXCOV=*maxcov*
specifies the maximum allowed value of the coefficient of variation for lead-time demand. Items with coefficient of variation (ratio of the standard deviation and mean) of lead-time demand greater than *maxcov* are not processed. The default value of *maxcov* is 10.

LOCATION Statement

LOCATION *variable / location options* **;**
LOC *variable / location options* **;**

The LOCATION statement identifies the character variable in the DATA= data set that identifies the retail locations for the two-echelon distribution inventory problem. The value of the LOCATION variable should be missing if the current observation corresponds to a warehouse. This statement is required to solve two-echelon distribution inventory problems. If this statement is omitted, each observation is treated as a separate single-location inventory problem.

NLOCATIONS=*nlocations*
NLOCS=*nlocations*
specifies an estimate of the maximum number of retail locations in a single item group for the two-echelon distribution inventory problem. This option is used for initial memory allocation. The default value is 50.

PENALTY Statement

PENALTY */ penalty options* **;**

The PENALTY statement enables you to specify backorder penalty cost information. This statement is ignored if the METHOD= option is specified as SERVICE.

COST=*variable*
identifies the variable in the DATA= input data set that specifies the per period per unit item penalty cost for backlogged demand. Negative, zero, and missing values are not permitted. The value of this variable must also be greater than or equal to 1.5 times the value of the HOLDINGCOST variable. This limitation is to avoid accidental user input errors and to guarantee a minimum ready rate of at least 60%. If the METHOD= option is specified as PENALTY and this option is not specified, PROC IRP looks for a default variable named PENALTYCOST. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

OPTIMAL
OPT
specifies that an optimal policy are calculated. This option is valid only if the LOCATION statement is not specified. By default, PROC IRP uses a heuristic method to calculate nearly optimal policies. See the "OPTIMAL Option" section on page 42 for more information.

SCALE=*scale*
controls the initial scaling of demand and cost parameters for optimal policy calculations. Initial scaling takes place if the calculated mean of (lead time + review time)-demand is greater than *scale*. This option is ignored if the OPTIMAL option is not specified. Valid values are between 50 and 10,000, and the default value is 100. In general, the default scaling is sufficient to produce fast and accurate results. If desired, more accuracy may be obtained at the expense of longer execution time by increasing *scale* (thus decreasing the effective scaling). However, increasing *scale* increases the demand on memory and may result in an error. See the "OPTIMAL Option" section on page 42 for more information.

POLICYTYPE Statement

POLICYTYPE *variable* **;**
PTYPE *variable* **;**

The POLICYTYPE statement identifies the variable in the DATA= input data set that specifies the type of inventory replenishment policy to be calculated. The values allowed for the variable specified in the POLICYTYPE statement are listed in Table 2.11. See the "Replenishment Policies" section on page 36 for more information on policy types.

Table 2.11. Valid Values for the POLICYTYPE Variable

Value	Policy Type
BS	base-stock policy
SS	(s, S) policy (default)
NQ	(s, nQ) policy, fixed ordering cost for each lot ordered
RQ	(s, nQ) policy, single fixed ordering cost independent of the number of lots ordered

If this statement is not specified, the default value, SS, is used.

REPLENISHMENT Statement

REPLENISHMENT */ replenishment options* **;**
ORDER */ replenishment options* **;**
REP */ replenishment options* **;**

DELTA= POWER | EOQ
specifies the method used for calculating the difference, $\Delta = S - s$, for (s, S) policies or the base lot size, Q, for (s, nQ) policies. Valid values of DELTA= are POWER and EOQ. The default value is POWER. See the "Policy Algorithm" section on page 39 for more information.

FCOST=*variable*
identifies the variable in the DATA= input data set that specifies the fixed ordering cost of placing a replenishment order. Negative and missing values are interpreted as 0. If this option is not specified, PROC IRP looks for a default variable named FIXEDCOST. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

LOTSIZE=*variable*
identifies the variable in the DATA= input data set that specifies the difference, $\Delta = S - s$, for (s, S) policies or the base lot size, Q, for (s, nQ) policies. Negative, zero, and missing values are ignored.

MINSIZE=*variable*
identifies the variable in the DATA= input data set that contains the minimum allowable replenishment order size. Negative and missing values are ignored, with the exception of -1. A value of -1 is a special flag and sets the minimum order size to 1.5 times the average one-period demand. If this statement is omitted, a value of 0 is used for all observations.

MAXFREQ=*variable*
identifies the variable in the DATA= input data set that contains the maximum allowable average ordering frequency. In practice, the fixed cost of placing an order can be difficult to estimate; therefore, this variable enables the user to put a limit on the frequency with which orders are placed. Negative, zero, and missing values are ig-

nored. Furthermore, for (s, S) policies the value of the maximum ordering frequency cannot be less then 1/*qgrid* where *qgrid* is the value specified by the QGRID= option.

QGRID=*qgrid*
identifies the granularity of the inventory position distribution for (s, S) policies. This option is only used for the heuristic algorithms and ignored when the OPTIMAL option is specified. Valid values are between 5 and 100, the default value is 10. The default value is appropriate for most situations, however increasing it may result in better accuracy (at the expense of computation time).

REVIEWTIMEDEMAND Statement

REVIEWTIMEDEMAND */ review-time demand options* **;**
RTDEMAND */ review-time demand options* **;**

The REVIEWTIMEDEMAND statement identifies the variables in the DATA= input data set that contain the mean and variance of the review-time demand. When using the REVIEWTIMEDEMAND statement, demand over the review periods is assumed to be stationary and independent.

MEAN=*variable*
identifies the variable in the DATA= input data set that contains the mean of the demand during a single inventory review period. Missing values and values less than 1 are not permitted. However, the mean of review-time demand at the warehouse (in the two-echelon distribution problem) can be set to missing to instruct PROC IRP to automatically calculate the mean and variance of demand at the warehouse as the sum of the means and variances of demand at the retail locations. If this option is omitted, PROC IRP looks for a default variable named RTDMEAN. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

VARIANCE=*variable*
VAR=*variable*
identifies the variable in the DATA= input data set that contains the variance of the demand during a single inventory review period. Negative and missing values are interpreted as 0. If this statement is omitted, PROC IRP looks for a default variable named RTDVAR. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

MAXCOV=*maxcov*
specifies the maximum allowed value of the coefficient of variation for review-time demand. Items with coefficient of variation (ratio of the standard deviation and mean) of review-time demand greater than *maxcov* are not processed. The default value of *maxcov* is 10.

SERVICE Statement

SERVICE */ service options* **;**

The SERVICE statement identifies the variables in the DATA= input data set that specify the type and the desired level of the service measure to be used by the inventory policy algorithm. This statement is ignored if the METHOD= option on the PROC IRP statement is specified as PENALTY.

LEVEL=*variable*
identifies the variable in the DATA= input data set that specifies the desired service level for the service measure specified with the TYPE= option. Common ranges of service level are $[0.80, 0.99]$ for fill rate and ready rate, and $[0.01, 0.20]$ for backorder ratio. Valid values for fill rate and ready rate are between 0.600 and 0.999 and for backorder ratio between 0.001 and 0.400. If the METHOD= option is specified as SERVICE and this option is not specified, PROC IRP looks for a default variable named SERVICELEVEL. If this variable is not found in the DATA= input data set, PROC IRP halts with an error.

TYPE=*variable*
identifies the variable in the DATA= input data set that specifies the type of service measure to be used by the inventory replenishment algorithm. Only one service measure per item can be specified in a single procedure invocation. The values allowed for the variable specified in the TYPE= option are listed in Table 2.12.

Table 2.12. Valid Values for the TYPE Variable

Value	Service Measure
FR	Fill rate (default)
RR	Ready rate
BR	Backorder ratio

If this option is not specified, the default value, FR, is used.

Details

This section provides detailed information about the use of the IRP procedure. The material is organized in subsections that describe different aspects of the procedure.

Input Data Set

PROC IRP uses data from the DATA= input data set with key variable names being used to identify the appropriate information. Table 2.13 lists all of the variables associated with the input data set and their interpretation by the IRP procedure. The variables are grouped according to the statement with which they are specified.

Table 2.13. PROC IRP Input Data Set and Associated Variables

Statement	Variable Name	Interpretation
HOLDINGCOST	HOLDINGCOST	holding cost
ITEMID	ITEMID	item identifier
LOCATION	LOCATION	retail location identifier
LEADTIME	MEAN VARIANCE	lead time mean lead time variance
LEADTIMEDEMAND	MEAN VARIANCE	lead-time demand mean lead-time demand variance
PENALTY	COST	backorder penalty cost
POLICYTYPE	POLICYTYPE	policy type
REPLENISHMENT	FCOST LOTSIZE MAXFREQ MINSIZE	fixed ordering cost base lot size maximum ordering frequency minimum order size
REVIEWTIMEDEMAND	MEAN VARIANCE	review-time demand mean review-time demand variance
SERVICE	LEVEL TYPE	desired service level service measure type

Some variables have default names and do not need to be specified in any of the procedure statements. These variables are listed in Table 2.14.

Table 2.14. PROC IRP Input Data Set Default Variable Names

Statement	Variable Name	Default Variable Name
HOLDINGCOST	HOLDINGCOST	HOLDINGCOST
LEADTIME	MEAN	LTMEAN
PENALTY	COST	PENALTYCOST
REPLENISHMENT	FCOST	FIXEDCOST
REVIEWTIMEDEMAND	MEAN VARIANCE	RTDMEAN RTDVAR
SERVICE	LEVEL	SERVICELEVEL

Missing Values in the Input Data Set

Table 2.15 summarizes the treatment of missing values for variables in the DATA= input data set.

Table 2.15. Treatment of Missing Values in the IRP Procedure

Statement	Variable Name	Value / Action Taken
HOLDINGCOST	HOLDINGCOST	input error: procedure moves to processing of next ITEMID group
LOCATION	LOCATION	current observation defines a warehouse
LEADTIME	MEAN	input error: procedure moves to processing of next ITEMID group
	VARIANCE	0
LEADTIMEDEMAND	MEAN	input error: procedure moves to processing of next ITEMID group
	VARIANCE	input error: procedure moves to processing of next ITEMID group
PENALTY	COST	input error: procedure moves to processing of next ITEMID group if METHOD= PENALTY, otherwise ignored
POLICYTYPE	POLICYTYPE	'SS'
REPLENISHMENT	FCOST	0
	LOTSIZE	value ignored
	MAXFREQ	value ignored
	MINSIZE	0
REVIEWTIMEDEMAND	MEAN	input error (unless the value of the LOCATION variable is also missing): procedure moves to processing of next ITEMID group
	VARIANCE	0 (or the sum of other ITEMID group values if the value of the LOCATION variable is missing)
SERVICE	LEVEL	input error: procedure moves to processing of next ITEMID group if METHOD= SERVICE, otherwise ignored
	TYPE	'FR'

OUT= Data Set

The OUT= data set contains the inventory replenishment policies for the items identified in the DATA= input data set. There is one observation for each observation in the DATA= input data set. If an error is encountered while processing an observation, information about the error is written to the OUT= data set.

Definitions of Variables in the OUT= Data Set

Each observation in the OUT= data set is associated with an individual inventory item (or SKU). The variables specified with the ITEMID statement are copied to the

OUT= data set. The following variables are also added to the OUT= data set:

AVGBACKORDER
contains the estimated average backorders for the calculated inventory replenishment policy. Average backorders is the average amount of cumulative backorders in a review period.

AVGCOST
contains the estimated average cost per period for the calculated inventory replenishment policy. Average cost is the average cost (including holding, ordering and backorder penalty costs) incurred per review period.

AVGINVENTORY
contains the estimated average inventory for the calculated inventory replenishment policy. Average inventory is the average on-hand inventory at the end of a review period.

AVGORDERFREQ
contains the estimated average ordering frequency for the calculated inventory replenishment policy. Average ordering frequency is the average number of replenishment orders placed per review period.

BACKORDERRATIO
contains the estimated backorder ratio for the calculated inventory replenishment policy. Backorder ratio is equal to average backorders divided by average demand.

FILLRATE
contains the estimated fill rate for the calculated inventory replenishment policy. Fill rate is the fraction of demand that is satisfied from on-hand inventory. If the OPTIMAL option is specified on the PENALTY statement, the FILLRATE variable is not added to the OUT= data set.

INVENTORYRATIO
contains the estimated inventory ratio for the calculated inventory replenishment policy. Inventory ratio is equal to the average inventory divided by average demand.

ORDERUPTOLEVEL
specifies the order-up-to level, S, for (s, S) policies or the sum of the reorder level and the base lot size, $s + Q$, for (s, nQ) policies.

READYRATE
contains the estimated ready rate for the calculated inventory replenishment policy. Ready rate is the probability of no stockout in a review time period.

REORDERLEVEL
specifies the reorder level, s. The reorder level is the inventory level at which a replenishment order should be placed.

TURNOVER
contains the estimated turnover for the calculated inventory replenishment policy. Turnover is equal to the average demand divided by average inventory. The value of this variable is set to missing if the estimated average inventory is 0.

ALGORITHM
indicates which algorithm was used to calculate the inventory replenishment policy. The value of the _ALGORITHM_ variable is in the form of XX-YY- ZZ, where XX indicates the type of optimization used, YY indicates type of policy calculated, and ZZ indicates the approximation used for both lead-time demand and (lead time + review time)-demand distributions. Possible values for the _ALGORITHM_ variable are specified in Table 2.16.

Table 2.16. Possible Values of the _ALGORITHM_ Variable

String	Value	Description
XX	PC	Penalty cost
	FR	Fill rate
	RR	Ready rate
	BR	Backorder ratio
YY	BS	$(S-1, S)$ base-stock policy
	SS	(s, S) policy (or (s, nQ, S) policy if a base lot size Q is specified)
	NQ	(s, nQ) policy, fixed ordering cost for each lot ordered
	RQ	(s, nQ) policy, single fixed ordering cost independent of the number of lots ordered
ZZ	NO	Normal distribution
	GA	Gamma distribution

The ZZ portion of this variable will have a slightly different format when the OPTIMAL option is specified. See the "OPTIMAL Option" section on page 42 for details.

For two-echelon distribution systems, the XX portion of this variable has value '__' when the current value of the LOCATION variable defines a warehouse (as no service constraints or penalty costs are applied at the warehouse).

SCALE
contains the value used to scale the demand and cost parameters during policy calculations. In the event that scaling is performed (the value of _SCALE_ is greater than 1), it should be noted that all values written to the OUT= data set are in original units. This variable is only added to the OUT= data set when the OPTIMAL option is specified on the PENALTY statement. For more information about scaling, see the "OPTIMAL Option" section on page 42.

STATUS
contains the completion status of the inventory replenishment algorithm. Possible values for the _STATUS_ variable are listed in Table 2.17.

Table 2.17. Possible Values of the _STATUS_ Variable

Value	Explanation
SUCCESSFUL	Successful completion
INVD_VALUE	Invalid value in the DATA= input data set
MAX_ITER	Maximum number of iterations reached
INSUF_MEM	Insufficient memory
BAD_DATA	Numerical or scaling problem encountered

Note that the estimates for average inventory and average backorders lose their accuracy if lead time is not an integer multiple of the review period, or if the variance of lead time is high.

Error Processing

For single-location systems, PROC IRP processes each item (observation) individually. If an error occurs, PROC IRP stops processing the current item and writes information about the type of error to the _STATUS_ variable in the OUT= data set. Execution resumes with the next item.

For two-echelon distribution systems, PROC IRP processes items in groups (multiple observations) representing the warehouse and retail locations. If an error is detected for any of the corresponding observations, PROC IRP stops processing the current item group and the type of error is noted in the _STATUS_ variable for all items in the group. Execution resumes with the next item group.

At procedure termination, the value of the macro variable, _IRPIRP_, will be set appropriately to reflect the fact that errors were encountered during execution.

Macro Variable _IRPIRP_

PROC IRP defines a macro variable named _IRPIRP_ . This variable contains a character string that indicates the status of the procedure. It is set at procedure termination. The form of the _IRPIRP_ character string is STATUS=*status* NSUCCESS=*nsuccess* NFAIL=*nfail*, where *nsuccess* is the number of items successfully processed, *nfail* is the number of items for which the policy calculation has failed, and *status* can be one of the following:

- SUCCESSFUL (indicates successful completion of the procedure)
- RUNTIME_ERROR (indicates that policy calculations failed for at least one item or item group in the DATA= input data set)
- SYNTAX_ERROR (indicates failure due to a procedure syntax error)
- MEMORY_ERROR (indicates failure during procedure initialization or data input parsing due to insufficient memory)

This information can be used when PROC IRP is one step in a larger program that needs to determine whether the procedure terminated successfully or not. Because _IRPIRP_ is a standard SAS macro variable, it can be used in the ways that all macro variables can be used.

Replenishment Policies

PROC IRP calculates four types of replenishment policies, based on the specified policy type:

- **SS=**(s, S) *policy*: When the inventory position falls to or below the reorder level, s, an order is placed so as to bring the inventory position to the order-up-to level, S. In other words, if the inventory position is y, and $y \leq s$, then an order of size $S - y$ is placed. The (s, S) policy is sometimes referred to as the *min-max* policy. Note that the size of the replenishment order is always greater than or equal to $S - s$.

- **BS=**(s, S) *policy when S=s+1 (base-stock policy)*: When the inventory position falls to or below the reorder level, s, an order is placed so as to bring the inventory position to the order-up-to level, S. When $S = s + 1$, the (s, S) policy is called a *base-stock policy* . (A base stock policy is also called an "order-up-to policy," "one-to-one replenishment policy," or "installation stock policy").

- **NQ=**(s, nQ) *policy when you have a fixed ordering cost for each lot ordered*: You incur a fixed ordering cost for each lot ordered. When the inventory position falls to or below the reorder level, s, an order is placed to bring the inventory position just above s. The size of this order is a multiple of the base lot size, Q.

  In other words, if the inventory position is y, and $y \leq s$, then an order of size nQ is placed, where n is the smallest integer such that $y + nQ > s$. In this case, both s and Q are decision variables; you can use the LOTSIZE= option if Q is to be a previously specified value rather than a decision variable. When $Q = 1$, the (s, nQ) policy becomes a base-stock policy.

- **RQ=**(s, nQ) *policy when you incur a single fixed ordering cost*: You incur a single fixed ordering cost independent of the number of lots ordered. When the inventory position falls to or below the reorder level, s, an order is placed to bring the inventory position just above s. The size of this order is a multiple of the base lot size, Q.

  In other words, if the inventory position is y, and $y \leq s$, then an order of size nQ is placed, where n is the smallest integer such that $y + nQ > s$. In this case, both s and Q are decision variables; you can use the LOTSIZE= option if Q is to be a previously specified value rather than a decision variable. When $Q = 1$, the (s, nQ) policy becomes a base-stock policy.

For single-location inventory systems under standard assumptions (independent customer demands, full backordering of unfulfilled demand, fixed replenishment ordering costs, linear inventory holding costs, and linear backorder penalty costs), (s, nQ) policies are known to be suboptimal and (s, S) policies are known to be optimal.

Although (s, S) policies are optimal, the restricted order size under an (s, nQ) policy may better facilitate easy packaging, transportation, and coordination in some situations.

Inventory Costs

Since the objective of inventory planning is usually to minimize costs, the assumptions about the cost structure are important. There are three types of costs: ordering, holding, and penalty (backordering) costs.

Ordering cost is the cost incurred every time a replenishment order is placed. This fixed cost includes the expense associated with processing the order and is typically independent of the size of the order.

Holding cost is the cost of carrying inventory and may include the opportunity cost of money invested, the expenses incurred in running a warehouse, handling and counting costs, the costs of special storage requirements, deterioration of stock, damage, theft, obsolescence, insurance and taxes. The most common convention is to specify holding cost (per period per unit item) as a fixed percentage of the unit cost of the item. This cost is then applied to the average inventory.

Penalty (*backordering or shortage*) *cost* is the cost incurred when a stockout occurs. This cost may include the cost of emergency shipments, cost of substitution of a less profitable item, or cost of lost goodwill. For instance, will the customer ever return? Will the customer's colleagues be told of the disservice? The most common convention is to specify penalty cost as per period per unit item and then apply it to average backorders.

In practice, it is often difficult to estimate the ordering (replenishment) cost and the penalty cost. As a result, practitioners often put restrictions on the ordering frequency rather than estimating the cost of ordering. Likewise, specific target levels for service measures can be substituted for the penalty cost.

Service Measures

Service measures are often used to evaluate the effectiveness of an inventory replenishment policy. You can influence policy calculations by imposing desired service level requirements. PROC IRP supports the use of three different service constraints:

- *Fill Rate* — the fraction of demand satisfied directly from on-hand inventory. Fill rate is one of the most frequently used service measures in practice. You can set a minimum fill rate as a service constraint.
- *Ready Rate* — the probability of no stockout in a review period. You can set a minimum ready rate as a service constraint.
- *Backorder Ratio* — average backorders divided by average demand. You can set a maximum backorder ratio as a service constraint.

These service constraints provide different ways of penalizing backorders. When using fill rate as a service measure, the focus is only on the size of backorders, whereas

with backorder ratio as a service measure, the focus is both on the amount and length of backorders. When using ready rate as a service measure, the focus is not on the size and length of backorders, but whether or not a stockout occurs.

Note that setting a high target service level may result in high inventory levels, which can be very costly if demand is intermittent (slow-moving). In these cases, estimating penalty costs and performing a cost optimization may be preferred.

PROC IRP reports several other measures to evaluate the performance of a policy:

- *Average Ordering Frequency* — number of replenishment orders placed per review period. You can set a limit on the average ordering frequency.
- *Average Inventory* — average on-hand inventory at the end of a review period.
- *Average Backorder* — average amount of outstanding backordered demand in a review period.
- *Inventory Ratio* — average inventory divided by average demand.
- *Turnover* — average demand divided by average inventory.
- *Average Cost* — average cost (holding and replenishment) incurred per period. If backorder penalty costs are present, these are included as well.

Lost Sales

A *lost-sales* inventory system enables unsatisfied demand to be lost rather than backordered. For an (s, S) policy, this system can be approximated by using the fill rate service measure with some slight modifications (Tijms and Groenevelt 1984).

Let β_l represent the fraction of satisfied demand in the lost-sales case. Therefore, $1 - \beta_l$ represents the fraction of demand that is lost. The reorder and order-up-to levels for the lost-sales inventory system are approximately the same as those in a backordering inventory system that has a target fill rate service level specified as $\beta_f = 2 - 1/\beta_l$. This approximation should be used only when β_l is close to 1.

Two-Echelon Distribution Inventory System

PROC IRP can find nearly optimal policies for two-echelon distribution systems with different service constraints on multiple retail locations. A two-echelon distribution system consists of a single warehouse and N retail locations. The retail locations pull items from the warehouse and the items are supplied to the warehouse by an exogenous supplier. Figure 2.5 shows a two-echelon distribution system, where node 0 designates the warehouse and nodes 1 through N designate the retail locations.

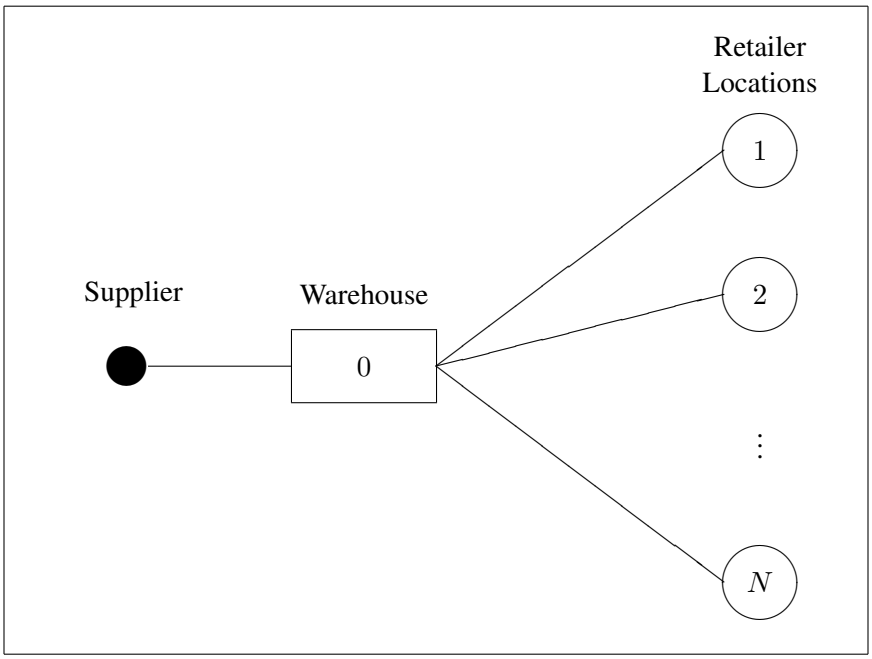

Figure 2.5. Two-Echelon Distribution Inventory System

Retail locations place replenishment orders at the warehouse according to a base-stock policy. The replenishment cost of retail locations is negligible or constant. There is a lead time L_i from the warehouse to retailer i. In addition, there is a lead time L_0 from the outside supplier to the warehouse. If the warehouse has sufficient inventory on hand, it immediately dispatches the order, so that the order will arrive at the retail location after the appropriate lead time. If the warehouse has some inventory on hand, but not enough to fill the entire order, it partially fills the order with the on-hand inventory and backorders the rest. If the warehouse has no inventory on hand when a retail location places an order, the entire order is backordered at the warehouse. Note that the average lead time realized at retail location i is greater than L_i since the retail locations have to wait longer if the warehouse is out of stock. All orders received at the warehouse have the same priority. The warehouse follows an (s, S) or (s, nQ) policy and incurs a fixed replenishment cost every time it places an order from the outside supplier. The retail locations can have different demand patterns and service constraints. If the penalty costs on backorders at the retail locations are known, the total system cost incurred per period is minimized.

Policy Algorithm

Single-Location Systems

When the IRP procedure is used to calculate replenishment policies for single-location inventory systems, the underlying assumptions of the optimization model are as follows:

- The holding and stockout costs are linear.
- The probability that replenishment orders cross in time or arrive simultaneously is negligible.

- The stock on hand just after arrival of a replenishment order is positive except for a negligible probability.
- Review period demand is independent and identically distributed (stationary). If demand is nonstationary, PROC IRP can still find nearly optimal myopic policies using the LEADTIMEDEMAND statement. See Example 2.4 for an illustration.

Let

$$
\begin{aligned}
D_R &= \text{demand during review time} \\
D_L &= \text{demand during lead time} \\
L &= \text{lead time (in number of review periods)} \\
D_{LR} &= \text{demand during lead time plus review time} \\
OF &= \text{ordering frequency per period} \\
I &= \text{on-hand inventory at the end of a period} \\
B &= \text{outstanding backorders in a period} \\
K &= \text{fixed cost of replenishment} \\
h &= \text{holding cost per period} \\
p &= \text{penalty cost per period}
\end{aligned}
$$

PROC IRP supports two different methods for solving single-location inventory problems. When the METHOD= option is specified as SERVICE, PROC IRP uses a service level requirement to constrain the optimization. Alternatively, when the METHOD= option is specified as PENALTY, PROC IRP uses backorder penalty costs to drive the optimization.

By default, PROC IRP uses a heuristic algorithm to calculate nearly optimal policies. If the penalty cost method is used, the OPTIMAL option can be specified on the PENALTY statement to indicate that PROC IRP should calculate optimal policies.

The type of policy calculated by the IRP procedure is determined by the value of the POLICYTYPE variable. See the "POLICYTYPE Statement" section on page 27 for more information.

Service Constraint Method

If the METHOD= option is specified as SERVICE, PROC IRP finds nearly optimal policies where the replenishment and holding costs are minimized subject to a service level constraint. The policy calculation is done in three steps.

Step 1: The mean and variance of D_L and D_{LR} are calculated (unless they are specified with the LEADTIMEDEMAND statement).

Step 2: The algorithm finds $\Delta = S - s$ (the gap between S and s for (s, S) policies) if the value of the POLICYTYPE variable is 'SS,' or $\Delta = Q$ (the base lot size for (s, nQ) policies) if the value of the POLICYTYPE variable is 'RQ' or 'NQ.'

If the fixed replenishment cost, K, is known (specified by the FCOST= variable), Δ is determined according to the specification of the DELTA= option. If the DELTA= option is specified as EOQ, Δ is set to the *classic economic order quantity* (EOQ). If the DELTA= option is specified as POWER, a power approximation (similar to the one in Ehrhardt and Mosier 1984) is used to determine Δ.

If the fixed replenishment cost, K, is not known and/or there is a constraint on Δ (specified by the MINSIZE= variable) or a constraint on the ordering frequency (specified by the MAXFREQ= variable), Δ is adjusted so that these constraints are met. If a base lot size, Q, is specified (by the LOTSIZE= variable), Δ is set to Q and all other constraints are ignored.

Step 3: The reorder level, s, is found such that the user-specified service type and desired service level are met.

The optimization algorithm applied in Step 3 depends on the ALGORITHM= option. If it is specified as 1, the calculations are exact. If it is specified as 2, an approximation algorithm is used.

The approximation is fast and works well for large Δ ($\Delta \geq 1.5 \times E(D_R)$) and leads to nearly optimal solutions. If Δ is small, policy parameters are modified. For (s, S) policies, the reorder level, s, and order-up-to level, S, are determined as

$$s = \begin{cases} S_b, & S_b < s_p \\ s_p, & s_p \leq S_b \leq S_p \\ S_b - \Delta, & S_b > S_p \end{cases}$$

$$S = S_b$$

where (s_p, S_p) is the policy found assuming Δ is large and S_b is the base stock level for the same problem. If there are constraints on the order size or the ordering frequency, these are taken into account as well. For (s, nQ) policies, the reorder level, s, and base lot size, Q, are determined as

$$s = S_b - \Delta/2$$

$$Q = \Delta$$

where S_b is the base stock level for the same problem.

A suitable distribution must be chosen to represent both lead-time demand and (lead time + review time)-demand distributions. In practice, the normal distribution is widely used to approximate these distributions. However, this choice can lead to very poor results if the coefficients of variation are not small. To overcome this drawback of the normal distribution, PROC IRP uses the gamma distribution if the coefficient of variation of (lead time + review time)- demand is greater than 0.5. In both cases, a two-moment approximation is used.

Penalty Cost Method

If the METHOD= option is specified as PENALTY, PROC IRP finds nearly optimal policies where the replenishment, holding, and backorder penalty costs are minimized. The policy calculation is the same as outlined in the "Service Constraint Method" section on page 40, except for the final step.

In the final step, the reorder level, s, is found such that the average cost per period

$$C(s, \Delta) = K\,E(OF) + h\,E(I) + p\,E(B)$$

is minimized. The choice of distribution used to represent both lead-time demand and (lead time + review time)-demand is the same as described in the "Service Constraint Method" section on page 40.

Note that this heuristic finds Δ and s sequentially. You can specify the OPTIMAL option to find true optimal policies for single-location systems where Δ and s are jointly optimized. See the "OPTIMAL Option" section on page 42 for detailed information.

Base-Stock Policies

If the value of the POLICYTYPE variable is 'BS,' or if there is no cost of ordering $(K = 0)$ and no constraints on the order size or the ordering frequency, a base-stock policy is calculated. The policy calculation is similar to what is outlined in the "Service Constraint Method" section on page 40. The difference is that step 2 is skipped since $\Delta = 0$ for base-stock policies.

OPTIMAL Option

If the OPTIMAL option on the PENALTY statement is specified, PROC IRP finds optimal (s, S) or (s, nQ) policies for single-location systems (see Zheng and Federgruen (1992) and Zheng and Chen (1992) for details). The decision variables are s and S for (s, S) policies and s and Q for (s, nQ) policies. In this case, the variables specified by the LOTSIZE=, MAXFREQ=, and MINSIZE= options on the REPLENISHMENT statement are ignored. The algorithm used when the OPTIMAL option is specified is slower than the heuristic algorithm. Note that the OPTIMAL option is not available for two-echelon distribution systems.

Define the following notation:

$C_{SS}(s, S)$ = average cost of an (s, S) policy
(when the value of the POLICYTYPE variable is 'SS')

$C_{NQ}(s, Q)$ = average cost of an (s, nQ) policy when the fixed cost K is incurred for each lot Q ordered
(when the value of the POLICYTYPE variable is 'NQ')

$C_{RQ}(s, Q)$ = average cost of an (s, nQ) policy when the fixed cost K is incurred independent of the number of lots ordered
(when the value of the POLICYTYPE variable is 'RQ')

In each instance, PROC IRP finds optimal values of the decision variables s^*, S^*, and Q^* such that the average cost per period is minimized:

$$\begin{aligned} C_{SS}(s^*, S^*) &= \min_{\forall s,S} C_{SS}(s, S) \\ C_{NQ}(s^*, Q^*) &= \min_{\forall s,Q} C_{NQ}(s, Q) \\ C_{RQ}(s^*, Q^*) &= \min_{\forall s,Q} C_{RQ}(s, Q) \end{aligned}$$

Each optimal policy is optimal within its own class. Note that (s, S) policies are optimal among *all* classes of policies for single-location systems:

$$C_{SS}(s^*, S^*) \leq C_{RQ}(s^*, Q^*) \leq C_{NQ}(s^*, Q^*)$$

Suitable distributions must be chosen to represent lead-time demand and review- time demand. These distributions are assumed to be discrete. If the variance is greater than the mean, the distribution under consideration is approximated by a negative binomial distribution. If the variance is less than or equal to the mean, a shifted Poisson distribution is used. The negative binomial and shifted Poisson distributions are fit such that the resulting mean and variance match the mean and variance of the original distribution. The chosen distributions are indicated by a 'B' or 'P' in the 'ZZ' part of the _ALGORITHM_ variable, where the first Z indicates the approximation used for lead-time demand plus review-time demand distribution, and the second Z indicates the approximation used for the review-time demand distribution. While choosing an appropriate distribution, the algorithm may choose a deterministic distribution (a fixed number) to represent these distributions if the variance is close to zero or considerably smaller than the mean. In that case, this number matches the mean of the estimated distribution and this choice is indicated by a 'D' (for deterministic) in the _ALGORITHM_ variable. If the chosen policy is NQ, the review-time demand distribution does not play a role in the optimization algorithm. This is indicated by a '_' in the _ALGORITHM_ variable.

The OUT= data set contains a new variable named _SCALE_, which gives the value used to scale the demand and cost parameters. Initial scaling takes place if the calculated mean of (lead time + review time)-demand is greater than the value specified by the SCALE= option. Further scaling may be performed by the procedure to obtain a suitable fit to the shifted Poisson or negative binomial distribution. If the procedure is unable to find a suitable fit, it stops processing the current item and writes the value "BAD_DATA" to the _STATUS_ variable. Increasing the value of the SCALE= option may resolve this issue.

The magnitude of demand and cost parameters affect the algorithm's memory requirement to calculate policies. In some cases, if insufficient scaling is performed, PROC IRP may run out of memory. In this case PROC IRP stops processing the current item and writes the value "INSUF_MEM" to the _STATUS_ variable. Usually, decreasing the value of the SCALE= option will correct this problem. Note that a smaller value for the SCALE= option will result in scaling by an equal or larger value.

Two-Echelon Distribution Inventory Systems

When the IRP procedure is used to calculate replenishment policies for two-echelon distribution systems, the underlying assumptions of the optimization model are the same as in single-location systems. Let

$$
\begin{aligned}
OF_0 &= \text{ordering frequency per period at the warehouse} \\
S_0 &= \text{order-up-to level at the warehouse} \\
s_i &= \text{reorder level at location } i = 0, \ldots, N \\
\mu_i &= \text{review-time demand at location } i = 0, \ldots, N \\
K_i &= \text{fixed cost of replenishment at location } i = 0, \ldots, N \\
h_i &= \text{holding cost per period at location } i = 0, \ldots, N \\
p_i &= \text{penalty cost per period at location } i = 1, \ldots, N \\
I_i &= \text{on-hand inventory at end of period at location } i = 0, \ldots, N \\
B_i &= \text{outstanding backorders in a period at location } i = 0, \ldots, N
\end{aligned}
$$

Location $i = 0$ refers to the warehouse.

PROC IRP supports two different methods for solving two-echelon distribution problems. When the METHOD= option is specified as SERVICE, PROC IRP uses a service level requirement to constrain the optimization. Alternatively, when the METHOD= option is specified as PENALTY, PROC IRP uses backorder penalty costs to drive the optimization.

For two-echelon distribution systems, PROC IRP calculates base-stock policies for each retail location. The type of policy for the warehouse is determined by the value of the POLICYTYPE variable.

Service Constraint Method

If the METHOD= option is specified as SERVICE, PROC IRP finds nearly optimal policies where the replenishment and holding costs are minimized subject to service level constraints on the retail locations. The policy calculation is done in three steps.

Step 1 The mean and variance of review-time demand at the warehouse are calculated as the sum of the means and the sum of the variances of review-time demand at the retail locations, respectively. However, if the mean and variance of review-time demand at the warehouse are explicitly specified in the DATA= input data set, those values are used instead.

Note that a collaborative forecast (for all retail locations) may yield a better prediction of the variance of review-time demand at the warehouse than the sum of the variances at the retail locations. Also, note that specifying a value for review-time demand at the warehouse that is significantly different than the sum of the means of review-time demand at the retail locations may cause numerical problems.

Next, the mean and variance of lead-time demand at the warehouse are calculated.

Step 2 This step is the same as for the single-location problem in the "Service Constraint Method" section on page 40, $\Delta_0 = S_0 - s_0$ is calculated for the warehouse. In order to increase the performance of the algorithm for (s, S) policies, if the calculated Δ_0 is less than $0.75\mu_0$ and there are no constraints on the order size, a base- stock policy at the warehouse is assumed.

Step 3 The average cost per period incurred by the system is given as

$$C(s_0, \Delta_0, s_1, s_2, \ldots, s_N) = K_0 E(OF_0) + \sum_{i=0}^{N} h_i E(I_i)$$

In this final step, the cost function is minimized subject to the service level constraint at each retail location. The decision variables are $s_i, i = 0, 1, \ldots, N$. Note that $\Delta_0 = S_0 - s_0$ is calculated and fixed in Step 2.

Each retail location may have a constraint on fill rate, ready rate, or backorder ratio.

As with the single-location problem, a distribution needs to be chosen to represent the lead-time demand and (lead time + review time)-demand at the warehouse and the retail locations. PROC IRP uses the gamma distribution to represent these demand distributions (see the "Service Constraint Method" section on page 40).

Note that PROC IRP ignores any information regarding service levels, service types, and penalty costs at the warehouse, since the backorders at the warehouse are treated implicitly. Similarly, any information regarding policy type, fixed ordering costs, fixed lot size, minimum order size, and maximum ordering frequency are ignored at the retailers since they follow a base-stock policy.

Penalty Cost Method

If the METHOD= option is specified as PENALTY, PROC IRP finds nearly optimal policies where the replenishment, holding, and backorder penalty costs are minimized. The policy calculation is the same as outlined in the "Service Constraint Method" section on page 44 except for the final step.

In the final step, the cost function

$$C(s_0, \Delta_0, s_1, s_2, \ldots, s_N) = K_0 E(OF_0) + \sum_{i=0}^{N} h_i E(I_i) + \sum_{i=1}^{N} p_i E(B_i)$$

is minimized. The decision variables are again $s_i,\ i = 0, 1, \ldots, N$.

Note that this function does not penalize backorders at the warehouse directly (there is no $p_0 E(B_0)$ component). This is justified because customer transactions occur only at the retail locations. Backorders at the warehouse translate to poor performance at the retail locations by increasing the replenishment order lead time.

Retail Location Replenishment Order Lead Time

The replenishment order lead time from the warehouse to any retail location is equal to the shipping and handling time as long as the warehouse has the necessary quantity in stock. In the event of shortages at the warehouse, the retail location has to wait for an extra amount of time, which is dependent upon the time required to replenish the warehouse from a supplier. This extra *warehouse wait* is a function of the warehouse reorder and order-up-to levels. This makes the lead-time demand process at a retail location dependent upon both the retailer base-stock level (s_i) and the warehouse reorder and order-up-to levels (s_0 and S_0). PROC IRP estimates the mean and variance of the warehouse wait using techniques similar to those described in Matta and Sinha (1995).

Examples

This section illustrates how PROC IRP can be used to calculate inventory replenishment policies. Example 2.1 through Example 2.5 focus on a single-location system, while Example 2.6 and Example 2.7 focus on a two-echelon distribution system.

Example 2.1. Single-Location System: Service Level Heuristic

In this example, inventory replenishment policies are calculated for a single-location system using service level heuristics. The retailer purchases finished goods from its suppliers and sells them to its customers. There are 10 items to be considered, identified by the SKU variable. Estimates of the holding and fixed costs, mean and variance of the lead time, and mean and variance of the review-time demand are given in Table 2.18. The missing value for Fixed Cost in the last observation indicates that the fixed cost for S10 is difficult to estimate; a maximum order frequency is placed on this item to account for this.

Table 2.18. Data Estimates for Single-Location System

SKU	Supplier	Holding Cost	Fixed Cost	Lead Time		Review-Time Demand	
				Mean	Variance	Mean	Variance
S01	ABC Company	0.78	70	1	0.6	39	557
S02	JKL Company	0.96	3	2	1.9	35	404
S03	XYZ Company	0.94	52	2	0	26	199
S04	XYZ Company	0.74	17	3	2.2	75	2541
S05	QRS Company	0.48	19	5	0	9	75
S06	QRS Company	0.68	0	5	6.1	92	4132
S07	ABC Company	0.95	60	2	1.5	94	3266
S08	JKL Company	0.39	90	3	0	20	289
S09	ABC Company	0.47	25	1	0	5	6
S10	ABC Company	0.53	.	4	1.6	62	1437

Based on contracts with its suppliers, the retailer must follow an (s, nQ) policy for items S02, S05, and S08. For items S02 and S08, there is a fixed ordering cost for each lot ordered, and for item S05, there is a single fixed ordering cost, independent of the number of lots ordered. In addition, item S06 has a fixed cost of 0, so the retailer follows a base-stock policy. For the remaining items, the retailer follows (s, S) policies.

The retailer faces additional constraints on some items. When placing an order for item S01, the retailer must take into account that the supplier does not fill any orders that are smaller than 15 items. The supplier for item S07 only fills orders in multiples of 10 items. The fixed cost of item S10 is unknown, so the retailer imposes a maximum order frequency of 25% (that is, on average, the retailer will order at most once every four review periods). The retailer also imposes a maximum order frequency of 50% for S04, even though the fixed ordering cost is known for this item.

Using this information, the retailer first wants to calculate inventory policies using a target fill rate of 97%. This means that 97% of all incoming customer orders can be filled from on-hand inventory. The information is stored in the following data set in1_fr:

```
data in1_fr;
   format sku $3. supplier $11. policyType $2. serviceType $2.;
   input  sku $ supplier $ holdingCost fixedCost LTmean LTvar
          RTDmean RTDvar serviceLevel serviceType $
          policyType $ fixedLotSize minOrderSize maxFreq ;
   datalines;
S01 ABC Company  0.78  70  1  0.6  39   557  0.97  FR  SS   .  15     .
S02 JKL Company  0.96   3  2  1.9  35   404  0.97  FR  NQ   .   .     .
S03 XYZ Company  0.94  52  2  0    26   199  0.97  FR  SS   .   .     .
S04 XYZ Company  0.74  17  3  2.2  75  2541  0.97  FR  SS   .   .  0.50
S05 QRS Company  0.48  19  5  0     9    75  0.97  FR  RQ   .   .     .
S06 QRS Company  0.68   0  5  6.1  92  4132  0.97  FR  BS   .   .     .
S07 ABC Company  0.95  60  2  1.5  94  3266  0.97  FR  SS  10   .     .
S08 JKL Company  0.39  90  3  0    20   289  0.97  FR  NQ   .   .     .
S09 ABC Company  0.47  25  1  0     5     6  0.97  FR  SS   .   .     .
S10 ABC Company  0.53   .  4  1.6  62  1437  0.97  FR  SS   .   .  0.25
;
```

The retailer then uses the following call to PROC IRP to compute the inventory policies. Because the METHOD= option is specified as SERVICE, heuristics are used to compute inventory policies based on target service levels. The variables in the input data set are specified using the HOLDINGCOST, ITEMID, LEADTIME, POLICYTYPE, REPLENISHMENT, REVIEWTIMEDEMAND, and SERVICE statements. Note that two variables, sku and supplier, are specified on the ITEMID statement. The specified variables are copied from the input data set to the output data set; this enables the retailer to include additional information about the suppliers in the output data set.

```
proc irp data=in1_fr out=out1_fr method=service;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   policytype policyType;
   replenishment / fcost=fixedCost lotsize=fixedLotSize
                   minsize=minOrderSize maxfreq=maxFreq;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.1.1. This data set contains two variables that define the computed policy: reorderLevel and orderUpToLevel. The remaining variables give more details about the policy, including statistics regarding average inventory, average backorders, and so on, as well as the type of algorithm used to compute the policy.

Note that the first two characters in the _algorithm_ variable are 'FR' for all observations; this is because the algorithm used the fill rate target level in the heuristic. The second set of characters in the _algorithm_ variable gives the type of policy computed. The third set of characters in the _algorithm_ variable indicates which distribution is used to approximate both the lead-time demand and (lead time + review time)- demand. This is either 'GA' for the gamma distribution, or 'NO' for the normal distribution.

Output 2.1.1. Inventory Policies Using 97% Target Fill Rate

PROC IRP Results
Target Measure: 97% Fill Rate

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	119	209	101.694	1.58259	0.38244	106.092
2	S02	JKL Company	203	238	117.655	2.15474	1.00000	115.949
3	S03	XYZ Company	90	147	50.264	0.70077	0.40027	68.062
4	S04	XYZ Company	532	643	315.581	4.26971	0.49855	242.006
5	S05	QRS Company	77	118	43.841	0.34130	0.21771	25.180
6	S06	QRS Company	1121	1122	577.537	7.53671	0.99995	392.725
7	S07	ABC Company	570	580	302.990	5.02517	0.99491	347.535
8	S08	JKL Company	103	228	86.188	0.68808	0.16000	48.013
9	S09	ABC Company	9	32	12.855	0.10577	0.24150	12.079
10	S10	ABC Company	395	662	249.004	2.27842	0.24993	131.972

Obs	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	2.60754	0.040579	0.38350	0.97200	0.95418	FR-SS-GA	SUCCESSFUL
2	3.36156	0.061564	0.29748	0.97005	0.95156	FR-NQ-GA	SUCCESSFUL
3	1.93323	0.026953	0.51727	0.97356	0.94380	FR-SS-NO	SUCCESSFUL
4	4.20775	0.056929	0.23766	0.97033	0.95958	FR-SS-GA	SUCCESSFUL
5	4.87126	0.037922	0.20529	0.96986	0.96356	FR-RQ-NO	SUCCESSFUL
6	6.27757	0.081921	0.15930	0.97062	0.96101	FR-BS-GA	SUCCESSFUL
7	3.22330	0.053459	0.31024	0.97029	0.95450	FR-SS-GA	SUCCESSFUL
8	4.30940	0.034404	0.23205	0.96998	0.95956	FR-NQ-NO	SUCCESSFUL
9	2.57091	0.021154	0.38897	0.97892	0.94960	FR-SS-NO	SUCCESSFUL
10	4.01619	0.036749	0.24899	0.97100	0.95821	FR-SS-NO	SUCCESSFUL

The fill rates for all items are near 97%, the specified target level. However, suppose the retailer thinks the resulting backorder ratios are unacceptably high. Only one service measure per observation can be specified in a single call to PROC IRP, so now the retailer specifies a 3% target backorder ratio for all items, which ignores the 97% target fill rate. The DATA step used to make this change is as follows.

```
data in1_br;
   set in1_fr;
   serviceLevel = 0.03;
   serviceType = 'BR';
run;
```

The retailer then calls PROC IRP as follows. Note that this call is exactly the same as the previous call to PROC IRP, except for a different name for the output data set. Some of the variable values (for the serviceLevel and serviceType variables) have changed, but the variable names have not changed.

```
proc irp data=in1_br out=out1_br method=service;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   policytype policyType;
   replenishment / fcost=fixedCost lotsize=fixedLotSize
                   minsize=minOrderSize  maxfreq=maxFreq;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.1.2. Notice that the average inventory increased for the 3% backorder ratio target level, as compared to the 97% fill rate target level. More inventory is required to meet this more restrictive target service measure.

Output 2.1.2. Inventory Policies Using 3% Target Backorder Ratio

PROC IRP Results
Target Measure: 3% Backorder Ratio

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	130	220	112.259	1.14802	0.38244	114.333
2	S02	JKL Company	234	269	147.562	1.06201	1.00000	144.660
3	S03	XYZ Company	90	147	50.264	0.70077	0.40027	68.062
4	S04	XYZ Company	600	711	381.532	2.21989	0.49855	290.809
5	S05	QRS Company	79	120	45.775	0.27482	0.21771	26.108
6	S06	QRS Company	1311	1312	762.757	2.75746	0.99995	518.675
7	S07	ABC Company	634	644	364.758	2.79347	0.99491	406.215
8	S08	JKL Company	105	230	88.111	0.61110	0.16000	48.763
9	S09	ABC Company	9	32	12.855	0.10577	0.24150	12.079
10	S10	ABC Company	407	674	260.547	1.82169	0.24993	138.090

Obs	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	2.87845	0.029437	0.34741	0.97997	0.96624	BR-SS-GA	SUCCESSFUL
2	4.21606	0.030343	0.23719	0.98637	0.97542	BR-NQ-GA	SUCCESSFUL
3	1.93323	0.026953	0.51727	0.97356	0.94380	BR-SS-NO	SUCCESSFUL
4	5.08709	0.029599	0.19658	0.98481	0.97833	BR-SS-GA	SUCCESSFUL
5	5.08609	0.030536	0.19661	0.97534	0.96980	BR-RQ-NO	SUCCESSFUL
6	8.29084	0.029972	0.12062	0.98986	0.98510	BR-BS-GA	SUCCESSFUL
7	3.88041	0.029718	0.25771	0.98404	0.97405	BR-SS-GA	SUCCESSFUL
8	4.40556	0.030555	0.22699	0.97313	0.96341	BR-NQ-NO	SUCCESSFUL
9	2.57091	0.021154	0.38897	0.97892	0.94960	BR-SS-NO	SUCCESSFUL
10	4.20237	0.029382	0.23796	0.97644	0.96549	BR-SS-NO	SUCCESSFUL

As the average inventory increases, so does the average cost. This is because the retailer has not specified a penalty cost for the backorders in order to balance the holding cost of inventory. The only costs considered in the service level heuristics are fixed ordering costs and inventory holding costs. You can see from Output 2.1.1 and Output 2.1.2 that the average ordering frequency (avgOrderFreq) does not change much between the two target level specifications. Therefore, the bulk of the increase in average cost comes from the increase in average inventory.

The retailer now has two policies to choose from. While one policy does have a higher average cost, the decision should not be based on cost alone. Both policies are heuristic policies, and are derived using different target service levels. With a higher level of service comes a higher cost, and the retailer must decide based on the desired levels of service which policy best fits the needs of the company.

Another option for the retailer is to use backorder penalty cost information to find inventory policies. This problem is explored in Example 2.2 and Example 2.3.

Example 2.2. Single-Location System: Penalty Costs

In this example, assume that the retailer from Example 2.1 is able to obtain estimates of backorder penalty costs. Rather than using a service level heuristic, as in Example 2.1, the retailer uses the penalty costs to calculate inventory policies. First, the retailer uses a heuristic method to calculate nearly optimal inventory policies. In Example 2.3, the optimal inventory policy is calculated.

The backorder penalty costs are given in the following data set.

```
data pcosts;
   format sku $3. penaltyCost;
   input sku $ penaltyCost;
   datalines;
S01   7.4
S02  10.2
S03   8.1
S04   6.6
S05   9.2
S06   9.0
S07   7.1
S08   3.7
S09   5.2
S10  10.8
;
```

This data set is merged with in1_fr to produce the input data set in2. The serviceType and serviceLevel variables are dropped from the in1_fr data set, as these will not be needed when using penalty costs. However, if these variables are left in the data set, they will simply be ignored when the METHOD= option is specified as PENALTY.

```
data in2;
   merge in1_fr (drop=serviceType serviceLevel)
         pcosts;
   by sku;
run;
```

The retailer then calls PROC IRP using the following code. There are several differences between this call and the calls to PROC IRP in Example 2.1. First, the METHOD= option is specified as PENALTY. Second, the PENALTY statement is included, and the penalty cost variable is identified as penaltyCost. Note that there are no other options specified in the PENALTY statement, so the policy is calculated using a heuristic. Finally, the SERVICE statement is no longer listed in the PROC IRP call; if it had been listed, it would be ignored.

```
proc irp data=in2 out=out2 method=penalty;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   penalty / cost=penaltyCost;
   policytype policyType;
   replenishment / fcost=fixedCost lotsize=fixedLotSize
                   minsize=minOrderSize maxfreq=maxFreq;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.2.1. For all items in this example, the average inventory is lower than that shown in Output 2.1.1 and Output 2.1.2, and the average backorders are higher. As a result, the fill rates for this policy are less than 97%, and the backorder ratios are greater than 3%.

The average cost for the penalty cost policy may be higher than that of the previous policies (as is the case for items S03, S05, S09, and S10), lower than that of the previous policies (as is the case for items S04 and S07), or may lie between the average costs for the two previous policies (as is the case for the remaining items). For example, the average cost may be lower because a lower service level is implied by the specified penalty costs. On the other hand, the penalty cost heuristics include penalty costs for backorders; these costs are not included in the service level heuristics, so an increase in average cost for the penalty cost method may result from including this extra cost parameter. Therefore, caution should be used when comparing output from the service level method and penalty cost method, as the two methods use different levels of information to compute costs and determine policies.

Output 2.2.1. Inventory Policies Using Penalty Cost Heuristic

PROC IRP Results
Penalty Cost Heuristic

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	93	183	77.418	3.3069	0.38244	111.628
2	S02	JKL Company	176	211	92.418	3.9176	1.00000	131.680
3	S03	XYZ Company	81	138	41.947	1.3832	0.40027	71.447
4	S04	XYZ Company	427	538	217.530	11.2186	0.49855	243.491
5	S05	QRS Company	74	115	40.967	0.4669	0.21771	28.096
6	S06	QRS Company	998	999	461.089	14.0894	0.99995	440.345
7	S07	ABC Company	457	467	198.559	13.5939	0.99491	344.842
8	S08	JKL Company	83	208	67.504	2.0040	0.16000	48.141
9	S09	ABC Company	9	32	12.855	0.1058	0.24150	12.629
10	S10	ABC Company	389	656	243.266	2.5412	0.24993	156.376

Obs	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	1.98508	0.08479	0.50376	0.94017	0.90844	PC-SS-GA	SUCCESSFUL
2	2.64050	0.11193	0.37872	0.94287	0.91467	PC-NQ-GA	SUCCESSFUL
3	1.61333	0.05320	0.61984	0.94865	0.90175	PC-SS-NO	SUCCESSFUL
4	2.90041	0.14958	0.34478	0.92173	0.90005	PC-SS-GA	SUCCESSFUL
5	4.55188	0.05188	0.21969	0.95979	0.95229	PC-RQ-NO	SUCCESSFUL
6	5.01184	0.15315	0.19953	0.94397	0.92963	PC-BS-GA	SUCCESSFUL
7	2.11232	0.14462	0.47341	0.91689	0.88398	PC-SS-GA	SUCCESSFUL
8	3.37520	0.10020	0.29628	0.92066	0.90307	PC-NQ-NO	SUCCESSFUL
9	2.57091	0.02115	0.38897	0.97892	0.94960	PC-SS-NO	SUCCESSFUL
10	3.92365	0.04099	0.25486	0.96791	0.95415	PC-SS-NO	SUCCESSFUL

Example 2.3. Single-Location System: OPTIMAL Option

In Example 2.2, the retailer used penalty costs to compute nearly optimal inventory replenishment policies. By specifying the OPTIMAL option on the PENALTY statement, you can compute optimal policies using backorder penalty costs. PROC IRP computes the optimal reorder level and order-up-to-level within the class of policy specified by the policyType variable (that is, SS, BS, NQ, or RQ).

The call to PROC IRP is shown in the following SAS code. The OPTIMAL option is specified on the PENALTY statement. In addition, the LOTSIZE=, MINSIZE=, and MAXFREQ= options are no longer included on the REPLENISHMENT statement, since these options are ignored when the OPTIMAL option is used.

```
proc irp data=in2 out=out3 method=penalty;
   holdingcost holdingCost;
   itemid sku supplier;
   leadtime / mean=LTmean variance=LTvar;
   penalty / cost=penaltyCost optimal;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.3.1. Notice that the average cost for most items (except for S03, S05, and S08) is lower than the average cost in Output 2.2.1. This is expected, as the OPTIMAL option finds the optimal (i.e., lowest-cost) inventory replenishment policy. However, the average cost for the remaining items actually rises. There are two reasons why this might happen. First, the penalty cost heuristic given in Output 2.2.1 uses an approximation to the cost of the policy; the actual cost may be slightly higher or lower than the value given in the avgCost variable. Moreover, the heuristic uses either a gamma distribution or a normal distribution to approximate both the lead-time demand and (lead time + review time)-demand, whereas the optimization uses either a negative binomial distribution or a shifted Poisson distribution. Therefore, the underlying assumptions of the models are different, and care should be used when comparing results across the two models. The policy calculated when using the OPTIMAL option is the optimal policy with respect to the lead time and demand distributions used by PROC IRP, but may reflect a higher cost than a policy calculated using different distributions for lead time and demand.

In this example, the negative binomial distribution is used for the (lead time + review time)-demand of all items, as indicated by a 'B' in the fifth character of the _algorithm_ variable. This distribution is also used for the review-time demand, as indicated by a 'B' in the sixth character of the _algorithm_ variable. Note that the sixth character of the _algorithm_ variable is '_' for items following an 'NQ' policy, which indicates that the review-time demand distribution does not play a role in the optimization algorithm.

Recall from Example 2.1 that the fixed cost for item S10 was not easily estimated, so a maximum ordering frequency was used instead. However, the OPTIMAL option ignores the LOTSIZE=, MINSIZE=, and MAXFREQ= options, so item S10 is no longer constrained by a maximum ordering frequency of 25%. In addition, because the fixed cost for item S10 was not specified, PROC IRP assumes it is zero. As a result, the policy for S10 in Output 2.3.1 is a base-stock policy (as indicated by the 'BS' in the _algorithm_ variable), and the reorderLevel and orderUpToLevel are quite different from those in the previous examples. However, the original intention of including a missing value for the fixed cost for S10 was to account for the fact that the cost was unknown, not to imply that the cost was zero. Therefore, when using the OPTIMAL option, you should specify estimates for fixed costs of all items, unless the fixed cost is assumed to be zero.

Output 2.3.1. Inventory Policies Using the OPTIMAL Option

PROC IRP Results
Penalty Cost with OPTIMAL Option

Obs	sku	supplier	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost
1	S01	ABC Company	93	188	77.538	3.4709	0.32194	108.700
2	S02	JKL Company	173	213	92.623	3.8979	0.87719	131.308
3	S03	XYZ Company	83	138	41.674	1.7570	0.36451	72.360
4	S04	XYZ Company	441	525	214.188	11.2445	0.54722	242.016
5	S05	QRS Company	76	117	43.684	0.6844	0.21791	31.405
6	S06	QRS Company	999	1000	461.055	13.9348	0.99252	438.931
7	S07	ABC Company	375	522	206.377	13.8543	0.44856	321.338
8	S08	JKL Company	83	208	68.445	2.4453	0.16000	50.141
9	S09	ABC Company	8	31	11.429	0.2308	0.19531	11.455
10	S10	ABC Company	526	527	220.297	3.2970	0.99800	152.365

Obs	inventory Ratio	backorder Ratio	turnover	ready Rate	_scale_	_algorithm_	_status_
1	1.98815	0.08900	0.50298	0.90281	1.00	PC-SS-BB	SUCCESSFUL
2	2.64637	0.11137	0.37788	0.91369	1.05	PC-NQ-B_	SUCCESSFUL
3	1.60286	0.06758	0.62388	0.89413	1.00	PC-SS-BB	SUCCESSFUL
4	2.85584	0.14993	0.35016	0.89768	3.00	PC-SS-BB	SUCCESSFUL
5	4.85382	0.07605	0.20602	0.94879	1.00	PC-RQ-BB	SUCCESSFUL
6	5.01147	0.15147	0.19954	0.92892	5.52	PC-BS-BB	SUCCESSFUL
7	2.19550	0.14739	0.45548	0.88118	2.82	PC-SS-BB	SUCCESSFUL
8	3.42226	0.12226	0.29220	0.90366	1.00	PC-NQ-B_	SUCCESSFUL
9	2.28584	0.04615	0.43748	0.90048	1.00	PC-SS-BB	SUCCESSFUL
10	3.55318	0.05318	0.28144	0.95311	3.10	PC-BS-BB	SUCCESSFUL

Example 2.4. Single-Location System: LEADTIMEDEMAND Statement

This example illustrates the use of PROC IRP for a retailer that faces nonstationary demand with a lead time that is longer than the review period. The IRP procedure uses the review-time demand and lead time information to calculate the parameters of lead-time demand. When demand is nonstationary (i.e., demand fluctuates over time), it is not sufficient to know just the lead time and mean review-time demand information. In such situations, you can directly specify the mean and variance of lead-time demand with the LEADTIMEDEMAND statement.

For example, suppose the lead time for an item is three periods, but the demands over the next four review periods are 25, 32, 40, and 28. If the mean of the review-time demand is specified as 25 (the mean of the current period's demand), and the lead-time mean is specified as 3 using the LEADTIME statement, PROC IRP computes the mean lead-time demand as 75 (= 3 $\times$ 25). This is an inaccurate calculation of lead-time demand, as it does not account for the fluctuations in demand in the subsequent periods. Rather, the correct calculation of lead-time demand is the demand over the next three periods following the current review period, which is 100 (= 32 + 40 + 28). This example illustrates how the LEADTIMEDEMAND statement is used to overcome such a problem.

The data set in4 gives the input to PROC IRP. The mean and variance of lead-time demand are given by the LTDmean and LTDvar variables, respectively. The mean and variance of review-time demand are given by the RTDmean and RTDvar variables, respectively. Note that the mean and variance of lead time are not included in this data set. When using the LEADTIMEDEMAND statement, these variables are not used.

```
data in4;
   format sku $3.;
   input  sku $ holdingCost fixedCost
          LTDmean LTDvar RTDmean RTDvar
          serviceLevel;
datalines;
B01  0.52  62  100   894  25   56  0.95
B02  0.86  17   80   633  50  227  0.95
B03  0.27  48  275  4101  90  506  0.95
B04  0.94  23   64   719  15   38  0.95
B05  0.62  38   90  1188  32  163  0.95
B06  0.44  82  122  4324  52  675  0.95
B07  0.75  68  170  2823  84  632  0.95
B08  0.78  73   30   365  10   35  0.95
B09  0.46  18   91   989  66  533  0.95
B10  0.55  25  144  3741  71  807  0.95
;
```

The following call to PROC IRP computes inventory replenishment policies using a 95% target fill rate. In the PROC IRP statement, the METHOD= option is not specified, so the default value of SERVICE is used. The HOLDINGCOST statement is not required because the holding cost variable is named holdingCost, the default name for PROC IRP. Because the POLICYTYPE statement is not specified, PROC IRP computes (s, S) policies for all items in the data set. In addition, the REPLENISHMENT statement is not required because the fixed cost variable is named fixedCost, the default name for PROC IRP. Finally, the SERVICE statement is not specified, so PROC IRP uses fill rate, the default service measure, for all items.

```
proc irp data=in4 out=out4;
   itemid sku;
   leadtimedemand / mean=LTDmean variance=LTDvar;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.4.1. Note that because the LEADTIMEDEMAND statement was used, these policies should be interpreted as the policies to follow only for the current review period. Because demand is nonstationary and the lead times are longer than the review period, you should compute new policies each period, using updated information about lead-time demand and review-time demand.

Output 2.4.1. Inventory Policies Using the LEADTIMEDEMAND Statement

PROC IRP Results
Target Measure: 95% Fill Rate

Obs	sku	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost	inventory Ratio
1	B01	131	214	58.397	1.41382	0.30916	49.535	2.33588
2	B02	130	177	40.256	2.35698	0.75140	47.394	0.80511
3	B03	356	539	118.050	4.52323	0.44883	53.417	1.31166
4	B04	102	134	44.889	0.88307	0.41680	51.782	2.99259
5	B05	137	205	61.300	1.65391	0.41876	53.919	1.91563
6	B06	209	361	134.347	3.64541	0.33104	86.258	2.58360
7	B07	257	385	94.237	4.10412	0.52055	106.075	1.12186
8	B08	51	101	41.743	0.93225	0.22602	49.059	4.17433
9	B09	156	231	57.522	3.22583	0.63384	37.869	0.87154
10	B10	247	334	98.831	3.57699	0.59779	69.302	1.39199

Obs	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	0.056553	0.42810	0.95271	0.91768	FR-SS-NO	SUCCESSFUL
2	0.047140	1.24207	0.95327	0.86847	FR-SS-NO	SUCCESSFUL
3	0.050258	0.76239	0.95215	0.89042	FR-SS-NO	SUCCESSFUL
4	0.058872	0.33416	0.95552	0.93183	FR-SS-NO	SUCCESSFUL
5	0.051685	0.52202	0.95373	0.91486	FR-SS-NO	SUCCESSFUL
6	0.070104	0.38706	0.95191	0.92702	FR-SS-GA	SUCCESSFUL
7	0.048859	0.89137	0.95231	0.88285	FR-SS-NO	SUCCESSFUL
8	0.093225	0.23956	0.95425	0.93756	FR-SS-GA	SUCCESSFUL
9	0.048876	1.14740	0.95139	0.86808	FR-SS-NO	SUCCESSFUL
10	0.050380	0.71840	0.95214	0.89932	FR-SS-NO	SUCCESSFUL

Example 2.5. Continuous Review Approximation: Review Period Shorter Than Forecast Interval

In this example, consider a retailer that forecasts demand data on a monthly basis but reviews inventory on a weekly basis. For the purpose of this illustration, it is assumed that there are exactly four weeks in a month.

For example, consider Table 2.1 on page 16 in the "Getting Started" section, but suppose that the mean and variance of demand specify the demand over one month. In addition, suppose the lead time of all items is one week (the same as the review period). This is not an example of continuous review, since the retailer still makes decisions at discrete time periods. However, it may be considered an approximation to a continuous review system, because decisions are made at points throughout the demand forecast interval. By choosing smaller review periods (for example, one day or one hour), this becomes a closer approximation to a continuous review system.

The data for this example are given in the following data set, data5. The variables MeanOfDemand and VarianceOfDemand give the mean and variance of demand over an entire month.

```
data data5;
   format Sku $1.;
   input  Sku $ HoldingCost OrderingCost
          LeadTime MeanOfDemand VarianceOfDemand;
datalines;
A  0.35  90  1  125.1  2170.8
B  0.05  50  1  140.3  1667.7
C  0.12  50  1  116.0  3213.4
D  0.10  75  1  291.8  5212.4
E  0.45  75  1  134.5  1980.5
;
```

This data set is transformed to the input data set for PROC IRP using the following DATA step. From the assumption that there are four weeks in a month, the mean and variance of review-time demand (RTDmean and RTDvar, respectively) are calculated by dividing MeanOfDemand by 4 and VarianceOfDemand by 16. For this calculation to be valid, the demand for one month must be assumed to be uniform over the entire month, so that the demand for a single week is one quarter of the monthly demand.

```
data in5;
   set data5;
   RTDmean = MeanOfDemand / 4 ;
   RTDvar = VarianceOfDemand / 16 ;
   serviceLevel = 0.96 ;
run;
```

The call to PROC IRP is as follows. Notice that the VARIANCE= option is not specified on the LEADTIME statement, since the lead times are assumed to be deterministic (that is, zero variance).

```
proc irp data=in5 out=out5 method=service;
   holdingcost HoldingCost;
   itemid Sku;
   leadtime / mean=LeadTime;
   replenishment / fcost=OrderingCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel;
run;
```

The output data set from this call to PROC IRP is given in Output 2.5.1.

Output 2.5.1. Inventory Policies When Review Period Is Shorter Than Forecast Interval

PROC IRP Results
Target Measure: 96% Fill Rate

Obs	Sku	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq	avgCost	inventory Ratio
1	A	47	169	59.268	1.10664	0.27373	45.3798	1.89505
2	B	35	277	105.737	1.29462	0.18716	14.6450	3.01461
3	C	42	192	73.833	1.11256	0.22262	19.9912	2.54598
4	D	95	404	135.546	2.82158	0.25877	32.9626	1.85807
5	E	53	155	50.198	1.19067	0.32802	47.1905	1.49287

Obs	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	0.035384	0.52769	0.96471	0.91056	FR-SS-NO	SUCCESSFUL
2	0.036910	0.33172	0.96317	0.91166	FR-SS-NO	SUCCESSFUL
3	0.038364	0.39278	0.96206	0.92196	FR-SS-NO	SUCCESSFUL
4	0.038678	0.53819	0.96133	0.88895	FR-SS-NO	SUCCESSFUL
5	0.035410	0.66985	0.96463	0.89963	FR-SS-NO	SUCCESSFUL

Example 2.6. Two-Echelon System: Service Level Heuristic

This example illustrates the use of PROC IRP to compute inventory replenishment policies for a two-echelon system using service level heuristics. Example 2.7 then explores the same two-echelon system using penalty costs.

In this example, there are two warehouses and four retailers. There are two items (M01 and M02), but these items can be classified further by color (BLUE or RED). These items could be identified by four distinct values of the SKU variable; however, to illustrate the use of the ITEMID statement, they will be identified by SKU (M01 or M02) and COLOR (BLUE or RED). Warehouse W01 supplies item M01, and warehouse W02 supplies item M02. From warehouse W01, only retailers R01 and R02 require blue items, while retailers R01, R03, and R04 require red items. From warehouse W02, all four retailers (R01, R02, R03, and R04) require blue items, while only retailer R03 requires red items.

Estimates of the holding and fixed costs, mean and variance of the lead time, and mean and variance of the review-time demand are given in Table 2.19. Observations with a missing value for Retailer correspond to warehouses. For example, the first observation gives the holding cost and fixed cost at warehouse W01, in addition to the mean and variance of lead time from an external supplier to this warehouse. The mean and variance of review-time demand are missing for all the warehouse observations, since the warehouses do not see any external demand apart from the orders placed by the retailers.

The remaining observations correspond to retailers. The missing values of Fixed Cost indicate that the retailers incur no fixed cost for placing an order; therefore, the retailers follow base-stock policies. For these observations, the mean and variance of lead time give data about the lead time from the warehouse to the retailer. In this

problem, the lead time variance between the warehouses and the retailers is assumed to be zero, but positive variances can also be used in the two-echelon system. The review-time demand data give the mean and variance of the demand for that item (SKU and Color) at that retailer.

Table 2.19. Data Estimates for Two-Echelon System

SKU	Color	Warehouse	Retailer	Holding Cost	Fixed Cost	Lead Time		Review-Time Demand	
						Mean	Variance	Mean	Variance
M01	Blue	W01	.	0.20	61	1	0.12	.	.
M01	Blue	W01	R01	0.75	.	2	0	67	121
M01	Blue	W01	R02	1.42	.	1	0	23	87
M01	Red	W01	.	0.20	61	1	0.12	.	.
M01	Red	W01	R01	0.75	.	2	0	50	793
M01	Red	W01	R03	1.11	.	3	0	42	109
M01	Red	W01	R04	0.65	.	2	0	91	1267
M02	Blue	W02	.	0.17	88	1	0.41	.	.
M02	Blue	W02	R01	0.70	.	1	0	84	931
M02	Blue	W02	R02	1.35	.	2	0	59	1018
M02	Blue	W02	R03	1.04	.	1	0	71	775
M02	Blue	W02	R04	0.62	.	2	0	113	1689
M02	Red	W02	.	0.17	88	1	0.41	.	.
M02	Red	W02	R03	1.04	.	1	0	85	1954

Using this information, inventory policies are calculated using (s, S) policies for the warehouses and a target fill rate of 97% for the retailers. This means that 97% of all incoming customer orders (to the retailers) can be filled from on-hand inventory. The information is stored in the following data set in6_fr:

```
data in6_fr;
   format warehouse $3. retailer $3. sku $3. color $4.
          policyType $2. serviceType $2. ;
   input sku $ color $ warehouse $ retailer $
         holdingCost fixedCost
         LTmean LTvar RTDmean RTDvar
         policyType $ serviceType $ serviceLevel;
   datalines;
M01  BLUE  W01    .  0.20  61  1  0.12    .     .  SS   .     .
M01  BLUE  W01  R01  0.75   .  2  0      67   121   .  FR  0.97
M01  BLUE  W01  R02  1.42   .  1  0      23    87   .  FR  0.97
M01  RED   W01    .  0.20  61  1  0.12    .     .  SS   .     .
M01  RED   W01  R01  0.75   .  2  0      50   793   .  FR  0.97
M01  RED   W01  R03  1.11   .  3  0      42   109   .  FR  0.97
M01  RED   W01  R04  0.65   .  2  0      91  1267   .  FR  0.97
M02  BLUE  W02    .  0.17  88  1  0.41    .     .  SS   .     .
M02  BLUE  W02  R01  0.70   .  1  0      84   931   .  FR  0.97
M02  BLUE  W02  R02  1.35   .  2  0      59  1018   .  FR  0.97
M02  BLUE  W02  R03  1.04   .  1  0      71   775   .  FR  0.97
M02  BLUE  W02  R04  0.62   .  2  0     113  1689   .  FR  0.97
M02  RED   W02    .  0.17  88  1  0.41    .     .  SS   .     .
M02  RED   W02  R03  1.04   .  1  0      85  1954   .  FR  0.97
;
```

The following call to PROC IRP computes the inventory policies. Because the METHOD= option is specified as SERVICE, heuristics are used to compute inventory policies based on target service levels (in this case, 97% fill rate). The ITEMID statement specifies sku, color, and warehouse as the variables by which to group the items. The LOCATION statement specifies the retailer variable. The remaining variables in the input data set are specified using the HOLDINGCOST, LEADTIME, POLICYTYPE, REPLENISHMENT, REVIEWTIMEDEMAND, and SERVICE statements.

```
proc irp data=in6_fr out=out6_fr method=service;
   holdingcost holdingCost;
   itemid sku color warehouse;
   leadtime / mean=LTmean variance=LTvar;
   location retailer;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.6.1. This data set contains two variables that define the computed policy: reorderLevel and orderUpToLevel. The retailers follow base-stock policies, so the orderUpToLevel is one more than the reorderLevel for the retailers. Note that the value of fillRate for the retailers is very near 97%, the target service level. The fill rate for the warehouses is lower, but the fill rate at the retailers is the main concern since customers are only seen at the retailers.

Output 2.6.1. Inventory Policies Using 97% Target Fill Rate

PROC IRP Results for Two-Echelon System
Target Measure: 97% Fill Rate

Obs	sku	color	warehouse	retailer	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq
1	M01	BLUE	W01		138	365	101.849	5.4317	0.36590
2	M01	BLUE	W01	R01	227	228	25.029	2.0728	1.00000
3	M01	BLUE	W01	R02	67	68	21.321	0.7094	1.00000
4	M01	RED	W01		260	593	137.053	17.1120	0.47980
5	M01	RED	W01	R01	245	246	92.928	1.6038	1.00000
6	M01	RED	W01	R03	202	203	32.322	1.2494	1.00000
7	M01	RED	W01	R04	382	383	104.305	2.8142	1.00000
8	M02	BLUE	W02		588	1182	363.979	22.1052	0.48857
9	M02	BLUE	W02	R01	242	243	71.868	2.5462	1.00000
10	M02	BLUE	W02	R02	282	283	103.859	1.8473	1.00000
11	M02	BLUE	W02	R03	210	211	66.396	2.1958	1.00000
12	M02	BLUE	W02	R04	460	461	117.812	3.4509	1.00000
13	M02	RED	W02		160	463	182.018	5.6619	0.28765
14	M02	RED	W02	R03	291	292	119.059	2.7210	1.00000

Obs	avgCost	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	91.738	1.13165	0.060352	0.88366	0.94121	0.84200	__-SS-GA	SUCCESSFUL
2	18.772	0.37357	0.030937	2.67688	0.96907	0.84331	FR-BS-GA	SUCCESSFUL
3	30.276	0.92701	0.030845	1.07873	0.96950	0.91825	FR-BS-GA	SUCCESSFUL
4	230.050	0.74892	0.093508	1.33525	0.91030	0.78621	__-SS-GA	SUCCESSFUL
5	69.696	1.85857	0.032075	0.53805	0.97041	0.94846	FR-BS-GA	SUCCESSFUL
6	35.878	0.76957	0.029748	1.29942	0.97049	0.90239	FR-BS-GA	SUCCESSFUL
7	67.798	1.14621	0.030925	0.87244	0.96991	0.92666	FR-BS-GA	SUCCESSFUL
8	437.483	1.11309	0.067600	0.89840	0.94393	0.87098	__-SS-GA	SUCCESSFUL
9	50.307	0.85557	0.030311	1.16881	0.97006	0.91412	FR-BS-GA	SUCCESSFUL
10	140.210	1.76032	0.031310	0.56808	0.97084	0.94707	FR-BS-GA	SUCCESSFUL
11	69.052	0.93516	0.030927	1.06934	0.96956	0.91862	FR-BS-GA	SUCCESSFUL
12	73.044	1.04259	0.030539	0.95915	0.97010	0.92201	FR-BS-GA	SUCCESSFUL
13	180.077	2.14139	0.066611	0.46699	0.94745	0.91374	__-SS-GA	SUCCESSFUL
14	123.821	1.40069	0.032012	0.71393	0.96954	0.93921	FR-BS-GA	SUCCESSFUL

Suppose that the target service measure for retailer R01 is instead specified as a 3% backorder ratio. The remaining retailers continue to follow policies based on a 97% target fill rate. The DATA step to change the serviceType and serviceLevel variables is as follows.

```
data in6_br;
   set in6_fr;
   if retailer = 'R01' then do;
      serviceType = 'BR';
      serviceLevel = 0.03;
   end;
run;
```

The call to PROC IRP is as follows. Note that this call is exactly the same as the previous call to PROC IRP, except for a different name for the output data set. Some of the variable values (for the serviceLevel and serviceType variables) have changed, but the variable names have not changed.

```
proc irp data=in6_br out=out6_br method=service;
   holdingcost holdingCost;
   itemid sku color warehouse;
   leadtime / mean=LTmean variance=LTvar;
   location retailer;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
   service / level=serviceLevel type=serviceType;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.6.2. The values of the backorderRatio variable are near 3% for the three observations that correspond to retailer R01. Note that when the policies for retailer R01 change, the policies for the other retailers and the policies for the warehouses may also change as a result of the changes in the target service level for R01.

Output 2.6.2. Inventory Policies Using 3% Target Backorder Ratio for R01

PROC IRP Results for Two-Echelon System
Target Measure: 3% Backorder Ratio

Obs	sku	color	warehouse	retailer	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq
1	M01	BLUE	W01		142	369	105.242	4.8250	0.36590
2	M01	BLUE	W01	R01	227	228	25.235	1.9366	1.00000
3	M01	BLUE	W01	R02	67	68	21.416	0.6863	1.00000
4	M01	RED	W01		271	604	145.821	14.8801	0.47980
5	M01	RED	W01	R01	246	247	94.420	1.4857	1.00000
6	M01	RED	W01	R03	201	202	31.817	1.2321	1.00000
7	M01	RED	W01	R04	379	380	102.451	2.8499	1.00000
8	M02	BLUE	W02		588	1182	363.979	22.1052	0.48857
9	M02	BLUE	W02	R01	242	243	71.868	2.5462	1.00000
10	M02	BLUE	W02	R02	282	283	103.859	1.8473	1.00000
11	M02	BLUE	W02	R03	210	211	66.396	2.1958	1.00000
12	M02	BLUE	W02	R04	460	461	117.812	3.4509	1.00000
13	M02	RED	W02		160	463	182.018	5.6619	0.28765
14	M02	RED	W02	R03	291	292	119.059	2.7210	1.00000

Obs	avgCost	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	92.705	1.16936	0.053611	0.85517	0.94768	0.85460	__-SS-GA	SUCCESSFUL
2	18.926	0.37665	0.028904	2.65502	0.97110	0.84992	BR-BS-GA	SUCCESSFUL
3	30.410	0.93111	0.029837	1.07398	0.97047	0.92021	FR-BS-GA	SUCCESSFUL
4	231.157	0.79684	0.081312	1.25496	0.92177	0.80780	__-SS-GA	SUCCESSFUL
5	70.815	1.88840	0.029714	0.52955	0.97247	0.95168	BR-BS-GA	SUCCESSFUL
6	35.317	0.75755	0.029335	1.32005	0.97087	0.90220	FR-BS-GA	SUCCESSFUL
7	66.593	1.12583	0.031318	0.88823	0.96948	0.92511	FR-BS-GA	SUCCESSFUL
8	437.483	1.11309	0.067600	0.89840	0.94393	0.87098	__-SS-GA	SUCCESSFUL
9	50.307	0.85557	0.030311	1.16881	0.97006	0.91412	BR-BS-GA	SUCCESSFUL
10	140.210	1.76032	0.031310	0.56808	0.97084	0.94707	FR-BS-GA	SUCCESSFUL
11	69.052	0.93516	0.030927	1.06934	0.96956	0.91862	FR-BS-GA	SUCCESSFUL
12	73.044	1.04259	0.030539	0.95915	0.97010	0.92201	FR-BS-GA	SUCCESSFUL
13	180.077	2.14139	0.066611	0.46699	0.94745	0.91374	__-SS-GA	SUCCESSFUL
14	123.821	1.40069	0.032012	0.71393	0.96954	0.93921	FR-BS-GA	SUCCESSFUL

Example 2.7. Two-Echelon System: Penalty Costs

In this example, assume that estimates of backorder penalty costs are known for the problem in Example 2.6. Rather than using service level heuristics, as in Example 2.6, a penalty cost heuristic is used to calculate inventory policies.

The penalty costs are given in the following data set. There are no penalty costs for backorders at the warehouses, since customers are only seen at the retailers.

```
data pcosts;
   format warehouse $3. retailer $3. sku $3. color $4. ;
   input sku $ color $ warehouse $ retailer $ penaltyCost;
   datalines;
M01  BLUE  W01    .     .
M01  BLUE  W01  R01   6.7
M01  BLUE  W01  R02  10.2
M01  RED   W01    .     .
M01  RED   W01  R01   8.4
M01  RED   W01  R03   5.6
M01  RED   W01  R04   9.1
M02  BLUE  W02    .     .
M02  BLUE  W02  R01   3.4
M02  BLUE  W02  R02   6.9
M02  BLUE  W02  R03   7.7
M02  BLUE  W02  R04  12.4
M02  RED   W02    .     .
M02  RED   W02  R03   7.5
;
```

This data set is merged with in6_fr to produce the input data set in7. The serviceLevel and serviceType variables are dropped from the in6_fr data set, as these will not be needed when using penalty costs. However, if these variables are left in the data set, they are simply ignored when the METHOD= option is specified as PENALTY.

```
data in7;
   merge in6_fr (drop=serviceLevel serviceType)
         pcosts;
   by sku color warehouse retailer;
run;
```

The call to PROC IRP is as follows. There are two main differences between this call to PROC IRP and the call in Example 2.6. First, METHOD=PENALTY is specified on the PROC IRP statement, to indicate that a penalty cost heuristic should be used to compute the policies. Also, the SERVICE statement is removed and the PENALTY statement is added to specify the variable in the input data set that gives the penalty costs.

```
proc irp data=in7 out=out7 method=penalty;
   holdingcost holdingCost;
   itemid sku color warehouse;
   leadtime / mean=LTmean variance=LTvar;
   location retailer;
   penalty / cost=penaltyCost;
   policytype policyType;
   replenishment / fcost=fixedCost;
   reviewtimedemand / mean=RTDmean variance=RTDvar;
run;
```

The output data set from this call to PROC IRP is shown in Output 2.7.1.

Output 2.7.1. Inventory Policies Using Penalty Cost Heuristic

PROC IRP Results for Two-Echelon System
Penalty Cost Heuristic

Obs	sku	color	warehouse	retailer	reorder Level	order UpTo Level	avg Inventory	avg Backorder	avg Order Freq
1	M01	BLUE	W01		150	377	112.176	3.7594	0.36590
2	M01	BLUE	W01	R01	231	232	29.274	1.1642	1.00000
3	M01	BLUE	W01	R02	62	63	17.135	1.1271	1.00000
4	M01	RED	W01		279	612	152.344	13.4028	0.47980
5	M01	RED	W01	R01	227	228	76.983	2.6447	1.00000
6	M01	RED	W01	R03	192	193	24.233	2.3090	1.00000
7	M01	RED	W01	R04	382	383	105.852	2.5170	1.00000
8	M02	BLUE	W02		614	1208	386.836	18.9625	0.48857
9	M02	BLUE	W02	R01	216	217	49.642	5.5133	1.00000
10	M02	BLUE	W02	R02	234	235	61.109	6.5300	1.00000
11	M02	BLUE	W02	R03	197	198	55.173	3.2907	1.00000
12	M02	BLUE	W02	R04	481	482	138.413	1.9658	1.00000
13	M02	RED	W02		172	475	193.064	4.7078	0.28765
14	M02	RED	W02	R03	254	255	85.936	5.7170	1.00000

Obs	avgCost	inventory Ratio	backorder Ratio	turnover	fill Rate	ready Rate	_algorithm_	_status_
1	110.339	1.24641	0.04177	0.80231	0.95911	0.87870	__-SS-GA	SUCCESSFUL
2	29.756	0.43693	0.01738	2.28871	0.98262	0.89901	PC-BS-GA	SUCCESSFUL
3	35.828	0.74500	0.04901	1.34228	0.95155	0.87425	PC-BS-GA	SUCCESSFUL
4	271.227	0.83248	0.07324	1.20123	0.92940	0.82275	__-SS-GA	SUCCESSFUL
5	79.953	1.53966	0.05289	0.64950	0.95157	0.91782	PC-BS-GA	SUCCESSFUL
6	39.829	0.57698	0.05498	1.73317	0.94555	0.83212	PC-BS-GA	SUCCESSFUL
7	91.709	1.16321	0.02766	0.85969	0.97298	0.93254	PC-BS-GA	SUCCESSFUL
8	482.715	1.18299	0.05799	0.84532	0.95195	0.88700	__-SS-GA	SUCCESSFUL
9	53.495	0.59098	0.06563	1.69211	0.93533	0.82819	PC-BS-GA	SUCCESSFUL
10	127.554	1.03574	0.11068	0.96549	0.90022	0.83484	PC-BS-GA	SUCCESSFUL
11	82.718	0.77709	0.04635	1.28685	0.95440	0.88158	PC-BS-GA	SUCCESSFUL
12	110.192	1.22489	0.01740	0.81640	0.98287	0.95239	PC-BS-GA	SUCCESSFUL
13	190.384	2.27134	0.05539	0.44027	0.95623	0.92693	__-SS-GA	SUCCESSFUL
14	132.251	1.01101	0.06726	0.98911	0.93632	0.87851	PC-BS-GA	SUCCESSFUL

Statement and Option Cross-Reference Tables

The following tables reference the statements and options in the IRP procedure that are illustrated by the examples in this section.

Table 2.20. Statements Specified in Examples

Statement	Examples						
	1	2	3	4	5	6	7
HOLDINGCOST	X	X	X		X	X	X
ITEMID	X	X	X	X	X	X	X
LEADTIME	X	X	X		X	X	X
LEADTIMEDEMAND				X			
LOCATION						X	X
PENALTY		X	X				X
POLICYTYPE	X	X	X			X	X
REPLENISHMENT	X	X	X		X	X	X
REVIEWTIMEDEMAND	X	X	X	X	X	X	X
SERVICE	X				X	X	

Table 2.21. Options Specified in Examples

Option	Examples						
	1	2	3	4	5	6	7
COST=		X	X				X
DATA=	X	X	X	X	X	X	X
FCOST=	X	X	X		X	X	X
LEVEL=	X				X	X	
LOTSIZE=	X	X	X				
MAXFREQ=	X	X	X				
MEAN= (LEADTIME)	X	X	X		X	X	X
MEAN= (LEADTIMEDEMAND)				X			
MEAN= (REVIEWTIMEDEMAND)	X	X	X	X	X	X	X
METHOD=	X	X	X		X	X	X
MINSIZE=	X	X	X				
OPTIMAL			X				
OUT=	X	X	X	X	X	X	X
TYPE=	X					X	
VARIANCE= (LEADTIME)	X	X	X			X	X
VARIANCE= (LEADTIMEDEMAND)				X			
VARIANCE= (REVIEWTIMEDEMAND)	X	X	X	X	X	X	X

References

Ehrhardt, R. and Mosier, C. (1984), "A Revision of the Power Approximation for Computing (s, S) Policies," *Management Science*, 30, 618–622.

Graves, S. C., Rinnooy Kan, A. H. G., and Zipkin, P. H., eds. (1993), *Handbooks in Operations Research and Management Science, 4: Logistics of Production and Inventory*, Netherlands: North-Holland.

Matta, K. F. and Sinha, D. (1995), "Policy and Cost Approximations of Two-Echelon Distribution Systems with a Procurement Cost at the Higher Echelon," *IIE Transactions*, 27, 638–645.

Schneider, H. (1978), "Methods for Determining the Re-order Point of an (s, S) Ordering Policy When a Service Level Is Specified," *Journal of the Operational Research Society*, 29, 1181–1193.

Schneider, H. (1981), "Effects of Service-Levels on Order-Points or Order-Levels in Inventory Models," *International Journal of Production Research*, 19, 615–631.

Schneider, H. and Ringuest, J. L. (1990), "Power Approximation for Computing (s, S) Policies Using Service Level," *Management Science*, 36, 822–834.

Silver, E. A., Pyke, D. F., and Peterson, R. (1998), *Inventory Management and Production Planning and Scheduling*, New York: John Wiley and Sons, Inc.

Svoronos, A. and Zipkin, P. H. (1991), "Evaluation of One-for-One Replenishment Policies for Multi-Echelon Inventory Systems," *Management Science*, 37, 68–83.

Tijms, H. and Groenevelt, H. (1984), "Simple Approximations for the Reorder Point in Periodic and Continuous Review (s, S) Inventory Systems with Service Level Constraints," *European Journal of Operations Research*, 17, 175–190.

Zheng, Y. and Chen, F. (1992), "Inventory Policies with Quantized Ordering," *Naval Research Logistics*, 39, 285–305.

Zheng, Y. and Federgruen, A. (1992), "Finding Optimal (s, S) Policies Is About as Simple as Evaluating a Single Policy," *Operations Research*, 39, 654–665.

Zipkin, P. H. (2000), *Foundations of Inventory Management*, New York: McGraw-Hill.

Chapter 3
The MIRP Procedure

Chapter Contents

Chapter 3
The MIRP Procedure

Overview

PROC MIRP is a procedure designed for inventory replenishment planning. For a supply chain network, inventory replenishment planning answers three basic questions about inventory: where should it be allocated? when should it be allocated? and how much of it should be allocated? A replenishment plan consists of control parameters that determine replenishment quantities for each product at each location and at each period. PROC MIRP optimizes these control parameters so that service level requirements are satisfied at minimum inventory costs.

Getting Started

There are two inventory control theories: discrete-time theory and continuous-time theory. The discrete version assumes that inventory replenishment orders only occur periodically, i.e., only at certain time points. The continuous version, on the other hand, enables replenishment orders to occur at any time.

The discrete-time theory is closer to situations in practice and is the underlying methodology of PROC MIRP. Since replenishment orders occur periodically, a *base period* must be defined before using PROC MIRP. The base period is the time between two replenishment orders that may be placed. It could be any time unit, such as one day, one week, or one month. For simplicity, the term *period* refers to the base period.

In this section, examples with three different network structures are discussed: single location, two-echelon distribution network, and two-echelon assembly network. Each example starts with a problem description. Input data sets are given to demonstrate how each problem can be modeled. Then statements of PROC MIRP and output data sets are presented. Each of the first three examples computes for only one network. The forth example demonstrates how to organize input data sets so that a single call to PROC MIRP solves for multiple networks.

Single Location

A product R is sold at a store W. Replenishment orders are placed once every Monday morning and are received one week later from an external supplier. The product faces a fairly stable demand in the next eight weeks with a mean of 10 units and a variance of 9 units per week. When a demand occurs, it is filled with on-hand inventory. Any unsatisfied demand is lost, while the holding of any unsold products costs one dollar per unit per week. The store owner wants to satisfy at least 95% of demand during a week.

Figure 3.1 illustrates this example graphically. A circle is used to represent an SKU-Location, i.e., the product and the location that keeps it. The circle is labeled RW. A suggested labeling scheme is discussed in "Labeling of a Supply Chain". In addition, L stands for the lead time from the external supplier, and h represents the unit holding cost. Similar notations are used throughout this chapter.

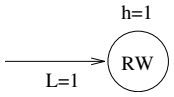

Figure 3.1. Single-Location Problem

PROC MIRP can be used here to determine the optimal replenishment parameters that meet the service level requirement at a minimum inventory holding cost. The procedure requires three input data sets, as follows.

```
data nodedata1;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         leadtime servicelevel holdingcost;
datalines;
N1 RW 1 0.95 1
;

data arcdata1;
   format networkid $2. predecessor successor $8.;
   input networkid $3. predecessor $9. successor $8.;
datalines;
N1 EXTERNAL RW
;

data demanddata1;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N1 RW 1 10 9
N1 RW 2 10 9
N1 RW 3 10 9
N1 RW 4 10 9
N1 RW 5 10 9
N1 RW 6 10 9
N1 RW 7 10 9
```

```
N1 RW 8 10 9
;
```

The nodedata data set contains information associated with an SKU-Location (i.e., node) in a network, such as name, lead time, service level requirement, and unit holding cost. The arcdata data set describes network structures by defining arcs with their predecessors and successors. The demanddata data set holds the forecast of external demand during a planning horizon.

The NetworkID variable, which exists in all three input data sets, specifies the name of a network that an SKU-Location or an arc belongs to. This variable enables users to determine inventory parameters for multiple independent networks with a single call to the procedure.

Note that "EXTERNAL"in the arcdata is a keyword reserved for external suppliers, and it must be **CAPITALIZED**. If an SKU at a location is purchased from an outside supplier, an arc from "EXTERNAL" to the location must be included in the data set.

The following call to PROC MIRP computes optimal inventory levels for product R at store W.

```
title 'Single-Location Problem';
proc mirp nodedata=nodedata1 arcdata=arcdata1
          demanddata=demanddata1 out=out_example1
          horizon=8 lostsales;

   node / networkid=networkid
          skuloc=skuloc leadtime=leadtime
          servicelevel=servicelevel
          holdingcost=holdingcost;

    arc / networkid=networkid
          predecessor=predecessor
          successor=successor;

 demand / networkid=networkid
          skuloc=skuloc period=period
          mean=mean variance=variance;
run;
```

The HORIZON option specifies the number of periods for which the inventory parameters need to be calculated. The LOSTSALES option tells the proc to treat unsatisfied demand as lost sales. This option may be omitted since the default is the LOSTSALES. The NODE, ARC, and DEMAND statements provide the proc with the information of variables in the input data sets. For instance, "LEADTIME=LEADTIME" in the NODE statement names the variable LEADTIME in the node data set that contains the lead time information. Such statement is not required if a default variable name is used. In this example, the variables in all three data sets use their default names. Therefore, PROC MIRP can also be called by the following statements. See "Syntax" for more on the default names.

```
proc mirp nodedata=nodedata1 arcdata=arcdata1
          demanddata=demanddata1 out=out_example1
          horizon=8;
run;
```

Figure 3.2 is the output data set. Variables in the data set are defined in "OUTPUT Statement". Note that the reorder level and the order-up-to level are given for each period in the planning horizon. To replenish the inventory, the store needs to calculate its inventory position, which is the sum of on-hand inventory and in-transit inventory (i.e., shipments yet to be received). If it is below the reorder level, the store should place an order so that its inventory position is raised up to the order up-to level. See "Determination of Replenishment Quantity" for more information.

Single-Location Problem

Obs	Network ID	Sku Loc	Target Service	Service Type	Policy	Unmet Demand	Period	Reorder Level	Order Up To Level	_STATUS_
1	N1	RW	0.95	FR	BS	LostSales	1	22	23	SUCCESSFUL
2	N1	RW	0.95	FR	BS	LostSales	2	22	23	SUCCESSFUL
3	N1	RW	0.95	FR	BS	LostSales	3	22	23	SUCCESSFUL
4	N1	RW	0.95	FR	BS	LostSales	4	22	23	SUCCESSFUL
5	N1	RW	0.95	FR	BS	LostSales	5	22	23	SUCCESSFUL
6	N1	RW	0.95	FR	BS	LostSales	6	22	23	SUCCESSFUL
7	N1	RW	0.95	FR	BS	LostSales	7	22	23	SUCCESSFUL
8	N1	RW	0.95	FR	BS	LostSales	8	22	23	SUCCESSFUL

Figure 3.2. Output Data Set of Single-Echelon Network

Two-Echelon Distribution Network

Consider a distribution network with one warehouse and two retail locations. A finished good F is stored at the warehouse W with a lead time of three weeks from an external supplier and a unit holding cost of \$16 per week. The finished good is shipped to two retailers, C and D. The unit holding costs at C and D are \$18 per week. The lead time between W and C is three weeks, and between W and D is four weeks. The minimum required service levels are 95% for both C and D, and 85% for W. Figure 3.3 illustrates this distribution network with notations similar to "Single Location". In this example, the base period is defined as one week.

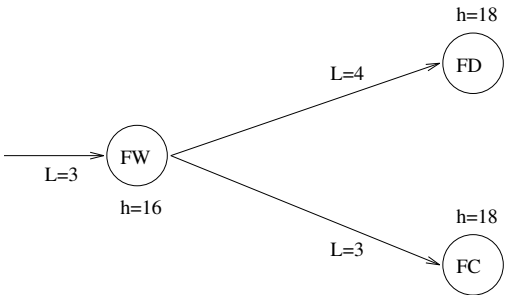

Figure 3.3. Two-Echelon Distribution Network

There are three nodes (i.e., circles) in the network. Their labels are the concatenation of the names of the product and the location where the product is stored. See "Labeling of a Supply Chain" for more information. PROC MIRP can be used to compute inventory parameters that minimize the inventory cost for the entire network while meeting the service level constraints at all locations.

To model and solve the inventory replenishment problem in this two-echelon distribution network, the following input data sets are created. The arc data set describes the network structure by specifying the predecessor and successor of each arc in the network.

```
data nodedata2;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3. description $15.
         lt sl hc;
datalines;
N2 FW warehouse        3 0.85 16
N2 FC retailer 1       3 0.95 18
N2 FD retailer 2       4 0.95 18
;

data arcdata2;
   format networkid $2. head tail $8.;
   input networkid $3. head $9. tail $3. qty;
datalines;
N2 EXTERNAL FW 1
N2 FW       FC 1
N2 FW       FD 1
;
```

```
data demanddata2;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N2 FC 1 20 25
N2 FC 2 15 16
N2 FC 3 10 9
N2 FC 4 10 9
N2 FC 5 15 16
N2 FC 6 15 16
N2 FC 7 20 25
N2 FC 8 20 25
N2 FD 1 10 9
N2 FD 2 10 9
N2 FD 3 15 16
N2 FD 4 15 16
N2 FD 5 20 25
N2 FD 6 20 25
N2 FD 7 15 16
N2 FD 8 15 16
;
```

The following statements call PROC MIRP to compute the inventory policy parameters at each location for each period in the planning horizon.

```
title 'Two-Echelon Distribution Network';
proc mirp nodedata=nodedata2 arcdata=arcdata2
          demanddata=demanddata2 out=out_example2
          horizon=8 lostsales;

   node / networkid=networkid
          skuloc=skuloc leadtime=lt
          servicelevel=sl holdingcost=hc;

    arc / networkid=networkid
          predecessor=head successor=tail
          quantity=qty;

 demand / networkid=networkid
          skuloc=skuloc period=period
          mean=mean variance=variance;
run;
```

Since variables in the input data sets are using their default names, a simpler statement can also be used. See "Syntax" for more on the default names.

```
proc mirp nodedata=nodedata2 arcdata=arcdata2
          demanddata=demanddata2 out=out_example2
          horizon=8;
run;
```

In the output data set (Figure 3.4), there are reorder levels and order up-to levels for every SKU-location and every period. Note that the demand at the store level has shifts from week to week; therefore, these control parameters change from week to week.

Two-Echelon Distribution Network

Obs	NetworkID	SkuLoc	Description	TargetService	ServiceTypeFR	PolicyBS	UnmetDemand	Period	ReorderLevel	OrderUpToLevel	_STATUS_
1	N2	FC	retailer 1	0.95	FR	BS	LostSales	1	54	55	SUCCESSFUL
2	N2	FC	retailer 1	0.95	FR	BS	LostSales	2	49	50	SUCCESSFUL
3	N2	FC	retailer 1	0.95	FR	BS	LostSales	3	49	50	SUCCESSFUL
4	N2	FC	retailer 1	0.95	FR	BS	LostSales	4	60	61	SUCCESSFUL
5	N2	FC	retailer 1	0.95	FR	BS	LostSales	5	70	71	SUCCESSFUL
6	N2	FC	retailer 1	0.95	FR	BS	LostSales	6	75	76	SUCCESSFUL
7	N2	FC	retailer 1	0.95	FR	BS	LostSales	7	81	82	SUCCESSFUL
8	N2	FC	retailer 1	0.95	FR	BS	LostSales	8	81	82	SUCCESSFUL
9	N2	FD	retailer 2	0.95	FR	BS	LostSales	1	69	70	SUCCESSFUL
10	N2	FD	retailer 2	0.95	FR	BS	LostSales	2	79	80	SUCCESSFUL
11	N2	FD	retailer 2	0.95	FR	BS	LostSales	3	84	85	SUCCESSFUL
12	N2	FD	retailer 2	0.95	FR	BS	LostSales	4	84	85	SUCCESSFUL
13	N2	FD	retailer 2	0.95	FR	BS	LostSales	5	84	85	SUCCESSFUL
14	N2	FD	retailer 2	0.95	FR	BS	LostSales	6	79	80	SUCCESSFUL
15	N2	FD	retailer 2	0.95	FR	BS	LostSales	7	74	75	SUCCESSFUL
16	N2	FD	retailer 2	0.95	FR	BS	LostSales	8	74	75	SUCCESSFUL
17	N2	FW	warehouse	0.85	FR	BS	LostSales	1	124	125	SUCCESSFUL
18	N2	FW	warehouse	0.85	FR	BS	LostSales	2	124	125	SUCCESSFUL
19	N2	FW	warehouse	0.85	FR	BS	LostSales	3	129	130	SUCCESSFUL
20	N2	FW	warehouse	0.85	FR	BS	LostSales	4	129	130	SUCCESSFUL
21	N2	FW	warehouse	0.85	FR	BS	LostSales	5	129	130	SUCCESSFUL
22	N2	FW	warehouse	0.85	FR	BS	LostSales	6	129	130	SUCCESSFUL
23	N2	FW	warehouse	0.85	FR	BS	LostSales	7	129	130	SUCCESSFUL
24	N2	FW	warehouse	0.85	FR	BS	LostSales	8	129	130	SUCCESSFUL

Figure 3.4. Output Data Set of Two-Echelon Distribution Network

Two-Echelon Assembly Network

Consider a manufacturing process in a plant P. Two components, A and B, are assembled into a finished good, F. Both components are purchased from external suppliers. The lead times are one period and two periods for A and B, respectively. The unit holding cost is \$1 per period for A to be stored at the plant, and \$2 for B. It takes three units of A and two units of B to produce one unit of F. The assembly process takes three periods. It costs the plant \$13 per period to store one unit of F. The minimum service level required is 95% for F and 50% for both A and B. The finished good F faces customer demand with a mean of 16 units and a variance of 36 units per period in the next eight periods. This assembly process is illustrated in Figure 3.5.

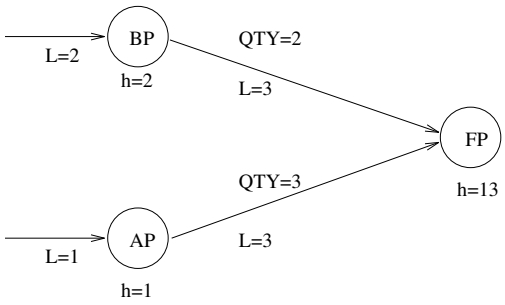

Figure 3.5. Two-Echelon Assembly Network

Similar to the previous examples, the labels of the nodes in the network are the concatenation of the names of the SKUs and the locations. See "Labeling of a Supply Chain" for more information.

The total holding cost of the two components that go into the finished goods can be computed as follows.

$$\begin{aligned} &\text{unit holding cost of } A \times \text{Quantity of } A \text{ into } F \\ &+ \text{unit holding cost of } B \times \text{Quantity of } B \text{ into } F \\ &= 1 \times 3 + 2 \times 2 = 7. \end{aligned}$$

Since the unit holding cost of F is \$13, a significant value is added during the assembly process. It also implies that it is much more expensive to hold the finished goods than the components.

```
data nodedata3;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3. description $15.
         lt sl hc;
datalines;
N3 AP component 1     1 0.5  1
N3 BP component 2     2 0.5  2
N3 FP finished goods  3 0.95 13
;

data arcdata3;
```

```
   format networkid $2. head tail $8.;
   input networkid $3. head $9. tail $3. qty;
datalines;
N3 EXTERNAL AP 1
N3 EXTERNAL BP 1
N3 AP       FP 3
N3 BP       FP 2
;

data demanddata3;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N3 FP 1 16 36
N3 FP 2 16 36
N3 FP 3 16 36
N3 FP 4 16 36
N3 FP 5 16 36
N3 FP 6 16 36
N3 FP 7 16 36
N3 FP 8 16 36
;


title 'Two-Echelon Assembly Network';
proc mirp nodedata=nodedata3 arcdata=arcdata3
          demanddata=demanddata3 out=out_example3
          horizon=8 lostsales;

   node / networkid=networkid
          skuloc=skuloc leadtime=lt
          servicelevel=sl holdingcost=hc;

    arc / networkid=networkid
          predecessor=head successor=tail quantity=qty;

 demand / networkid=networkid
          skuloc=skuloc period=period
          mean=mean variance=variance;
run;
```

Note that in the arcdata data set, the bill of material relationship (i.e., Quantity) is assigned between the components and the finished good. The output data set is shown in Figure 3.6.

Two-Echelon Assembly Network

Obs	NetworkID	SkuLoc	Description	TargetService	SericeType	Policy	UnmetDemand	Period	RecordLevel	OrderUpToLevel	_STATUS_
1	N3	FP	finished goods	0.95	FR	BS	LostSales	1	69	70	SUCCESSFUL
2	N3	FP	finished goods	0.95	FR	BS	LostSales	2	69	70	SUCCESSFUL
3	N3	FP	finished goods	0.95	FR	BS	LostSales	3	69	70	SUCCESSFUL
4	N3	FP	finished goods	0.95	FR	BS	LostSales	4	69	70	SUCCESSFUL
5	N3	FP	finished goods	0.95	FR	BS	LostSales	5	69	70	SUCCESSFUL
6	N3	FP	finished goods	0.95	FR	BS	LostSales	6	69	70	SUCCESSFUL
7	N3	FP	finished goods	0.95	FR	BS	LostSales	7	69	70	SUCCESSFUL
8	N3	FP	finished goods	0.95	FR	BS	LostSales	8	69	70	SUCCESSFUL
9	N3	AP	component 1	0.50	FR	BS	LostSales	1	113	114	SUCCESSFUL
10	N3	AP	component 1	0.50	FR	BS	LostSales	2	113	114	SUCCESSFUL
11	N3	AP	component 1	0.50	FR	BS	LostSales	3	113	114	SUCCESSFUL
12	N3	AP	component 1	0.50	FR	BS	LostSales	4	113	114	SUCCESSFUL
13	N3	AP	component 1	0.50	FR	BS	LostSales	5	113	114	SUCCESSFUL
14	N3	AP	component 1	0.50	FR	BS	LostSales	6	113	114	SUCCESSFUL
15	N3	AP	component 1	0.50	FR	BS	LostSales	7	113	114	SUCCESSFUL
16	N3	AP	component 1	0.50	FR	BS	LostSales	8	113	114	SUCCESSFUL
17	N3	BP	component 2	0.50	FR	BS	LostSales	1	105	106	SUCCESSFUL
18	N3	BP	component 2	0.50	FR	BS	LostSales	2	105	106	SUCCESSFUL
19	N3	BP	component 2	0.50	FR	BS	LostSales	3	105	106	SUCCESSFUL
20	N3	BP	component 2	0.50	FR	BS	LostSales	4	105	106	SUCCESSFUL
21	N3	BP	component 2	0.50	FR	BS	LostSales	5	105	106	SUCCESSFUL
22	N3	BP	component 2	0.50	FR	BS	LostSales	6	105	106	SUCCESSFUL
23	N3	BP	component 2	0.50	FR	BS	LostSales	7	105	106	SUCCESSFUL
24	N3	BP	component 2	0.50	FR	BS	LostSales	8	105	106	SUCCESSFUL

Figure 3.6. Output Data Set of Two-Echelon Assembly Network

Multiple Networks within a Single Call

All previous examples can be solved in a single call to PROC MIRP by combining their input data. In the following statements, the input data sets of the distribution and assembly networks are combined. Figure 3.7 gives the partial output data set for both networks.

```
data nodedata4;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3. description $15.
         lt sl hc;
datalines;
N2 FW warehouse       3 0.85 16
N2 FC retailer 1      3 0.95 18
N2 FD retailer 2      4 0.95 18
N3 AP component 1     1 0.5  1
N3 BP component 2     2 0.5  2
N3 FP finished goods  3 0.95 13
;

data arcdata4;
   format networkid $2. predecessor successor $8.;
   input networkid $3. predecessor $9. successor $3.
         qty;
datalines;
N2 EXTERNAL FW 1
N2 FW       FC 1
N2 FW       FD 1
N3 EXTERNAL AP 1
N3 EXTERNAL BP 1
N3 AP       FP 3
N3 BP       FP 2
;

data demanddata4;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N2 FC 1 20 25
N2 FC 2 15 16
N2 FC 3 10 9
N2 FC 4 10 9
N2 FC 5 15 16
N2 FC 6 15 16
N2 FC 7 20 25
N2 FC 8 20 25
N2 FD 1 10 9
N2 FD 2 10 9
N2 FD 3 15 16
N2 FD 4 15 16
N2 FD 5 20 25
N2 FD 6 20 25
```

```
N2 FD 7 15 16
N2 FD 8 15 16
N3 FP 1 16 36
N3 FP 2 16 36
N3 FP 3 16 36
N3 FP 4 16 36
N3 FP 5 16 36
N3 FP 6 16 36
N3 FP 7 16 36
N3 FP 8 16 36
;


title 'Multiple Networks Within a Single Call';
proc mirp nodedata=nodedata4 arcdata=arcdata4
          demanddata=demanddata4 out=out_example4
          horizon=8;
run;
```

Note that distinct network IDs are assigned to each network. Since PROC MIRP computes inventory control parameters for one network at a time, SKU-locations and arcs in the data sets must be grouped by their corresponding networks. In addition, the sequence of networks in the input data sets must be the same. In this example, networks are arranged in the order of N2 and N3 for all input data sets.

Multiple Networks Within a Single Call
(Partial Output)

Obs	NetworkID	Sku Loc	Description	Target Service Level	Service Type	Policy	UnmetDemand	Period	Record Level	Order Up To Level	_STATUS_
1	N2	FC	retailer 1	0.95	FR	BS	LostSales	1	54	55	SUCCESSFUL
2	N2	FC	retailer 1	0.95	FR	BS	LostSales	2	49	50	SUCCESSFUL
3	N2	FC	retailer 1	0.95	FR	BS	LostSales	3	49	50	SUCCESSFUL
4	N2	FC	retailer 1	0.95	FR	BS	LostSales	4	60	61	SUCCESSFUL
5	N2	FC	retailer 1	0.95	FR	BS	LostSales	5	70	71	SUCCESSFUL
6	N2	FC	retailer 1	0.95	FR	BS	LostSales	6	75	76	SUCCESSFUL
7	N2	FC	retailer 1	0.95	FR	BS	LostSales	7	81	82	SUCCESSFUL
8	N2	FC	retailer 1	0.95	FR	BS	LostSales	8	81	82	SUCCESSFUL
9	N2	FD	retailer 2	0.95	FR	BS	LostSales	1	69	70	SUCCESSFUL
10	N2	FD	retailer 2	0.95	FR	BS	LostSales	2	79	80	SUCCESSFUL
11	N2	FD	retailer 2	0.95	FR	BS	LostSales	3	84	85	SUCCESSFUL
12	N2	FD	retailer 2	0.95	FR	BS	LostSales	4	84	85	SUCCESSFUL
13	N2	FD	retailer 2	0.95	FR	BS	LostSales	5	84	85	SUCCESSFUL
14	N2	FD	retailer 2	0.95	FR	BS	LostSales	6	79	80	SUCCESSFUL
15	N2	FD	retailer 2	0.95	FR	BS	LostSales	7	74	75	SUCCESSFUL
16	N2	FD	retailer 2	0.95	FR	BS	LostSales	8	74	75	SUCCESSFUL
17	N2	FW	warehouse	0.85	FR	BS	LostSales	1	124	125	SUCCESSFUL
18	N2	FW	warehouse	0.85	FR	BS	LostSales	2	124	125	SUCCESSFUL
19	N2	FW	warehouse	0.85	FR	BS	LostSales	3	129	130	SUCCESSFUL
20	N2	FW	warehouse	0.85	FR	BS	LostSales	4	129	130	SUCCESSFUL
21	N2	FW	warehouse	0.85	FR	BS	LostSales	5	129	130	SUCCESSFUL
22	N2	FW	warehouse	0.85	FR	BS	LostSales	6	129	130	SUCCESSFUL
23	N2	FW	warehouse	0.85	FR	BS	LostSales	7	129	130	SUCCESSFUL
24	N2	FW	warehouse	0.85	FR	BS	LostSales	8	129	130	SUCCESSFUL
25	N3	FP	finished goods	0.95	FR	BS	LostSales	1	69	70	SUCCESSFUL
26	N3	FP	finished goods	0.95	FR	BS	LostSales	2	69	70	SUCCESSFUL
27	N3	FP	finished goods	0.95	FR	BS	LostSales	3	69	70	SUCCESSFUL
28	N3	FP	finished goods	0.95	FR	BS	LostSales	4	69	70	SUCCESSFUL
29	N3	FP	finished goods	0.95	FR	BS	LostSales	5	69	70	SUCCESSFUL
30	N3	FP	finished goods	0.95	FR	BS	LostSales	6	69	70	SUCCESSFUL
31	N3	FP	finished goods	0.95	FR	BS	LostSales	7	69	70	SUCCESSFUL
32	N3	FP	finished goods	0.95	FR	BS	LostSales	8	69	70	SUCCESSFUL

Figure 3.7. Partial Output Data Set for Two Networks

Syntax

The following statements are used in PROC MIRP.

PROC MIRP *<option(s)>* **;**
NODE */ <option(s)>* **;**
ARC */ <option(s)>* **;**
DEMAND */ <option(s)>* **;**
INVENTORY */ <option(s)>* **;**
OUTPUT */ <option(s)>* **;**

PROC MIRP Statement

PROC MIRP *<option(s)>* **;**

The following options can appear on the PROC MIRP statement. The NodeData=, ArcData=, and DemandData= are required, while others are optional.

NodeData=*SAS-data-set*
names the SAS data set with variables that describe each SKU-location in the network(s) under consideration. These variables are defined in "NODE Statement". The NodeData= data set is required and the data must be grouped by the NetworkID= variable.

ArcData=*SAS-data-set*
names the SAS data set with variables that describe each arc in the network(s) under consideration. These variables are defined in "ARC Statement". The ArcData= data set is required and the data must be grouped by the NetworkID= variable.

DemandData=*SAS-data-set*
names the SAS data set with variables that describe external demand at each SKU-location in the network(s) under consideration. These variables are defined in "DEMAND Statement". The DemandData= input data set is required and the data must be grouped by the NetworkID= variable.

InventoryData=*SAS-data-set*
names the SAS data set with variables that describe inventory status at the beginning of the planning horizon at each SKU-location in the network(s) under consideration. These variables are defined in "INVENTORY Statement". The InventoryData= data set is optional. If it is specified, the data must be grouped by the NetworkID= variable.

Out=*SAS-data-set*
names the output data set that contains computation results for all SKU-Locations in all networks. If the Out= is omitted, the SAS System creates a data set and names it according to the DATA*n* naming convention. The variables in this data set are defined in "OUTPUT Statement". The data is grouped by the NetworkID= variable.

Out1=*SAS-data-set*
names the output data set with computation results that are time independent. See

"OUTPUT Statement" for the time-independent results. The procedure does not create this data set unless it is specified.

Out2=*SAS-data-set*
names the output data set with computation results that are time dependent. See "OUTPUT Statement" for the time-dependent results. The procedure does not create this data set unless it is specified.

MaxMessages=*maxmessages*
specifies the maximum number of warning and error messages that the procedure displays. If the MaxMessages= option is omitted, the procedure displays all messages.

NetworkCnt=*networkcnt*
specifies the maximum number of networks that the procedure analyzes. If the NetworkCnt= option is omitted, the procedure processes all networks in input data sets. If the input data sets contain more networks than what is specified by NetworkCnt= option, the extras are ignored.

When PROC MIRP is used to process multiple networks, the data of the networks must be grouped in the same sequence in every input data sets.

Horizon=*horizon*
specifies the number of periods for which the inventory policy parameters are to be computed. The default value is 12.

Replications=*replications*
specifies the number of simulation replications to be used in the policy calculation and the performance evaluation. The default value is 200.

Timeout=*timeout*
specifies the maximum running time in milliseconds to be used per node per period per replication. The default value is 10. Since the multi-echelon inventory optimization problem is nonlinear, there will be cases where the computation results do not converge. Therefore, this option is necessary. Users can also use this option to control the running time of the procedure.

LostSales | Backlogs
specifies whether unsatisfied demands are treated as lost sales or backlogs.

The LostSales and Backlogs options are two distinct ways to handle unsatisfied demands, and are mutually exclusive. If both of them appear in the statement, the procedure gives a warning message and uses the LostSales option. If neither of them is specified, the default option is LostSales.

If the LostSales option is used, any unsatisfied demand is lost. If the Backlogs option is used, any unsatisfied demand is backlogged and fulfilled by future delivery.

Optimization | Evaluation
specifies whether the procedure is used to compute inventory policy parameters for given inputs or to evaluate inventory performance for given policy parameters.

The Optimization and Evaluation options are two different ways to use the procedure, and are mutually exclusive. If both of them appear in the statement, the procedure

gives a warning message and uses the Optimization option. If neither of them is specified, the default option is Optimization.

NODE Statement

NODE */<option(s)>* **;**

The NODE statement specifies variables in the NodeData= data set. If a variable is not specified, the procedure searches the variable with its default names.

Table 3.1. Node Statement Summary

Variable	Required	Data Type	Data Value	Missing Value	Default Name(s)	Default Value
NetworkID	Yes	Character		not allowed	NetworkID, NetID	
SkuLoc	Yes	Character	"EXTERNAL" not allowed	not allowed	SkuLoc, NodeID, ItemID	
Description	No	Character		allowed	Description, Desc	
LeadTimeMin	No	Numeric	nonnegative integers	allowed	LeadTimeMin, LTMin	zero
LeadTime	Yes	Numeric	nonnegative integers	allowed	LeadTime, LT	zero
LeadTimeMax	No	Numeric	nonnegative integers	allowed	LeadTimeMax, LTMax	zero
PolicyType	No	Character	{BS, SS}	allowed	PolicyType, Policy	BS
ServiceLevel	Yes	Numeric	$[0, 0.999]$	allowed	ServiceLevel, SL	0.5
ServiceType	No	Character	RR,FR,BR,WT	allowed	ServiceType, SlType	FR
HoldingCost	No	Numeric	nonnegative	allowed	HoldingCost, HC	0
FixedCost	No	Numeric	nonnegative	allowed	FixedCost, FC	0
PBR	No	Numeric	positive integers	allowed	PBR	one
BatchSize	No	Numeric	positive	allowed	BatchSize, LotSize	zero
OrderMin	No	Numeric	nonnegative	allowed	OrderMin, MinSize	one
OrderMax	No	Numeric	nonnegative	allowed	OrderMax, MaxSize	maximum double
NextReplenish	No	Numeric	nonnegative integers	allowed	NextReplenish, NRP	zero
WaitTime	No	Numeric	nonnegative integers	allowed	WaitTime, WaitingTime	zero

Table 3.1. Node Statement Summary (continued)

Variable	Required	Data Type	Data Value	Missing Value	Default Name(s)	Default Value
DemandInterval	No	Numeric	≥ 1	allowed	DemandInterval, DIT	one

NetworkID=*variable*
identifies the variable that contains the network ID for each SKU-location in the NodeData= data set. This variable enables users to compute policy parameters for multiple networks with a single call to the procedure, which is illustrated in "Multiple Networks within a Single Call".

SkuLoc=*variable*
identifies the variable in the NodeData= data set that contains the ID for each SKU-location. Users can use any form of IDs as long as the duplicates are avoided. A labeling scheme is suggested in "Labeling of a Supply Chain".

Description=*variable*
identifies the variable in the NodeData= data set that contains the description for each SKU-location.

LeadTime=*variable*
identifies the variable in the NodeData= data set that contains the average lead time for each SKU-location.

LeadTimeMin=*variable*
identifies the variable in the NodeData= data set that contains the minimum lead time for each SKU-location.

LeadTimeMax=*variable*
identifies the variable in the NodeData= data set that contains the max lead time for each SKU-location.

Lead times are integral multiples of the base period. In practice, there are two major components of lead time: fulfillment-related lead time and transit-related lead time. When the on-hand inventory at a location is insufficient to satisfy orders from downstream, a backlog occurs and only a fraction of orders are shipped immediately. The remaining portion is filled in some future period, which is likely to be random. This is called *fulfillment-related* lead time. The time to ship an order (full or partial) from one location to another is called *transit-related* lead time, which could be random as well but likely has less variation than the fulfillment-related lead time does.

When the lead time data is collected and calculated, only the transit-related lead time should be considered as the input into the proc, while the fulfillment-related lead time is taken into account implicitly by the algorithm (more specifically, by incorporating backlogs into the calculation of control parameters).

The LeadTime variable is required, while LeadTimeMin and LeadTimeMax are optional. If LeadTimeMin is specified, LeadTimeMax must be specified as well, and vice versa.

- When LeadTimeMin<LeadTime<LeadTimeMax, the triangular distribution is used to model the uncertainty in the lead time.
- When LeadTimeMin=LeadTime<LeadTimeMax, or LeadTimeMin<LeadTime=LeadTimeMax, the uniform distribution between LeadTimeMin and LeadTimeMax is used to model the uncertainty in the lead time.
- When LeadTimeMin=LeadTime=LeadTimeMax, the lead time is considered constant.
- When only LeadTime is specified, the lead time is considered constant.

PolicyType=*variable*
identifies the variable in the NodeData= data set that contains the policy type to be used at each SKU-location. Currently the procedure supports two policy types: BS and SS. BS stands for the base-stock policy, and SS for the min-max policy.

The base-stock policy is also called one-to-one replenishment policy, because the order up-to level is equal to the reorder level plus one. This policy is recommended when the fixed ordering cost is insignificant compared to the cost of inventory being replenished.

The min-max policy is recommended when the fixed ordering cost is significant compared with the cost of inventory replenished. In this case, a large amount of inventory should be ordered in order to make replenishment orders less frequent. The order up-to level in this policy is greater than the reorder level by at least one. When the difference is exactly one, this policy is reduced to the base-stock policy.

ServiceLevel=*variable*
identifies the variable in the NodeData= data set that contains the service level requirement for each SKU-location.

ServiceType=*variable*
identifies the variable in the NodeData= data set that contains the type of the service level used for each SKU-location. The procedure currently supports four service types: RR, FR, BR, and WT. Specifically, RR stands for the ready rate (or non-stockout probability), FR for the fill rate, BR for the backorder ratio, and WT for waiting time probability.

The ready rate is also called the non-stockout probability. It is the probability of having no stockout (i.e., positive on-hand inventory) at the end of a period. It can be measured as the ratio between the number of periods with positive on-hand inventory and the total number of periods in a planning horizon.

The fill rate is average demand being satisfied immediately by on-hand inventory.

The backorder ratio is the ratio between the average backlog at the end of a period and the average demand that occurred during the period.

The waiting time probability is the probability of satisfying an order within a pre-specified time window. This is used when an order can be fulfilled at a later time than the time it is issued.

HoldingCost=*variable*
identifies the variable in the NodeData= data set that contains the unit holding cost per period for each SKU-location.

FixedCost=*variable*
identifies the variable in the NodeData= data set that contains the fixed ordering cost for each SKU-location. The fixed order cost is the amount of cost incurred every time a purchasing order is placed. It is independent of the amount of the order being placed.

PBR=*variable*
identifies the variable in the NodeData= data set that contains the number of periods between two replenishment orders for each SKU-location.

When the PBR is used together with the base-stock policy, replenishment orders can only be placed once every PBR periods. For example, PBR=4 with the base-stock policy means that replenishment orders can only be issued once every four periods.

When the PBR is used together with the min-max policy, replenishment orders can be placed every period. However, the policy parameters are such that on average there are PBR periods between two replenishment orders. Note that the PBR depends upon policy parameters as well as other inputs, so the actual PBR could be different from the one specified here. The procedure ensures that they are as close as possible.

BatchSize=*variable*
identifies the variable in the NodeData= data set that contains the batch size constraint on the amount of orders that can be placed for each SKU-location.

OrderMin=*variable*
identifies the variable in the NodeData= data set that contains the minimum order amount to be placed at each SKU-location.

OrderMax=*variable*
identifies the variable in the NodeData= data set that contains the maximum order amount to be placed at each SKU-location.

NextReplenish=*variable*
identifies the variable in the NodeData= data set that contains the number of periods until the first replenishment order can be made for each SKU-location.

For example, a value of 0 for this variable means that the first replenishment order can be placed at period 1, while a value of 4 specifies that no replenishment order can be placed until period 5. This variable is used in conjunction with the variable PBR under the base-stock policy only, and is ignored under the min-max policy.

WaitTime=*variable*
identifies the variable in the NodeData= data set that contains the number of periods between the time an order is placed and the time it can be filled. This variable is used in conjunction with ServiceType=WT and Backlogs option. It is set to zero otherwise.

For example, suppose that WaitTime=3, ServiceLevel=0.99, and ServiceType=WT for an SKU-location. PROC MIRP computes policy parameters such that demand at the SKU-location is satisfied within three periods with 99% probability.

DemandInterval=*variable*
identifies the variable in the NodeData= data set that contains the number of periods between two positive demands. This variable is used to model intermittent demand. When this variable is specified, the demand mean and variance provided in the DemandData= data set must be the mean and variance of **positive** demand that might occur in a period.

ARC Statement

ARC */<option(s)>* **;**

The ARC statement identifies the variables contained in the ArcData= data set. If a variable is not specified, the procedure searches the variable with its default names.

Table 3.2. Arc Statement Summary

Variable	Required	Data Type	Data Value	Missing Value	Default Name(s)	Default Value
NetworkID	Yes	Character		not allowed	NetworkID, NetID	
Predecessor	Yes	Character		not allowed	Predecessor, Head	
Successor	Yes	Character	"EXTERNAL" not allowed	not allowed	Successor, Tail	
Quantity	No	Numeric	positive	allowed	Quantity, Qty	one
PipelineCost	No	Numeric	nonnegative	allowed	PipelineCost, PsCost	zero

NetworkID=*variable*
identifies the variable in the ArcData= data set that contains the network ID for each arc to be analyzed.

Predecessor=*variable*
identifies the variable in the ArcData= data set that contains the SkuLoc of the predecessor of each arc.

If an arc links an SKU-location with an external supplier, then Predecessor should be set to "EXTERNAL". Note that "EXTERNAL" is a reserved word and can only be used in this situation.

Successor=*variable*
identifies the variable in the ArcData= data set that contains the SkuLoc of the successor of each arc.

Quantity=*variable*
identifies the variable in the ArcData= data set that contains the bill of material (BOM) quantity between the predecessor and the successor of each arc. It is the number of units at the predecessor required to produce one unit at the successor.

PipelineCost=*variable*
identifies the variable in the ArcData= data set that contains the unit cost per period

of inventory in transit from the predecessor to the successor.

DEMAND Statement

DEMAND */<option(s)>* **;**

The DEMAND statement identifies the variables contained in the DemandData= data set. If a variable is not specified, the procedure searches the variable with its default names.

Table 3.3. Demand Statement Summary

Variable	Required	Data Type	Data Value	Missing Value	Default Name(s)	Default Value
NetworkID	Yes	Character		not allowed	NetworkID, NetID	
SkuLoc	Yes	Character	"EXTERNAL" not allowed	not allowed	SkuLoc, NodeID, ItemID	
Period	No	Numeric	integers $\geq$ one	not allowed	Period, Time	one
PeriodDesc	No	Numeric			PeriodDesc	
Mean	Yes	Numeric	nonnegative	not allowed	Mean, Average	
Variance	no	Numeric	nonnegative	not allowed	Variance, Var	

NetworkID=*variable*
identifies the variable in the DemandData= data set that contains the NetworkID for each SKU-location to be analyzed.

SkuLoc=*variable*
identifies the variable in the DemandData= data set that contains the ID for SKU-location to be analyzed.

Period=*variable*
identifies the variable in the DemandData= data set that contains the time period of demand of each SKU-location to be analyzed.

PeriodDesc=*variable*
identifies the variable in the DemandData= data set that contains the description of the time period of demand at each SKU-location to be analyzed. This variable may be used to contain the actual date information for each period.

Mean=*variable*
identifies the variable in the DemandData= data set that contains the mean of the demand of each SKU-location.

Variance=*variable*
identifies the variable in the DemandData= data set that contains the variance of demand of each SKU-location.

Note that mean and variance can be zero. However, zero mean and positive variance are not allowed.

When demand is intermittent, the mean and variance of **positive** demand should be provided here. The demand interval should be provided in the NodeData data set.

INVENTORY Statement

INVENTORY */ inventory options* **;**

The INVENTORY statement identifies variables contained in the InventoryData= data set. If a variable is not specified, the procedure searches the variable with its default names.

Table 3.4. Inventory Statement Summary

Variable	Required	Data Type	Data Value	Missing Value	Default Name(s)	Default Value
NetworkID	Yes	Character		not allowed	NetworkID, NetID	
SkuLoc	Yes	Character	"EXTERNAL" not allowed	not allowed	SkuLoc, NodeID, ItemID	
Period	No	Numeric	integers $\geq$ one	not allowed	Period, Time	one
Amount	No	Numeric	nonnegative	allowed	Amount, Amt	zero
ReorderLevel	Yes/No	Numeric		not allowed	ReorderLevel, ROL	
OrderUpToLevel	Yes/No	Numeric		not allowed	OrderUpToLevel, OUTL	

NetworkID=*variable*
identifies the variable in the InventoryData= data set that contains the NetworkID for each SKU-location to be analyzed.

SkuLoc=*variable*
identifies the variable in the InventoryData= data set that contains the ID for each SKU-location to be analyzed.

Period=*variable*
identifies the variable in the InventoryData= data set that contains the arrival time of inventory at a location to be analyzed.

Amount=*variable*
identifies the variable in the InventoryData= data set that contains the amount of inventory to arrive at a location for a specified time period.

ReorderLevel=*variable*
identifies the variable in the InventoryData= data set that contains the reorder level at an SKU-location for a specified time period. This variable is required when the EVALUATION option is used.

OrderUpToLevel=*variable*
identifies the variable in the InventoryData= data set that contains the order up to level at an SKU-location for a specified time period. This variable is required when the EVALUATION option is used.

OUTPUT Statement

OUTPUT */ output options* **;**

The OUTPUT statement identifies variables to be included in output data sets. By default, the output data sets include NetworkID, SkuLoc, Description if available, TargetService, ServiceType, PolicyType, UnmetDemand, Period, ReorderLevel, OrderUpToLevel, and Status. Users can specify other variables in this statement to be included in the output data sets.

NetworkID
This variable contains the ID of a network that an SKU-location belongs to. The value is the same as the one in the NodeData= data set.

SkuLoc
This variable contains the ID of an SKU-location. The value is the same as the one in the NodeData= data set.

TargetService
This variable contains the target service level of an SKU-location. This is the same as the required service level in the NodeData= data set.

ServiceType
This variable contains the service level type of an SKU-location. This is the same as the one in the NodeData= data set.

PolicyType
This variable contains the policy type of an SKU-location. This is the same as the one in the NodeData= data set.

UnmetDemand
This variable indicates whether the unsatisfied demand is backlogged or lost.

Period
This variable contains the index of each period in the planning horizon.

ReorderLevel
This variable contains the reorder level of an SKU-location for each period in the planning horizon.

The reorder level is used to determine whether a replenishment order should be triggered. When the inventory position at an SKU-location drops below the reorder level, such an order should be placed.

OrderUpToLevel
This variable contains the order up-to level of an SKU-location for each period in the planning horizon.

The order up-to level is used to determine the amount of replenishment order. The order amount is equal to the difference between the inventory position and the order up-to level.

Status

This variable contains the status of the policy calculation for an SKU-location. The status has four types.

- "INVD_VALUE": the procedure fails to compute the policy parameters due to data problems in the input data sets.
- "INFEASIBLE": the procedure stops with an infeasible solution, which means the solution does not satisfy the service level requirements. The likely causes of an infeasible solution are (1) small timeout value, (2) undesirable initial inventory condition.
- "FEASIBLE": the procedure stops with a feasible solution. However, a better solution could be found if more time is given.
- "SUCCESSFUL": the procedure has found the best solution.

ReadyRate

The variable ReadyRate is included in the output. This variable gives the average ready rate of an SKU-location over the planning horizon.

FillRate

The variable FillRate is included in the output. This variable gives the average fill rate of an SKU-location over the planning horizon.

BackorderRatio

The variable BackorderRatio is included in the output. This variable gives the average backorder ratio of an SKU-location over the planning horizon.

LostsalesRatio

The variable LostsalesRatio is included in the output. This variable gives the average lostsales ratio of an SKU-location over the planning horizon.

WT_Prob

The variables WT_Prob and WaitingTime are included in the output. These variables give the probability of meeting demand within a specified waiting time at an SKU-location.

Echelon

The variable Echelon is included in the output. This variable gives the echelon level of an SKU-location within a network. The echelon level of an SKU-location represents its relative position in a network. The echelon level of an SKU-location is equal to the maximum echelon level of all its successor SKU-locations plus one. If an SKU-location does not have successors, its echelon level is one.

PBR

The variable PBR is included in the output. This variable gives the average number of periods between two replenishment orders over the planning horizon. The value is

a function of many input variables, including demand, service level, etc. The value could be different from the one specified in the NodeData= data set.

DemandMean
The variable DemandMean is included in the output. It contains the average demand of an SKU-location at each period in the planning horizon.

DemandVar
The variable DemandVar is included in the output. It contains the variance of demand at an SKU-location for each period in the planning horizon.

BacklogMean
The variable BacklogMean is included in the output. This variable gives the average backlog at an SKU-location for each period in the planning horizon.

BacklogVar
The variable BacklogVar is included in the output. This variable gives the variance of the backlog at an SKU-location for each period in the planning horizon.

LostSalesMean
The variable LostSalesMean is included in the output. This variable gives the average lost sales at an SKU-location for each period in the planning horizon.

LostSalesVar
The variable LostSalesVar is included in the output. This variable gives the variance of the lost sales at an SKU-location for each period in the planning horizon.

ShortfallMean
The variable ShortfallMean is included in the output. This variable gives the average shortfall at an SKU-location for each period in the planning horizon.

ShortfallVar
The variable LostSalesVar is included in the output. This variable gives the variance of the shortfall at an SKU-location for each period in the planning horizon.

OrderMean
The variable OrderMean is included in the output. This variable gives the average order quantity at an SKU-location for each period in the planning horizon.

OrderVar
The variable OrderVar is included in the output. This variable gives the variance of the order quantity at an SKU-location for each period in the planning horizon.

OnHandMean
The variable OnHandMean is included in the output. This variable gives the average on-hand inventory at an SKU-location for each period in the planning horizon.

OnHandVar
The variable OnHandVar is included in the output. This variable gives the variance of the on-hand inventory at an SKU-location for each period in the planning horizon.

Pipeline
The variable Pipeline is included in the output. This variable gives the average pipeline inventory (i.e., inventory in-transit) for each period in the planning horizon.

SafetyStock
The variable SafetyStock is included in the output. This variable gives the average safety stock at an SKU-location for each period in the planning horizon.

RR_Period
The variable RR_Period is included in the output. This variable gives the average of the ready rate at an SKU-location for each period in the planning horizon.

FR_Period
The variable FR_Period is included in the output. This variable gives the average of the fill rate at an SKU-location for each period in the planning horizon.

BR_Period
The variable BR_Period is included in the output. This variable gives the average of the backorder ratio at an SKU-location for each period in the planning horizon.

LR_Period
The variable LR_Period is included in the output. This variable gives the average of the lostsales ratio at an SKU-location for each period in the planning horizon.

OnHandCost
The variable OnHandCost is included in the output. This variable gives the average cost of the on-hand inventory per period over the planning horizon.

PipelineCost
The variable PipelineCost is included in the output. This variable gives the average cost of the pipeline inventory per period over the planning horizon.

OrderCost
The variable OrderCost is included in the output. This variable gives the average of the fixed ordering cost per period over the planning horizon.

AvgCost
The variable AverageCost is included in the output. This variable gives the average cost per period over the planning horizon. The cost includes the inventory holding cost, the pipeline inventory cost, and the fixed ordering cost.

All
If this option is specified, all variables discussed above are included in the output.

Details

Determination of Replenishment Quantity

Given a reorder level and an order up-to level for an SKU-location, the replenishment quantity is computed as follows. First the inventory position is computed. It is the sum of on-hand inventory and pipeline inventory (i.e., inventory in transit). A replenishment order is triggered when the inventory position is below the reorder level. If an order is triggered, the amount of the order is equal to the difference between the order up-to level and the inventory position.

When there are constraints on order quantities, such as minimum size, maximum size, or batch size, then order quantities should be revised accordingly.

Labeling of a Supply Chain

In general, many SKUs are supported by one physical supply chain. One location in the network could hold many different SKUs, while one SKU could be held in many different locations. In order to use PROC MIRP, you must map a physical network to a "virtual" one where nodes are uniquely defined by the pair of (SKU, location). The following two examples demonstrate how this mapping works.

- A product is held in multiple physical locations.

  A product F is distributed from a warehouse W to three locations A, B, and C. In order to determine the inventory control parameters for F at all four locations, four nodes must be defined in the "virtual" supply chain and they have to be uniquely labeled. You are recommended to label them by their (SKU, location) pair, such as FW for F at the warehouse W, FA for F at the location A, etc.

- A physical location holds different products.

  Three components D, E, and F are assembled into a finish good G in a manufacturing plant P. You need to determine the inventory control parameters for the components as well as the finished good. Then a four-node network must be created to model this assembly process, with DP, EP, FP being the labels of the components D, E and F at the plant P, respectively, and with GP as the label for the finished good G at the plant P. Note that four nodes are necessary to model this process although the process occurs at one single location.

Error Processing

Various errors could occur during the running of PROC MIRP, which could be caused by incorrect syntax or problems with input data.

The procedure exits with an error message if any of the following occur:

- a required data set is not specified
- a required data set cannot be found

- a required data set is empty
- a required variable is not found in the data set
- a variable has incorrect data type

If the procedure detects invalid or missing data, it tries to reset it to some default value. If there is no default value, an error message is written to the SAS log file. PROC MIRP then continues to validate other data. PROC MIRP does not calculate policy parameters for networks with data errors.

Examples

Example 3.1. A Multi-Echelon Supply Chain

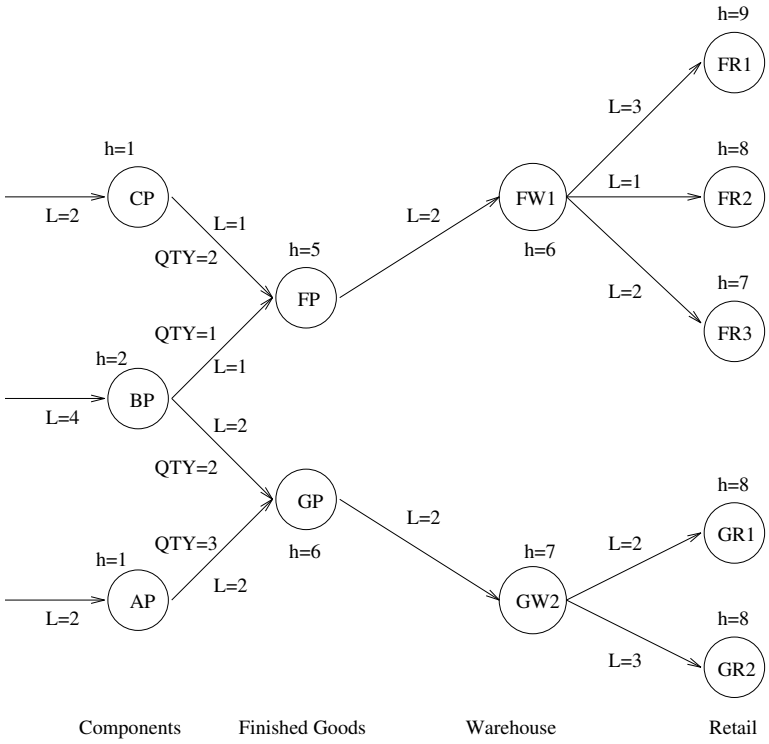

Figure 3.8. A Multi-Echelon Supply Chain Network

Figure 3.8 illustrates a fairly complex example with many features that are supported by PROC MIRP. Three components (A, B, C) are assembled into two finished goods (F, G) at a manufacturing plant (P). These two finished goods are transported to two warehouses (W1, W2) , each of which supports several retail locations (R1, R2, R3).

This is a four-echelon supply chain consisting of assembly and distribution subnetworks. The two finished goods are not independent of each other; they share a common component (B). If there is no common component between them, you can split the supply chain into two independent networks.

```
data nodedata5;
   format networkid servicetype policytype $2. skuloc $3.;
   input networkid $3. skuloc $4. description $16. leadTime
         holdingCost fixedCost serviceLevel serviceType $3.
         policyType $3. pbr;
datalines;
N1 AP  COMPONENT A       2 1 0   0.05 BR BS 1
N1 BP  COMPONENT B       4 2 0   0.05 BR BS 1
N1 CP  COMPONENT C       2 1 0   0.05 BR BS 1
N1 FP  FINISHED GOOD F   1 5 0   0.9  RR BS 1
N1 GP  FINISHED GOOD G   2 6 0   0.9  RR BS 1
```

```
N1 FW1 WAREHOUSE 1       2 6 0   0.9  RR BS 1
N1 GW2 WAREHOUSE 2       2 7 0   0.9  RR BS 1
N1 FR1 F AT RETAILER 1   3 9 450 0.95 FR SS 2
N1 FR2 F AT RETAILER 2   1 8 400 0.95 FR SS 2
N1 FR3 F AT RETAILER 3   2 7 350 0.95 FR SS 2
N1 GR1 G AT RETAILER 1   2 8 400 0.95 FR SS 2
N1 GR2 G AT RETAILER 2   3 8 400 0.95 FR SS 2
;

data arcdata5;
   format networkid $2. predecessor successor $8.;
   input networkid $3. predecessor $9. successor $4.
         quantity;
datalines;
N1 EXTERNAL AP  1
N1 EXTERNAL BP  1
N1 EXTERNAL CP  1
N1 AP       GP  3
N1 BP       GP  2
N1 BP       FP  1
N1 CP       FP  2
N1 GP       GW2 0.95
N1 FP       FW1 0.95
N1 GW2      GR1 1
N1 GW2      GR2 1
N1 FW1      FR1 1
N1 FW1      FR2 1
N1 FW1      FR3 1
;

data demanddata5;
   format networkid $2. skuloc $3.;
   input networkid $3. skuloc $4.
         period mean variance;
datalines;
N1 FW1 1 10  0
N1 GW2 1 6   25
N1 FR1 1 10  90
N1 FR1 2 9   81
N1 FR1 3 8   72
N1 FR1 4 7   63
N1 FR1 5 8   72
N1 FR1 6 9   81
N1 FR1 7 10  90
N1 FR1 8 12  100
N1 FR2 3 0   0
N1 FR2 4 10  225
N1 FR2 5 10  225
N1 FR2 6 10  225
N1 FR2 7 0   0
N1 FR3 1 10  9
N1 FR3 2 9   8
N1 FR3 3 8   7
N1 FR3 4 7   6
N1 FR3 5 8   7
N1 GR1 1 20  25
N1 GR1 2 20  25
N1 GR1 3 20  25
N1 GR1 4 20  25
```

```
N1 GR1 5 20  25
N1 GR1 6 20  25
N1 GR1 7 20  25
N1 GR1 8 20  25
N1 GR2 1 10  9
N1 GR2 2 10  9
N1 GR2 6 10  9
N1 GR2 7 10  9
N1 GR2 8 10  9
;
```

Key aspects of PROC MIRP include the following:

- Three service level types are used. The procedure supports a total of four service level types: RR, FR, BR, WT. You can have different service level types at different SKU-locations within a network.
- Base-stock policy (BS) and min-max policy (SS) are both used. You can choose between these replenishment policies at any SKU-location.
- PBR is specified in the node data set. It is used together with min-max policy to determine the proper policy parameters; for example, PBR=2 for SKU-location GR2. This tells the procedure that, on average, there is one replenishment order for every two periods at GR2. The procedure computes the policy parameters based on this information.
- The bill of material relationship is specified in the arc data set. Note that such a relationship can also be used in places different than the assembly processes; here Qty=0.95 between GP and GW2. You can view it as the "yield" or "loss" due to transportation.
- You can have external demand not only at the lowest echelon level, but also at internal SKU-locations. In this example, two internal SKU-locations (FW1, GW2) face external demand in addition to internal demand from retail locations. This feature can be used to model "direct sales" or the like at an internal SKU-location.
- You can have deterministic demand. SKU-location FW1 faces an external demand with a constant rate of ten units per period.
- When demand forecast is not specified for every period in a planning horizon, the procedure looks for the first period with specified demand, and sets the demand forecast of all previous periods to be equal to that at the first period. The procedure also searches for the last period with a specified demand, and sets the demand forecast of all later periods to be equal to that at the last period. This feature can be used to model several demand patterns.
  - When demand is stationary over the planning horizon, you can simply specify the mean and the variance of demand for one period. See SKU-locations GW2 and GR1.
  - Finished goods F is sold at retail location R2 from period 4 to 6. To model a situation like this, you set its demand forecast to zero at period 3 and 7. This is useful to model a situation where some product only appears at a location for a time window.

– If demand forecast is zero for some periods in the planning horizon, you do not have to specify it. See SKU-location GR2 that has zero demand from period 3 to period 5.

The following statement calls the procedure to compute policy parameters for the above network. Output 3.1.1 is the partial output data set.

```
title 'A Multi-Echelon Supply Chain';
proc mirp nodedata=nodedata5 arcdata=arcdata5
          demanddata=demanddata5 out=out_example5
          horizon=12 lostsales;

   node / networkid=networkid
          skuloc=skuloc leadtime=leadtime
          servicelevel=servicelevel
          holdingcost=holdingcost;

    arc / networkid=networkid
          predecessor=predecessor
          successor=successor
          quantity=quantity;

 demand / networkid=networkid
          skuloc=skuloc period=period
          mean=mean variance=variance;
run;
```

Output 3.1.1. Partial Output Data Set of Multi-Echelon Network

A Multi-Echelon Supply Chain
(Partial Output)

Obs	Network ID	Sku Loc	Description	Target Service Type	Service Level Policy	Policy Demand	Unmet Demand Period	Record Level	Order Up To Level	_STATUS_
1	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	1	24	37	SUCCESSFUL
2	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	2	23	35	SUCCESSFUL
3	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	3	23	34	SUCCESSFUL
4	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	4	23	34	SUCCESSFUL
5	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	5	23	35	SUCCESSFUL
6	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	6	23	35	SUCCESSFUL
7	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	7	23	35	SUCCESSFUL
8	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	8	23	35	SUCCESSFUL
9	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	9	23	35	SUCCESSFUL
10	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	10	23	35	SUCCESSFUL
11	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	11	23	35	SUCCESSFUL
12	N1	FR3	F AT RETAILER 3	0.95	FR SS	LostSales	12	23	35	SUCCESSFUL
13	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	1	16	22	SUCCESSFUL
14	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	2	16	22	SUCCESSFUL
15	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	3	16	22	SUCCESSFUL
16	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	4	23	32	SUCCESSFUL
17	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	5	30	42	SUCCESSFUL
18	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	6	37	52	SUCCESSFUL
19	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	7	37	52	SUCCESSFUL
20	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	8	37	52	SUCCESSFUL
21	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	9	37	52	SUCCESSFUL
22	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	10	37	52	SUCCESSFUL
23	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	11	37	52	SUCCESSFUL
24	N1	GR2	G AT RETAILER 2	0.95	FR SS	LostSales	12	37	52	SUCCESSFUL

Example 3.2. An Independent Solution vs. a Joint Solution

In a multi-echelon supply chain network, users might compute the inventory replenishment policies for each location independently from others in the network, resulting in what is referred to as an "independent solution." This approach is problematic, because it ignores the interactions between predecessor and successor locations in the network. The interactions are two-fold.

1. The predecessor's service level affects the inventory policies at the successor.

   When the service level at the predecessor is less than 100%, fulfillment shortages could occur when the successor places a replenishment order. Therefore, the calculation of the inventory policies at the successor must take these shortages into account.

2. The inventory policies at the successor affect the inventory policies at the predecessor.

   The inventory policies at the successor determine its order quantities, which become the demand at the predecessor. The inventory policies at the predecessor are functions of its demand. Therefore the inventory policies at the successor indirectly impact the inventory policies at the predecessor.

These interactions are very important factors and are taken into account in PROC MIRP. In other words, PROC MIRP solves for all locations in a network jointly, with a "Joint Solution." Ignoring the interaction may lead to a service level performance significantly lower than what is targeted. This is demonstrated by the following example.

The supply chain network considered here is exactly the same as the one in the "Two-Echelon Assembly Network" previously discussed in the "Getting Started" section on page 73. To get an independent solution, three new networks are created, each of which corresponds to a location in the original network. These new networks are described in the following data sets.

```
data nodedata6;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3. description $15.
         lt sl hc;
datalines;
N4 AP component 1     1 0.5  1
N5 BP component 2     2 0.5  2
N6 FP finished goods  3 0.95 13
;

data arcdata6;
   format networkid $2. head tail $8.;
   input networkid $3. head $9. tail $3. qty;
datalines;
N4 EXTERNAL AP 1
N5 EXTERNAL BP 1
N6 EXTERNAL FP 1
;

data demanddata6;
   format networkid skuloc $2.;
   input networkid $3. skuloc $3.
         period mean variance;
datalines;
N4 AP 1 48 324
N5 BP 1 32 144
N6 FP 1 16 36
;
```

There are two differences between the data sets here and those in the "Two-Echelon Assembly Network."

- In the arc data set, all three locations are linked to an external supplier. This is because they are treated independently instead of within a network. Note that the BOM quantities are also changed accordingly.
- In the demand data set, demand at the two components is also provided. Their demands are equal to the demand from the finished good multiplied by the BOM quantities. Such demand population is only valid under a very strict condition: all downstream locations must use the base-stock policy and must have stationary demand. Such a condition rarely exists in real applications.

```
title 'An Independent Solution vs. a Joint Solution';
title2 'Independent Solution';
proc mirp nodedata=nodedata6 arcdata=arcdata6
          demanddata=demanddata6 out=out_example6
          horizon=8 lostsales;

   node / networkid=networkid
          skuloc=skuloc leadtime=lt
          servicelevel=sl holdingcost=hc;

    arc / networkid=networkid
          predecessor=head successor=tail
          quantity=qty;

 demand / networkid=networkid
          skuloc=skuloc period=period
          mean=mean variance=variance;
run;
```

Output 3.2.1 contains the policy parameters from the independent solution. Note that the parameters are quite different from those in the "Output Data Set of Two-Echelon Assembly Network."

Output 3.2.1. Output Data Set of the Independent Solution

An Independent Solution vs. a Joint Solution
Independent Solution

Obs	NetworkID	SkuLoc	Description	TargetServiceType	SetupPolicy	UnmetDemand	Period	ReorderLevel	OrderUpToLevel	_STATUS_
1	N4	AP	component 1	0.50	FR	BS	LostSales	1	47	48	SUCCESSFUL
2	N4	AP	component 1	0.50	FR	BS	LostSales	2	47	48	SUCCESSFUL
3	N4	AP	component 1	0.50	FR	BS	LostSales	3	47	48	SUCCESSFUL
4	N4	AP	component 1	0.50	FR	BS	LostSales	4	47	48	SUCCESSFUL
5	N4	AP	component 1	0.50	FR	BS	LostSales	5	47	48	SUCCESSFUL
6	N4	AP	component 1	0.50	FR	BS	LostSales	6	47	48	SUCCESSFUL
7	N4	AP	component 1	0.50	FR	BS	LostSales	7	47	48	SUCCESSFUL
8	N4	AP	component 1	0.50	FR	BS	LostSales	8	47	48	SUCCESSFUL
9	N5	BP	component 2	0.50	FR	BS	LostSales	1	47	48	SUCCESSFUL
10	N5	BP	component 2	0.50	FR	BS	LostSales	2	47	48	SUCCESSFUL
11	N5	BP	component 2	0.50	FR	BS	LostSales	3	47	48	SUCCESSFUL
12	N5	BP	component 2	0.50	FR	BS	LostSales	4	47	48	SUCCESSFUL
13	N5	BP	component 2	0.50	FR	BS	LostSales	5	47	48	SUCCESSFUL
14	N5	BP	component 2	0.50	FR	BS	LostSales	6	47	48	SUCCESSFUL
15	N5	BP	component 2	0.50	FR	BS	LostSales	7	47	48	SUCCESSFUL
16	N5	BP	component 2	0.50	FR	BS	LostSales	8	47	48	SUCCESSFUL
17	N6	FP	finished goods	0.95	FR	BS	LostSales	1	69	70	SUCCESSFUL
18	N6	FP	finished goods	0.95	FR	BS	LostSales	2	69	70	SUCCESSFUL
19	N6	FP	finished goods	0.95	FR	BS	LostSales	3	69	70	SUCCESSFUL
20	N6	FP	finished goods	0.95	FR	BS	LostSales	4	69	70	SUCCESSFUL
21	N6	FP	finished goods	0.95	FR	BS	LostSales	5	69	70	SUCCESSFUL
22	N6	FP	finished goods	0.95	FR	BS	LostSales	6	69	70	SUCCESSFUL
23	N6	FP	finished goods	0.95	FR	BS	LostSales	7	69	70	SUCCESSFUL
24	N6	FP	finished goods	0.95	FR	BS	LostSales	8	69	70	SUCCESSFUL

PROC MIRP provides a functionality that enables users to evaluate the performance of a set of policy parameters of their choice. This functionality can be used here to see whether the independent solution indeed achieves the target service levels.

```
data inventorydata7;
   set out_example6;
   keep NetworkID SkuLoc
        Period ReorderLevel OrderUpToLevel;
   NetworkID="N3";
run;
```

The policy parameters were extracted from the output data set in the independent solution and stored in the inventory data set.

```
title 'An Independent Solution vs. a Joint Solution';
title2 'Independent Solution Using EVALUATION';
proc mirp nodedata=nodedata3 arcdata=arcdata3
          demanddata=demanddata3
          inventorydata=inventorydata7
          out=out_example7
          horizon=8 evaluation lostsales;

     node / networkid=networkid
            skuloc=skuloc leadtime=lt
            servicelevel=sl holdingcost=hc;

      arc / networkid=networkid
            predecessor=head successor=tail
            quantity=qty;

   demand / networkid=networkid
            skuloc=skuloc period=period
            mean=mean variance=variance;

inventory / networkid=networkid
            skuloc=skuloc period=period
            reorderlevel=reorderlevel
            orderuptolevel=orderuptolevel;

   output / fillrate;
run;
```

Notice that the original data sets from the "Two-Echelon Assembly Network" are used here along with the newly created inventory data set. The EVALUATION option in the statement tells the procedures to evaluate the policy parameters stored in the inventory data set. The statement "OUTPUT / FillRate" requests the procedure to output the actual fill rates for the given policy parameters. The evaluation results are shown in Output 3.2.2.

Output 3.2.2. Output Data Set of the Evaluation with the Independent Solution

An Independent Solution vs. a Joint Solution
Independent Solution Using EVALUATION

Obs	Network	Sku	Description	Target Service Type	Service Policy	Unmet Demand	Fill Rate	Period	Reorder Level	Order Up To Level	_STATUS_
1	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	1	69	70	SUCCESSFUL	
2	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	2	69	70	SUCCESSFUL	
3	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	3	69	70	SUCCESSFUL	
4	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	4	69	70	SUCCESSFUL	
5	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	5	69	70	SUCCESSFUL	
6	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	6	69	70	SUCCESSFUL	
7	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	7	69	70	SUCCESSFUL	
8	N3	FP	finished goods	0.95 FR	BS LostSales	0.50843	8	69	70	SUCCESSFUL	
9	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	1	47	48	SUCCESSFUL	
10	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	2	47	48	SUCCESSFUL	
11	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	3	47	48	SUCCESSFUL	
12	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	4	47	48	SUCCESSFUL	
13	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	5	47	48	SUCCESSFUL	
14	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	6	47	48	SUCCESSFUL	
15	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	7	47	48	SUCCESSFUL	
16	N3	AP	component 1	0.50 FR	BS LostSales	0.17453	8	47	48	SUCCESSFUL	
17	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	1	47	48	SUCCESSFUL	
18	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	2	47	48	SUCCESSFUL	
19	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	3	47	48	SUCCESSFUL	
20	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	4	47	48	SUCCESSFUL	
21	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	5	47	48	SUCCESSFUL	
22	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	6	47	48	SUCCESSFUL	
23	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	7	47	48	SUCCESSFUL	
24	N3	BP	component 2	0.50 FR	BS LostSales	0.17453	8	47	48	SUCCESSFUL	

From the output data set, it is quite clear that the independent solution achieves service levels far lower than what is required. The reason is obvious: the policy parameters in the independent solution are much lower than those in the joint solution.

Example 3.3. Intermittent Demand

Intermittent demand is commonly seen when products are slow-moving, such as spare parts. There are two types of uncertainty in intermittent demand: the time between two demands ("demand interval") is random, and so is the size of the demand ("demand size"). Croston's method is widely used in forecasting such demand. It assumes demand to occur as a Bernoulli process, with the demand size from a Normal distribution. With these assumptions, it estimates the average demand interval, and the mean and the variance of the demand size.

PROC MIRP uses similar assumptions as in Croston's method. Instead of Normal distribution, PROC MIRP uses the Gamma distribution to model the demand size. Users need to specify the average demand interval in the node data set, and the mean and the variance of demand size at the demand data set. Consider a single location problem described by the following data set.

```
data nodedata8;
   format networkid $2. skuloc $10.;
   input networkid $3. skuloc $11. leadtime
         servicelevel holdingcost demandinterval;
datalines;
N8 SpareParts 1 0.95 1 4.5
;

data arcdata8;
   format networkid $2. predecessor successor $10.;
   input networkid $3. predecessor $9. successor $11.;
datalines;
N8 EXTERNAL SpareParts
;

data demanddata8;
   format networkid $2. skuloc $10.;
   input networkid $3. skuloc $11.
         period mean variance;
datalines;
N8 SpareParts 1 10 9
;
```

Notice that the average demand interval is 4.5, meaning that demand occurs once every four and a half periods on average. When demand occurs, the average of the demand size is 10 and the variance of the size is 9.

The statements to call PROC MIRP are no different than those with regular demand patterns. The output is given in Output 3.3.1. Note that the estimated PBR is about 4.1 for the suggested policy parameters. This is consistent with the demand pattern: the demand comes once every four and a half periods on average, so the replenishment orders should have similar frequency.

```
title 'Intermittent Demand';
proc mirp nodedata=nodedata8 arcdata=arcdata8
          demanddata=demanddata8 out=out_example8
          horizon=20;

   node / networkid=networkid
          skuloc=skuloc leadtime=leadtime
          servicelevel=servicelevel
          holdingcost=holdingcost;

    arc / networkid=networkid
          predecessor=predecessor
```

```
            successor=successor;

 demand / networkid=networkid
          skuloc=skuloc period=period
          mean=mean variance=variance;

 output / pbr fillrate;
run;
```

Output 3.3.1. Output Data Set of the Problem with Intermittent Demand

Intermittent Demand

Obs	Network ID	Sku Loc	Target Service Policy	Unmet Demand	Fill Rate	PBR	Period	Reorder Level	Order Up To Level	_STATUS_
1	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		1	16	17	SUCCESSFUL
2	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		2	16	17	SUCCESSFUL
3	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		3	16	17	SUCCESSFUL
4	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		4	16	17	SUCCESSFUL
5	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		5	16	17	SUCCESSFUL
6	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		6	16	17	SUCCESSFUL
7	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		7	16	17	SUCCESSFUL
8	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		8	16	17	SUCCESSFUL
9	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		9	16	17	SUCCESSFUL
10	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		10	16	17	SUCCESSFUL
11	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		11	16	17	SUCCESSFUL
12	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		12	16	17	SUCCESSFUL
13	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		13	16	17	SUCCESSFUL
14	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		14	16	17	SUCCESSFUL
15	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		15	16	17	SUCCESSFUL
16	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		16	16	17	SUCCESSFUL
17	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		17	16	17	SUCCESSFUL
18	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		18	16	17	SUCCESSFUL
19	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		19	16	17	SUCCESSFUL
20	N8	SpareParts	0.95 FR BS LostSales	0.95144	6.48100		20	16	17	SUCCESSFUL

Glossary

(s,nQ) policy
: an order that is placed to bring an inventory level that falls at or below the reorder level, s, to just above s. The size of the order is the smallest multiple, n, of the base lot size, Q.

(s,S) policy
: an order that is placed to bring an inventory level that falls at or below the reorder level, s, back to the order-up-to level, S.

average backorder waiting time
: the average amount of time that an item stays in a backordered status. Also, the average time a customer waits for a backordered item.

average cost
: the average cost (holding and replenishment) that is incurred per inventory review period. If backorder penalty costs are present, these are included as well.

average inventory wait time
: the average amount of time an item stays in inventory.

backorder penalty cost
: See penalty cost.

backorder ratio
: the ratio of average backorders divided by average demand.

backordering
: the ordering of goods to replenish inventory.

backordering cost
: the cost that is incurred when a stockout occurs. This cost might include the cost of emergency shipments, the substitution of a less profitable item, or lost goodwill.

backorders
: the total of all outstanding backordered demand. See also backordering.

base lot size
: the minimum order quantity. Replenishment orders can be placed only in multiples of the base lot size. See also (s,nQ) policy.

base period
: the interval of time in which one inventory replenishment order is allowed.

base-stock policy
: an inventory policy that consists of a base-stock level, S, and a base lot size equal to 1. The goal of this policy is to maintain the base-stock level. In other words, when the inventory position falls below the base-stock level, S, an order is placed to bring the inventory position to S. This is a special case of an (s,nQ) policy where Q=1. See also (s,nQ) policy.

carrying cost
: See holding cost.

fill rate
: a service measure that indicates the fraction of demand that is satisfied from on-hand inventory.

filter
: in SAS Inventory Policy Studio, a tool to subset inventory data based on user-specified rules.

fixed lot size
: in inventory replenishment, the required quantity of goods that can be ordered at one time. Large orders must be placed in multiples of the fixed lot size.

fixed ordering cost
: the fixed cost that is incurred every time a replenishment order is placed. This cost includes the expense that is associated with processing the order and is usually independent of the order quantity.

holding cost
: the cost of holding inventory, which includes the expense that is incurred in running a warehouse, handling inventory, and counting inventory. Holding costs might also include the cost of special storage requirements, deterioration of stock, damage, theft, obsolescence, insurance, taxes, or the opportunity cost of money invested. Also called carrying cost.

inventory on order
: the total of all outstanding replenishment orders.

lead time
: the interval of time between placing a replenishment order and receiving the ordered item.

lead-time demand mean
: the average demand that occurs in the period between ordering a product and receiving that product.

lead-time demand variance
: the variability of the demand that occurs during the period between ordering a product and receiving that product.

maximum order frequency
: the maximum number of orders that can be placed during a fixed time period

minimum order size
: the minimum quantity of goods that can be ordered at one time

modified fill rate
: in SAS Inventory Policy Studio, a service measure that is calculated as 1 - backorder ratio. See also backorder ratio.

on-hand inventory
: the amount of inventory that is physically in stock.

optimal policy
: the inventory replenishment policy that minimizes the average cost, including ordering, holding, and backorder penalty costs.

order-up-to level
: the target inventory level.

ordering cost
: See fixed ordering cost.

penalty cost
: the cost that is incurred when a stockout occurs. This cost might include the cost of emergency shipments, the cost of substituting of a less profitable item, or the cost of lost goodwill.

project
: in SAS Inventory Policy Studio, a project enables you to manage information about a set of products in inventory. Within a project, various inventory replenishment policies are evaluated to determine the optimal policy.

Project Table
: in SAS Inventory Policy Studio, a data repository for a project that contains all variables that are used in the project and their associated inventory policy roles.

ready rate
: the probability that the on-hand inventory level at the end of a review time period is positive.

reorder level
: the inventory level at which a replenishment order should be placed.

review period
: in inventory replenishment, the amount of time that elapses between assessing inventory levels.

review-time demand
: the amount of demand that occurs during a single review period.

role
: a value that determines how a variable is analyzed for an inventory replenishment policy. For example, the role of a variable can be assigned to roles such as holding cost, policy type, or lead time mean.

scenario
: in SAS Inventory Policy Studio, a work area for a project in which inventory policies can be optimized, evaluated, and promoted to the Project Table.

sensitivity analysis
: a method to quantify the effect that an independent variable has on a dependent variable. For example, you can review the effect that varying minimum order size has on order costs for a product.

service measure
: a measure of inventory policy performance. Fill rate, ready rate, and backorder ratio are common service measures.

SKU (Stock-Keeping Unit)
: a specific item identification number.

unit cost
: the cost of acquiring an item of inventory from a supplier.

unit price
: the price that is charged to the consumer for an item of inventory.

Subject Index

T

U

V

W

Syntax Index

L

M

N

O

P

Q

R

S

T

U

V

W

Your Turn

If you have comments or suggestions about *SAS Inventory Optimization 1.3: User's Guide,* please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Publishing
SAS Campus Drive
Cary, NC 27513
E-mail: **yourturn@sas.com**

For suggestions about the software, please return the photocopy to

SAS Institute Inc.
Technical Support Division
SAS Campus Drive
Cary, NC 27513
E-mail: **suggest@sas.com**