AIMMS

Tutorial for Professionals

September 2018

AIMMS

AIMMS B.V.
Diakenhuisweg 29-35
2033 AP Haarlem
The Netherlands
Tel.: +31 23 5511512

AIMMS Inc.
11711 SE 8th Street
Suite 303
Bellevue, WA 98005
USA
Tel.: +1 425 458 4024

AIMMS Pte. Ltd.
55 Market Street #10-00
Singapore 048941
Tel.: +65 6521 2827

AIMMS
SOHO Fuxing Plaza No.388
Building D-71, Level 3
Madang Road, Huangpu District
Shanghai 200025
China
Tel.: ++86 21 5309 8733

Email: info@aimms.com
WWW: www.aimms.com

Contents

Common AIMMS Shortcut Keys

Key	Function
F1	Open AIMMS Help
F2	Rename the selected identifier
F3	Find and repeat find
F4	Switch between edit mode and end-user mode (for the active page)
F5	Compile all
F6	Run *MainExecution*
Alt+F6	Switch to debugger mode
F7	Save the active page
F8	Open Model Explorer
Ctrl+F8	Open Identifer Selector
F9	Open Page Manager
Alt+ F9	Open Template Manager
Ctrl+ F9	Open Menu Builder
F11	Open Identifer Info dialog
Ctrl+ B	Insert a break point in debugger mode
Ctrl+ D	Open Data Page
Ctrl+ F	Open Find dialog
Ctrl+ M	Open Message Window
Ctrl+ P	Open Progress Window
Ctrl+ T	View Text Representation of selected part(s)
Ctrl+Shift + T	View Text Representation of whole model
Ctrl+ W	Open Wizard
Ctrl+ Space	Name completion
Ctrl+ Shift+Space	Name completion for AIMMS Predeclared Identifers
Ctrl+ Enter	Check, commit, and close
Insert	Insert a node (when single insert choice) or Open Select Node Type dialog (when multiple insert choices)

Part I

Introduction

Chapter 1

Introduction

Ways to learn AIMMS ...

There are several ways in which you can learn the AIMMS language and acquire a basic understanding of its underlying development environment. The following opportunities are available.

- There are two *tutorials* on AIMMS to provide you with some initial working knowledge of the system and its language. One tutorial is intended for students, while the other is aimed at professional users of AIMMS.
- There is a *model library* with a variety of examples to illustrate simple and advanced applications together with particular aspects of both the language and the graphical user interface.
- There are three *reference books* on AIMMS, which are available in PDF format and in hard copy form. They are *The User's Guide* to introduce you to AIMMS and its development environment, *The Language Reference* to describe the modeling language in detail, and *Optimization Modeling* to enable you to become familiar with building models.
- There is a *Function Reference* that provides a detailed description of all available functions in AIMMS, including their arguments and return type. It also provides detailed information on predeclared identifiers available in AIMMS.
- There is an *Online Help* that provides many details on the usage of AIMMS. You can get online help for most of the tools, attribute forms and objects within the AIMMS system through the Context Help facilities.
- There are *workshops* on AIMMS that take you through the entire development cycle of a complete decision support application by means of a sequence of 'hands-on' sessions. For more information about the workshops refer to our site www.aimms.com.

... for beginners

As a student studying optimization modeling, you may not have much time for learning yet another tool in order to finish some course work or homework requirements. In this case, concentrate your efforts on the tutorial for beginners. After completing that tutorial, you should be able to use the system to build your own simple models, and to enter your own small data sets for subsequent processing. The book on *Optimization Modeling* may teach you some useful tricks, and will show you different (mostly non-trivial) examples of optimization models.

... for professionals

As a professional in the field of optimization modeling you are looking for a tool that simplifies your work and minimizes the time needed for model construction and model maintenance. In this situation, you cannot get around the fact that you will need to initially invest substantial time to get to know several of the advanced features that will subsequently support you in your role as a professional application builder. Depending on your skills, experience, and learning habits you should determine your own individual learning path. Along this path you are advised to work through the extensive tutorial especially designed for professionals. This tutorial for professionals provides a good start, and should create excitement about the possibilities of AIMMS. Individual examples in the library, plus selected sections of the three books, will subsequently offer you additional ideas on how to use AIMMS effectively when building your own advanced applications.

Tutorials are different in scope

The one-hour tutorial for students is designed as the bare minimum needed to build simple models using the AIMMS **Model Explorer**. Data values are entered manually using data pages, and a student can build a page with objects to view and modify the data. The extensive tutorial for professionals is an elaborate tour of AIMMS covering a range of advanced language features plus an introduction to all the building tools. Especially of interest will be the modeling of time using the concepts of horizon and calendar, the use of quantities and units, the link to a database, and the connection to an external DLL (Dynamic Link Library). Even then, some topics such as efficiency considerations (execution efficiency, matrix manipulation routines) and the AIMMS API (Application Programming Interface) will remain untouched.

Several days are required . . .

The current extensive tutorial for professionals requires a substantial amount of input. Several days are required to build the entire application from scratch. It is possible, however, to import portions of the model and its interface to adapt the tutorial to your own time restrictions.

. . . plus access to MS Access

This tutorial reads data from a database stored in MS Access format using ODBC (Open DataBase Connectivity). Therefore, you will need to have Microsoft Access on your machine in order to complete the course.

Preview of your output

In this tutorial you will build your own end-user interface. One of the pages that you will construct is shown in Figure 1.1.

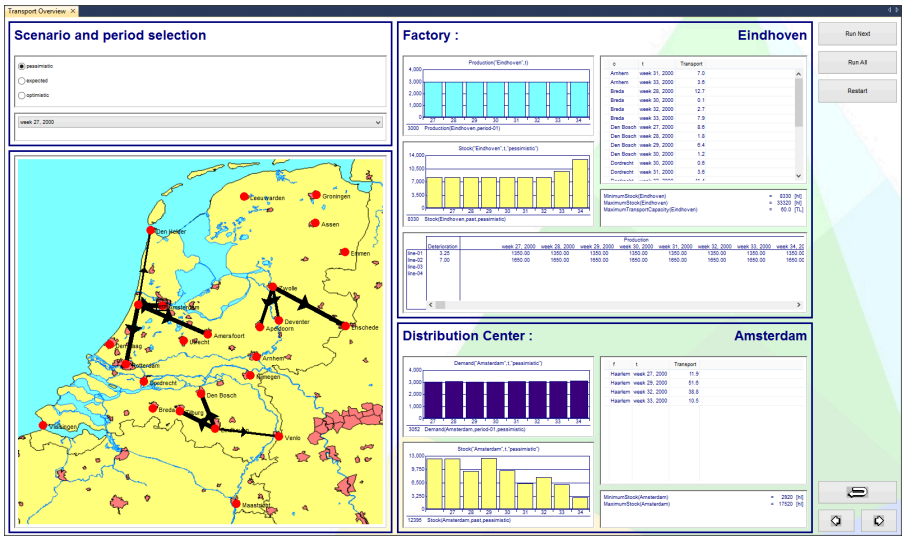


Figure 1.1: An overview of optimal transport data

Chapter 2

Problem Description

This chapter

In this chapter you will find a description of the problem to be translated into an optimization model. The problem statement covers several pages, typical for a professional application in the field of planning and scheduling. The overall goal in this problem is to obtain a production and maintenance plan on a weekly basis for a total planning horizon of one year. The corresponding mathematical model is provided in Chapter 3.

2.1 Initial problem components

Planning horizon

The application discussed in this tutorial considers a planning horizon of one year and individual planning periods of one week. The overall goal of the application will be to develop a robust production and maintenance schedule.

Production and distribution

Consider the production and distribution of a specific soft drink on a weekly basis. There are 3 factories and 22 distribution centers, all located in the Netherlands (see Figure 2.1). Every week, truckloads of soft drinks are distributed from the factories to the distribution centers. There is an upper bound on the number of truck loads that can be moved from a particular factory during a single week.

Production lines

Each factory has several production lines each with a fixed production level measured in terms of hectoliters per day. During any particular week, a production line is either operational at a fixed production level, or does not produce at all.

Mode switches

The term *mode switch* of a production line refers to an on/off change in production. Thus a mode switch occurs when a production line becomes operational during a particular week if it was not operational during the previous week, and vice versa.

Figure 2.1: The Netherlands

Storage

There are storage facilities at both factories and distribution centers. Stock, like production, is measured in hectoliters. There is a reserve stock at each location, and storage is limited.

Cost components

Total cost, measured in terms of dollars, is made up of several cost components related to production, distribution, storage, and mode switches. The first three of these components are self-explanatory, but the final component deserves some explanation. In this application some of the workers employed to work on the production line are temporary workers, but it is assumed that frequent hiring and layoffs are undesirable. Therefore, an extra artificial cost term is introduced to penalize mode switches.

2.2 Maintenance and vacation planning

Maintenance requirement ...

Production lines need to be maintained on a regular basis dependent on their associated deterioration rate. It is assumed that when a production line has been in full use for a period of 16 weeks, then shortly thereafter it must be closed for a week of maintenance which will be performed by the crew previ-

ously working on that line. If a production line has not been in use for more than 64 weeks, then it must have maintenance in the week prior to becoming operational. If the line has been in and out of use over a period of weeks, then every week of non-use increases the deterioration level by an amount equal to one quarter of a week of use.

... causes no mode switches

The workers on a production line also perform the line maintenance. Therefore, the mode switch penalty, described in the previous section, does not apply when production comes to a halt or starts again as a result of maintenance.

... and preserves continuity

To guarantee continuity of production in each factory, there exists an additional requirement that only one production line per factory can be maintained at the same time.

Inactive days

The production lines in the factories are closed during weekends and official holidays. In addition, there is no distribution of soft drinks from the factories to the distribution centers on these particular days. As a result, a production week always consists of five or less working days.

Vacation periods

In addition to the official holidays, there are whole periods reserved when workers have the opportunity to take a vacation. For planning purposes, it is assumed that not every worker will be on vacation, and that the level of production for all the lines in use will drop by a particular percentage during such a vacation period. The mode switch penalty does not apply when such a drop or subsequent increase in production takes place.

2.3 Multiple demand scenarios

Demand is uncertain

The weekly demand for soft drinks to be supplied by the distribution centers to customers is not exactly known. Variations over the years have been observed, which is why there is a reserve stock. Nevertheless, when building a model with demand as a parameter, demand values for the weeks to come must be chosen. Such a set of demand values is referred to as a *demand scenario*.

Three scenarios

Instead of selecting a single demand scenario, the use of three demand scenarios is proposed in order to obtain a more robust production and maintenance plan. These scenarios reflect an expected, a somewhat pessimistic and a somewhat optimistic demand, thereby capturing overall demand behavior over the previous several years.

Robust planning

The key idea of robust planning is to make a single production and maintenance plan that is feasible for all three demand scenarios. The only decisions that are allowed to be different with each demand scenario are those related to distribution and storage. For more details on scenario-based optimization you may want to consult Chapters 16 and 17 of AIMMS, *Optimization Modeling*.

2.4 Planning objective

Overall goal

The overall goal of the company is to obtain a production and maintenance plan on a weekly basis for a total planning horizon of one year. The resulting plan should be in the form of a Gantt chart (see Figure 2.2) at the level of the individual production lines at each of the three factories. Such a plan provides insight into the use of capacity, the build up of inventories, and the need to make arrangements for temporary workers to be hired in each of the factories.

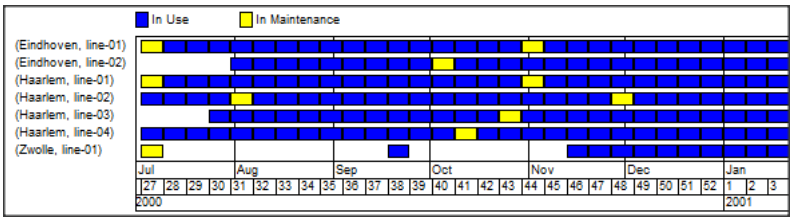


Figure 2.2: Selected portion of a Gantt chart

Specific goal

The specific objective of the mathematical programming model to be built is to minimize total cost over the planning horizon. It is straightforward to specify the individual cost components related to production and mode switches. The cost components related to storage and distribution, however, are scenario-dependent and thus should be weighted in the objective with the scenario probabilities. In this application, the assumption has been made that the probabilities of the pessimistic and optimistic scenarios are each equal to 0.25.

2.5 A rolling horizon approach

Size problematic

In practical applications of the type described in this chapter the number of factories and distribution centers is usually much larger than the few locations specified here. In addition, most applications have more than one product. With the one-year planning horizon, on a weekly basis, the mathematical program as built in this tutorial is likely to be too large to be solved all at once in a real life situation.

Restrict horizon

One remedy would be to consider a shorter planning horizon. The effect on the number of decision variables is immediate, as all of them are indexed with weeks. The disadvantage of this approach is clear: it does not satisfy the management requirement to plan for a full year.

Rolling horizon

The approach followed in this application is to run a sequence of mathematical programs each with a planning horizon for intervals of 8 weeks. Once the first program is solved for week one, all decisions concerning this first week are considered to be final. The subsequent mathematical program then starts at week two, and again, all production and maintenance decisions concerning this second week are fixed. This process continues until the mathematical program covers the last 8 weeks of the full year planning horizon.

Dependency on future data

Rolling horizon models are a compromise between speed and accuracy. If the planning interval is long, the solution should be more optimized. The corresponding mathematical program is however larger in size, and could take up a considerable amount of computational time. The length of the planning interval should certainly reflect the insensitivity of future data to first-period decisions. This choice is application dependent. A planning interval of 8 weeks was adequate for the problem in this tutorial.

Maintenance external

An advantage of this rolling horizon approach is that maintenance planning can, for the most part, be placed outside the mathematical program. Every time the decisions corresponding to a first week are committed, their effect on maintenance can be registered by adjusting a deterioration parameter for each production line. Once maintenance for a particular production line is due within the next horizon of 8 weeks, the level of production during the corresponding estimated maintenance period is set to zero. The specific implementation details are discussed later.

Evaluation

From the point of view of a tutorial, it is an interesting exercise to work with time and a rolling horizon. In practical applications, however, caution is needed: a short planning horizon may not be sufficient to take the relevant future into account. In this example, a planning horizon of 8 weeks was considered sufficiently large because demand fluctuations are not drastic, and storage safety buffers at the locations are of a reasonable size.

Chapter 3

Model Description

This chapter

In this chapter you will find a description of the mathematical program corresponding to the problem description of the previous chapter.

3.1 Product flow

Indices

The following indices capture the dimensions of the problem, and are used throughout this chapter.

Indices:

l	*locations*
f	*factories* $\subset$ *locations*
c	*distribution centers* $\subset$ *locations*
p	*production lines*
t	*time periods*
s	*demand scenarios*

Decision variables

The following product flow decision variables determine the levels of production, distribution and storage.

Variables:

q_{ft}	*total factory production [hl (hectoliter)]*
u_{fpt}	*binary to indicate that production line is in use*
x_{fcts}	*transport [TL (truckload)]*
y_{lts}	*stock [hl]*

Note that the production variables are identical for all demand scenarios, while the distribution and storage variables can vary for each scenario. Note also that both hectoliters and truckloads are used to measure the quantities of soft drinks. In this tutorial a truckload is defined as 12 cubic meters.

Parameters ...

The following product flow related parameters are used in this chapter.

Parameters:

D_{cts}	*demand [hl]*
L_t	*actual period length [day]*

Q_{fp}	*production at full operation [hl/day]*
M_{fpt}	*binary to indicate that production line is in maintenance*
V_{ft}	*binary to indicate a vacation period*
F	*drop in workforce during vacation periods (fraction)*
A_{fpt}	*potential production [hl]*
X_f	*number of available truckloads [TL]*
$\overline{Y}_l$	*maximum stock level [hl]*
$\underline{Y}_l$	*minimum stock level [hl]*

... and their data source

The parameters related to production line capacity, demand and vacations will be read from external data sources. The maintenance parameter will be determined as part of the rolling horizon solution process.

Potential production determination

The potential production of a production line, A_{fpt}, is dependent on the maintenance and vacation parameters, and is defined as follows.

$$A_{fpt} = L_t(1 - M_{fpt})(1 - F \cdot V_{ft})Q_{fp}, \quad \forall (f, p, t)$$

Note that nonzero values of parameters M_{fpt}, F and V_{ft} result in the potential production, A_{fpt}, being less than the production level at full operation Q_{fp}.

Balance constraint

The following stock balance constraint relates stock to previous stock, production, distribution and demand.

$$\begin{aligned} y_{lts} &= y_{l,t-1,s} + q_{lt} + \sum_f x_{flts} - \sum_c x_{lcts} - D_{lts}, \quad \forall (l, t, s) \\ y_{lts} &\in [\underline{Y}_l, \overline{Y}_l], \quad \forall (l, t, s) \end{aligned}$$

Domain restrictions

Note that this balance constraint is used for all locations (thus both factories and distribution centers), and that particular terms inside this constraint must on some occasions be interpreted as non-existent. For instance, the production term is non-existent for distribution centers, while the demand term is non-existent for factories. In AIMMS you can specify a global index domain for each identifier, and the system will automatically restrict all identifier references to such an index domain.

Factory production

Using the potential production parameter A_{fpt} as defined previously, it is now straightforward to determine the total weekly production at each of the factories.

$$q_{ft} = \sum_p A_{fpt} u_{fpt}, \quad \forall (f, t)$$

Transport limitation

It is also straightforward to model the restriction that the number of truckloads to be moved from a factory during a particular week is limited by the number of trucks available at that factory.

$$\sum_{c} x_{fcts} \leq X_f, \quad \forall (f,t,s)$$

Note that the above planning constraint is, in practice, a simplification of the detailed transport capacity scheduling limitations. In scheduling applications the routing of vehicles, the distances to be traveled, plus the time-windows for the drivers would all be key factors in the determination of a final schedule. These factors are considered to be less important for the current one-year plan.

3.2 Mode switches

Additional notation

The following variable is needed to register the mode switches,

Variable:

v_{fpt}	*binary to register a mode switch*

Mode switch registration

The registration of mode switches seems tricky at first, but becomes straightforward with some additional explanation. Consider the following two inequalities.

$$\begin{aligned} v_{fpt} &\geq u_{fpt} - u_{fp,t-1}, \quad \forall (f,p,t) \\ v_{fpt} &\geq u_{fp,t-1} - u_{fpt}, \quad \forall (f,p,t) \end{aligned}$$

Whenever a production line switches from being used to not being used, or vice versa, the switch-registration variable v will be greater than or equal to unity. The penalty term in the objective discussed in the next section will ensure that this variable remains as small as possible. Thus, without a switch in the use of a production line, the variable v will be zero.

Effect on maintenance

Consider a production line in use. Whenever such a line needs to be maintained, its production drops to zero. Immediately following the maintenance week, its production is likely to restart. In this case, the change in production is not considered to be a mode switch. The definition of the potential production parameter, A_{fpt}, in the previous section is consistent with this observation. The maintenance parameter, M_{fpt}, is set to one when maintenance is planned, which forces the potential production parameter, A_{fpt}, to be zero for that week. The penalty term in the objective function, however, will cause the u variable to remain at level one, thus avoiding the unwanted mode switch. A similar argument applies to maintenance while a line is not in use.

3.3 Objective

The following parameters and variables are needed to specify the objective function of the mathematical program. *Additional notation*

Parameters:

C_f^q	*unit production cost [$/hl]*
C_l^y	*unit stock cost [$/hl]*
C_{fc}^x	*unit transport cost [$/TL]*
C^v	*penalty cost due to mode switch [$]*
P_s	*demand scenario probability*

Variables:

r_s	*demand scenario cost [$]*
z	*total cost [$]*

The cost per single demand scenario is the sum of the production costs, the scenario-specific storage and distribution costs, plus a penalty term to reflect the costs associated with mode switching. *Cost per scenario*

$$r_s = \sum_{ft} C_f^q q_{ft} + \sum_{lt} C_l^y y_{lts} + \sum_{fct} C_{fc}^x x_{fcts} + \sum_{fpt} C^v v_{fpt}, \quad \forall s$$

The total cost to be minimized is simply the weighted sum of the scenario costs. *Minimize total cost*

Minimize:

$$z = \sum_s P_s r_s$$

3.4 Model summary

The full mathematical description of the optimization model can now be summarized as follows.

Minimize:

$$z = \sum_s P_s r_s$$

Subject to:

$$\begin{aligned}
y_{lts} &= y_{l,t-1,s} + q_{lt} + \sum_f x_{flts} - \sum_c x_{lcts} - D_{lts} && \forall (l,t,s) \\
q_{ft} &= \sum_p A_{fpt} u_{fpt} && \forall (f,t) \\
\sum_c x_{fcts} &\le X_f && \forall (f,t,s) \\
v_{fpt} &\ge u_{fpt} - u_{fp,t-1} && \forall (f,p,t) \\
v_{fpt} &\ge u_{fp,t-1} - u_{fpt} && \forall (f,p,t) \\
r_s &= \sum_{ft} C_f^q q_{ft} + \sum_{lt} C_l^y y_{lts} + \\
&\quad \sum_{fct} C_{fc}^x x_{fcts} + \sum_{fpt} C^v v_{fpt} && \forall s \\
u_{fpt} &\in \{0,1\} && \forall (f,p,t) \\
x_{fcts} &\ge 0 && \forall (f,c,t,s) \\
y_{lts} &\in [\underline{Y}_l, \overline{Y}_l] && \forall (l,t,s) \\
v_{fpt} &\ge 0 && \forall (f,p,t)
\end{aligned}$$

Part II

Model Declarations

Chapter 4

Auxiliary Project Files

This chapter

In this chapter you will find instructions on how to install the auxiliary files that are needed to complete this tutorial. In addition, the process to import model sections and pages is explained.

4.1 Directory structure

Creating folders

You are advised to use Windows Explorer to first create a dedicated folder in which to store your AIMMS projects, and then create a subfolder to store the particular AIMMS project of this tutorial. Figure 4.1 serves as an illustration.

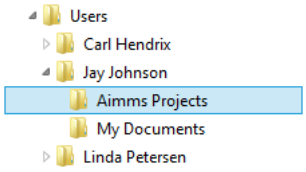


Figure 4.1: A selection of subfolders

Auxiliary project files

There are several files that you will need or find convenient while building the AIMMS project described in this tutorial. Among these files are:

- a text file containing example project data,
- an MS Access database containing project data,
- a DLL with a function external to AIMMS,
- several bitmaps for the end-user interface,
- a number of model sections for possible import,
- a number of cases and datasets for possible,
- a copy of this tutorial in PDF format.

Download the auxiliary project files

On request you can obtain a copy of the auxiliary project files listed above as well as a copy the completed tutorial project. You can also download the files yourself from the two following links. Download the file containing the correct version based on the version of AIMMS you plan to use.

AIMMSTutorialProjectFiles(32bit).zip

AIMMSTutorialProjectFiles(64bit).zip

Copying the relevant subdirectories

Extract the compressed zip file to a known location on your computer. The file contains two subdirectoies, 'Softdrink Planning - Auxiliary Files' and 'Softdrink Planning - Completed Project' In the directory 'Tutorial Softdrink Planning - Auxiliary Files', you will find six subdirectories. Please copy these six subdirectories from the Aimms directory to a newly created Softdrink Planning project subdirectory.

Directory structure

The directory structure of your project should now look like the one shown in Figure 4.2.

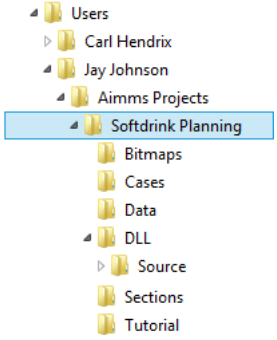


Figure 4.2: The structure of the tutorial project directory

4.2 External project files

Data subdirectory

The 'Data' subdirectory should contain three files. The file 'Softdrink Planning.mdb' contains a MS Access database containing the input data required in this tutorial, the files 'Softdrink Planning.dsn' specifies a ODBC File Data Source that AIMMS uses to connect to the MS Access database, and the third file 'Locations.dat' contains some example data that will be used in Chapter 5.

DLL subdirectory

The 'Dll' subdirectory of your tutorial project should contain a file 'External Routines.dll' and a subdirectory 'Source' for text based systems. The DLL file contains a function that is external to AIMMS, but that can be called from within AIMMS using the external function concept. The 'Source' subdirectory of the 'Dll' directory contains the Microsoft Visual C++ 6.0 project that has been used to create the 'External Routines.dll' file.

Bitmaps directory

The 'Bitmaps' subdirectory contains several bitmap files that you will use when developing the end-user interface. These bitmaps will enhance the appearance of your end-user interface. The following files are available:

- 'AIMMS Logo.bmp'
- 'Background.bmp'
- 'Button Next.bmp'
- 'Button Prev.bmp'
- 'Button Up.bmp'
- 'Netherlands.bmp'

4.3 Importing model sections

Importing serves a need

When working through the several chapters of this extensive tutorial for professionals, you may arrive at a point where you want to skip some of the work required from you. In this case you can bypass your own entries, and import one or more model sections to continue with the tutorial in a more advanced state.

Sections subdirectory

The 'Sections' subdirectory contains several model section files for possible import:

- 'Absentee Overview.ams'
- 'Data Management.ams'
- 'Database Link.ams'
- 'DLL Link.ams'
- 'Planning Overview.ams'
- 'Production Overview.ams'
- 'Production and Maintenance Model.ams'
- 'Quantities and Units.ams'
- 'Rolling Horizon Procedures.ams'
- 'Scenario Overview.ams'
- 'Softdrink Planning Menubar.ams'
- 'Time.ams'
- 'Transport Overview.ams'

Illustrating the import process

When you import the Quantities and Units section (equivalent to the model section that is created in Section 6.2) into your model, all the identifiers that you normally would have created in Section 6.2 will be part of your model. Note that at this point in the tutorial you should not execute any import step. The actions described below are really for later reference when there is a need to import.

- ▶ select the Quantities and Units in the model tree,
- ▶ from the **Edit** menu, select the **Import** command,

- select the file 'Quantities and Units.ams' in the **Import Model Section** dialog box, and
- press the *Open* button.

Confirming import

At this point a **Confirm Import** dialog box will appear as in Figure 4.3. This dialog box lists the changes as a consequence of the planned import. To confirm, you should press the *OK* button.

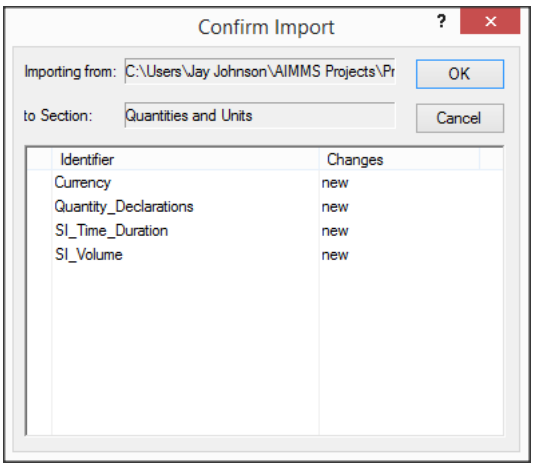


Figure 4.3: The **Confirm Import** dialog box

Verifying a successful import

To verify that the import step is correctly executed, one can inspect the contents of the Quantities and Units section in the **Model Explorer**.

4.4 Loading cases

Cases

To save time and effort while completing this tutorial, you may want to import data instead of entering or computing these data. The specification of the holidays and vacation weeks can be avoided by importing the corresponding case.

Cases and datasets directory

The 'Cases' subdirectory should contain the following three data files:

- 'Holiday and Vacation Data.data'
- 'Initial Data From Database.data'
- 'Solution After First Roll.data'

Illustrating the 'Load Case' process

In this section, the loading of cases will be illustrated by importing the data from the case 'Holiday and Vacation Data.data'. This case contains specified holidays and vacation weeks described in the end of Chapter 12.2. To load the case you should perform the following steps:

- Select **Data** in the menubar,
- go to **Load Case - into Active...**
- select the file 'Holiday and Vacation Data.data' from the **Open Case File** dialog box.

Now you have loaded the data into your active case.

Chapter 5

Getting Acquainted

This chapter

In this chapter, you will create your first very small AIMMS model plus an end-user page that requires minimal effort. The main purpose of this chapter is to give you a quick introduction to the basic functionality of AIMMS.

5.1 Starting a new project

Starting AIMMS

Assuming that AIMMS 4 has already been installed on your machine, execute the following sequence of actions to start AIMMS:

- press the **Launch AIMMS** button in the taskbar,
- select the latest version of **Aimms 4** on your computer from the list, and
- select and click on the **Launch** button to start AIMMS.

Next you will see the AIMMS splash screen. Once AIMMS is ready for use the splash screen will disappear and the AIMMS window will open and display the **Start Page**. Should you encounter the AIMMS **Tip of the Day** dialog box, please close it, because it is not relevant at this point.

Creating a new project from within AIMMS

Press the **New Project** button, which is located in the leftmost position on the AIMMS toolbar. The dialog box shown in Figure 5.1 will then appear, requiring you to take the following actions:

- specify 'Softdrink Planning' as the project name,
- press the **Wizard** button to select, e.g., the folder '`C:\Documents and Setting\Jay Johnson\AIMMS Projects\`' for your AIMMS projects,
- change Default UI from 'WebUI' to 'WinUI'. Uncheck the options 'Include WebUI Library' and 'Include Pro Library', and
- press the **OK** button.

Note that AIMMS will automatically extend the project folder with the project name. This automatic facility is linked to the use of the **Wizard** button. If you enter the project folder by hand, no automatic extension takes places and AIMMS will accept the folder name as you specified.

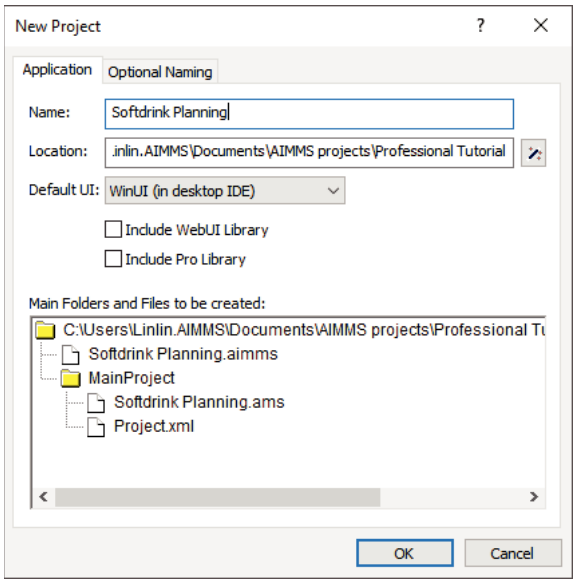


Figure 5.1: The **New Project** wizard

Having completed the **New Project** wizard, AIMMS will open the **Model Explorer** (see Figure 5.2) for the 'Softdrink Planning' project, and you are ready to specify your model.

Project toolbar

You will notice that the AIMMS toolbar has been extended with a project toolbar  to help you further develop the model and its associated end-user interface. The available tools are:

- the *Model Explorer*,
- the *Identifier Selector*,
- the *Page Manager*,
- the *Template Manager*,
- the *Menu Builder*.

These tools can be accessed through the **Tools** menu as well.

Creating a new project from within the Windows Explorer

Alternatively, you can use the right-mouse popup menu command **New-AIMMS Project File** from within the Windows Explorer to create a new project from scratch. In that case, the **New Project** wizard shown in Figure 5.1 will automatically pop up, and the new AIMMS project will be created in the current subdirectory.

5.2 The Model Explorer

Opening the **Model Explorer**

Once a new project is created, the **Model Explorer** will be opened automatically, and the initial model tree as shown in Figure 5.2 will be shown. The **Model Explorer** can also be opened manually by pressing the **Model Explorer** button on the toolbar or by pressing the *F8* key. In the initial model tree you will see a predefined empty *declaration section* together with three predefined *procedures*.

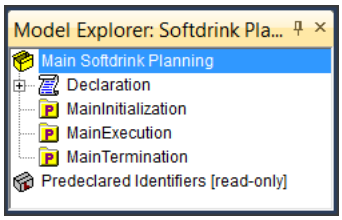


Figure 5.2: The initial model tree

5.2.1 Entering a set identifier

Opening the declaration section

The declaration of model identifiers requires you to first expand the declaration node by double-clicking on the scroll icon (and not on the name itself). Instead of double-clicking you can open the declaration section by pressing the right arrow key after first having selected the corresponding node in the model tree. Once you have opened the declaration section, the **New Identifier** buttons on the toolbar will be enabled.

Creating the set `Locations`

To create a set of locations you should take the following actions:

- press the **New Set** button to create a set identifier in the model tree,
- specify 'Locations' as the name of the set, and
- press the *Enter* key to register the name.

There are alternative ways to create a new identifier using either the **Insert** command in the right-mouse pop-up menu or the *Insert* key.

Opening an attribute form

For every node in the model tree, you can specify additional information as *attributes* belonging to that node. AIMMS lets you view and change the values of these attributes in an *attribute form*. To open an attribute form you can choose any one of the following possibilities:

- select a node in the model tree and press the *Enter* key,
- double-click on the name of the node in the model tree, or

- select a node in the model tree and press the **Attributes** button .

Double-clicking on icon or name

You have now observed the different results obtained when double-clicking on either the *icon* or the *name* of an intermediate node. The first option opens a lower level in the model tree, while the second option opens the corresponding attribute form.

The initial attribute form

Next, you need to declare the index `l` as an attribute of the set `Locations`. You should first open the attribute form of the set `Locations`. The resulting initial attribute form is shown in Figure 5.3.

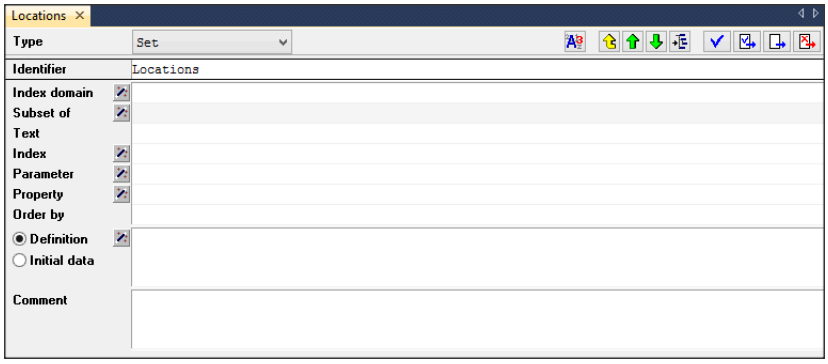


Figure 5.3: The initial attribute form of the set 'Locations'

Declaring the index `l`

To declare the index `l` as an attribute of the set `Locations`, execute the following sequence of actions:

- move the mouse cursor to the **Index** attribute field, and click in the empty edit field,
- enter the letter 'l' (without the quotes), and
- complete the attribute form by pressing the **Check, commit and close** button .

Instead of using the **Check, commit and close** button you could have also used the *Ctrl-Enter* key combination to commit your changes. Figure 5.4 contains the resulting model tree.

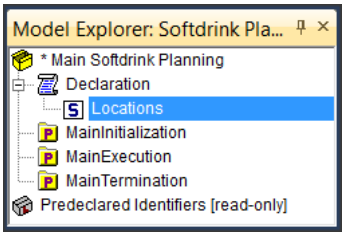


Figure 5.4: The intermediate model tree with the set `Locations`

Saving your changes

The asterisk ('*') on the left of the model node `Main Softdrink Planning` indicates that the edits to your project have not yet been saved to disk. To save your work, please press the **Save Project** button on the toolbar. Alternatively, you could have used the *Ctrl-S* key combination.

Creating the parameter `XCoordinate`

The declaration of a parameter is similar to the declaration of a set. In this chapter, two parameters are introduced to contain the geographical longitude (x) and latitude (y) coordinates of every location in the set `Locations`. To enter the parameter `XCoordinate(l)`, you should execute the following actions:

- press the **New Parameter** button on the toolbar to create a new parameter in the model tree,
- specify 'XCoordinate(l)' as the name of the parameter, and
- press the *Enter* key to register the name.

Note that parentheses are used to automatically add the index domain `l` to the identifier `XCoordinate`.

Creating the parameter `YCoordinate`

The parameter `YCoordinate(l)` can be added in the same way. Should you make a mistake in entering the information, you can always re-edit a name field by first selecting the corresponding node in the model tree followed by a single mouse click within the name field. Alternatively, you can use the *F2* key to enter edit mode.

Checking your model

You have now entered the set `Locations` and the two parameters `XCoordinate` and `YCoordinate`. The resulting model tree is shown in Figure 5.5. By pressing the *F5* key you can instantly check the validity of your model. You will only receive a message in the event of an error or warning. Once the validity of your model has been verified, you should save your work by pressing the **Save Project** button on the toolbar.

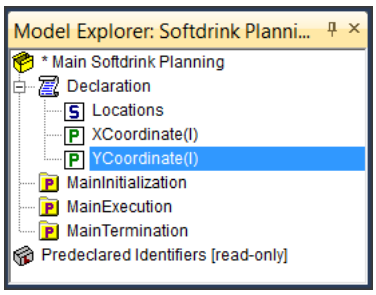


Figure 5.5: The model tree thus far

5.3 Reading data

Data initialization

To be able to briefly illustrate some AIMMS features at this point in the tutorial we will read in some initial data from an external text file named 'Locations.dat' located in the 'Data' directory. This file contains initial data for the set Locations as well as the corresponding coordinates for these locations.

Viewing text files

To view the contents of the initial data file, you can open it with an external text editor or use the internal AIMMS text editor which can be accessed from the **File** - **Open** - **Text File**... menu. In the **Open File** dialog box you should select the 'All Files (*.*)' option to be able to select the file 'Locations.dat'. Figure 5.6 shows the result if you use the internal AIMMS text editor.

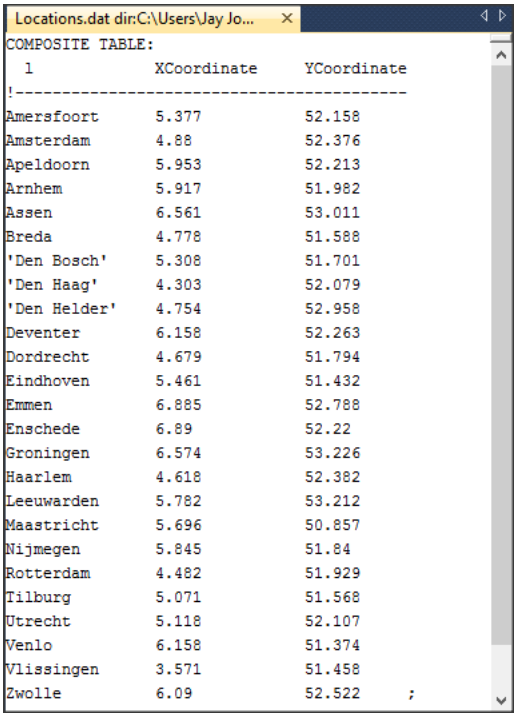


Figure 5.6: The AIMMS internal text editor containing the file '`Locations.dat`'

MainInitialization ...

To instruct AIMMS to initialize its data using the file '`Locations.dat`', you should now enter a read statement in the standard `MainInitialization` procedure. This procedure is automatically executed whenever the project is opened. To achieve this, you should perform the following actions:

- select the `MainInitialization` procedure node in the model tree,
- open its attribute form,
- specify the following line of text as its body argument:

```
read from file "Data\\Locations.dat";
```

- and complete the attribute form by pressing the **Check, commit and close** button  .

Note that AIMMS uses the double backslash in the **Body** attribute of the `MainInitialization` procedure. The single backslash character has already been reserved by AIMMS to denote special characters inside strings. This choice corresponds to the conventions in the C programming language. For instance, '`\n`' denotes the 'return' character, and '`\t`' denotes the 'tab' character.

Figure 5.7 contains the attribute form of the procedure MainInitialization. *... and its attribute form*

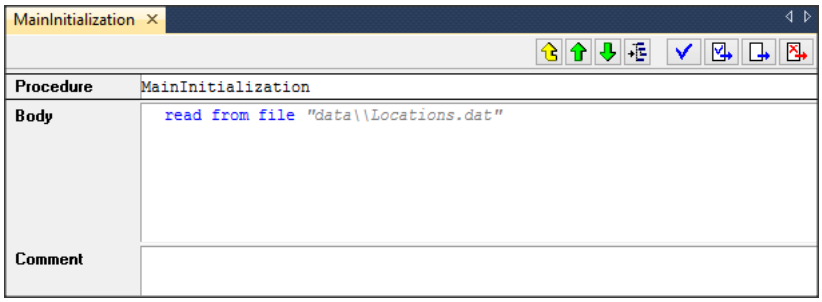


Figure 5.7: The completed attribute form of the MainInitialization procedure

To execute the MainInitialization procedure without having to reopen the project, you can: *Run procedure*

- select the MainInitialization procedure in the model tree, and
- use the right mouse pop-up menu to issue the **Run Procedure** command (see Figure 5.8).

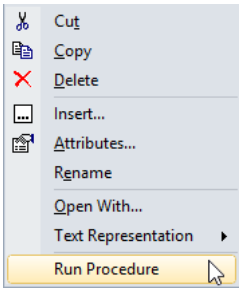
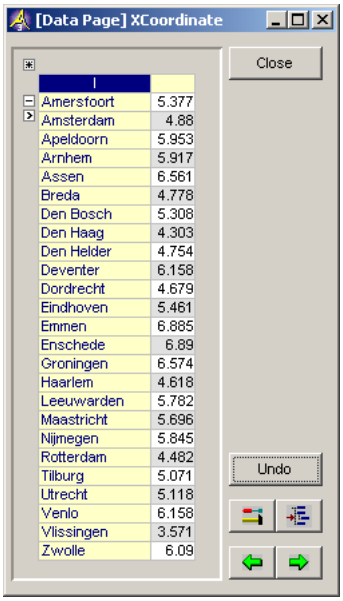


Figure 5.8: A right mouse pop-up menu

Once AIMMS has read the data file, all model identifiers are initialized. You can look at the current data values by opening one or more data pages. For instance, to open a data page for the identifier XCoordinate, you should perform the following actions: *Data pages*

- select the XCoordinate parameter in the model tree, and
- use the right mouse pop-up menu to issue the **Data...** command.

The data page that will appear is displayed in Figure 5.9. By pressing the **Left Arrow** button you will get the data page for the set of locations, while pressing the **Right Arrow** button will lead to the parameter YCoordinate.



I	
Amersfoort	5.377
Amsterdam	4.88
Apeldoorn	5.953
Arnhem	5.917
Assen	6.561
Breda	4.778
Den Bosch	5.308
Den Haag	4.303
Den Helder	4.754
Deventer	6.158
Dordrecht	4.679
Eindhoven	5.461
Emmen	6.885
Enschede	6.89
Groningen	6.574
Haarlem	4.618
Leeuwarden	5.782
Maastricht	5.696
Nijmegen	5.845
Rotterdam	4.482
Tilburg	5.071
Utrecht	5.118
Venlo	6.158
Vlissingen	3.571
Zwolle	6.09

Figure 5.9: The data page for the parameter `XCoordinate`

5.4 A first page

Pages with objects

To illustrate some of AIMMS's graphical features, we can now make a page containing a network object displaying the locations geographically on a map. AIMMS uses the concept of pages to display data objects in the form of tables and graphs.

Using the Page Manager

To create a new empty page you should execute the following steps:

- press the **Page Manager** button on the toolbar (or alternatively, use the *F9* key),
- press the **New Page** button on the toolbar to create a page,
- specify 'Locations' as the name of this new page, and
- press the *Enter* key to register the page.

The **Page Manager** with the new page is shown in Figure 5.10.

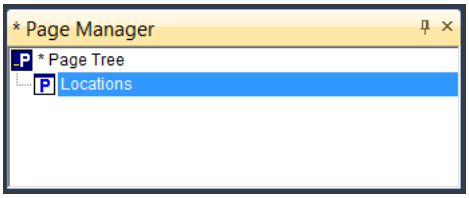


Figure 5.10: the **Page Manager** with a single page

Two important page modes

Two important page modes are the **Edit** mode and the **User** mode. The **Edit** mode is used for creating and modifying the objects on a page. The **User** mode is for viewing and editing the data displayed within objects on a page.

Opening the page

To open this new page in **Edit** mode:

- select the *Locations* page in the **Page Manager**, and
- press the **Edit Mode** button on the toolbar to open the selected page in **Edit** mode.

Drawing a new network object

To create a new network object, perform the following actions:

- press the **New Network Object** button on the toolbar,
- position the mouse cursor where you like the upper left corner of the new object to be,
- press the left mouse button and drag the mouse cursor to a point on your screen such that the resulting rectangle has a height-width ratio of approximately 2, and
- release the mouse button.

Network object identifiers

The **Network Object** dialog box will appear. Please use the three **Wizard** buttons on the dialog box to fill in the 'Node index', 'X coordinate' and 'Y coordinate' fields according to Figure 5.11. Note that in the 'Node Index' field you need to enter the character 'l' and not the number '1'.

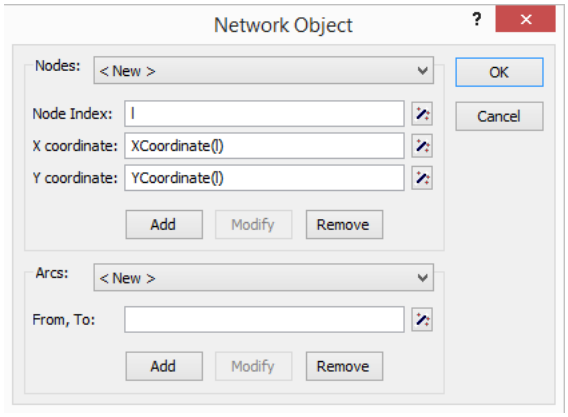


Figure 5.11: The **Network Object** dialog box

Initial network object

After you have pressed the **OK** button, the network object created at this point should look like the one in Figure 5.12. By adding the appropriate background bitmap, the locations will become more meaningful.

Figure 5.12: The initial **Network Object**

Network bitmap

To furnish the network object with a background bitmap, you need to change its properties. To do so, you should perform the following actions:

- press the **Properties** button on the toolbar to access the **Properties** dialog box,
- select the **Background** tab,
- click on the "No Image" at the right of **Background** property, press button and select **From File** command from the popup menu,
- click on the value field of the **Image File Name**, press the button, select the **Select File Name**... command from the popup menu, and select the filename '`Bitmaps\Netherlands.bmp`',
- position the picture by entering 3.3 in the 'Left' edit field, 7.3 in the 'Right' edit field, 53.5 in the 'Top' edit field, and 50.7 in the 'Bottom' edit field,
- press the **Apply** button, but do not press the **OK** button yet.

Figure 5.13 shows the network object with the background bitmap.

Figure 5.13: The intermediate **Network Object**

Positioning the bitmap

The four values you just entered, position the bitmap to match the locations. These values reflect the longitude and latitude coordinates of the boundaries of the bitmap. Even though the bitmap and the locations are now consistent, the bitmap is not yet consistent with the size of the rectangle. The coordinates of the rectangle must be made consistent with the coordinates of the bitmap.

Network area

In a professional application one would typically use model identifiers to adapt the size of the rectangle, thereby controlling the zoom and scroll behavior of the network object. In this chapter the coordinates of the rectangle are set equal to the coordinates of the bitmap resulting in a tight match. To complete the layout of the network object you should do the following:

- select the **Network** tab,
- fill in the four edit fields as in Figure 5.14.
- uncheck all checkboxes, and
- press the **OK** button.

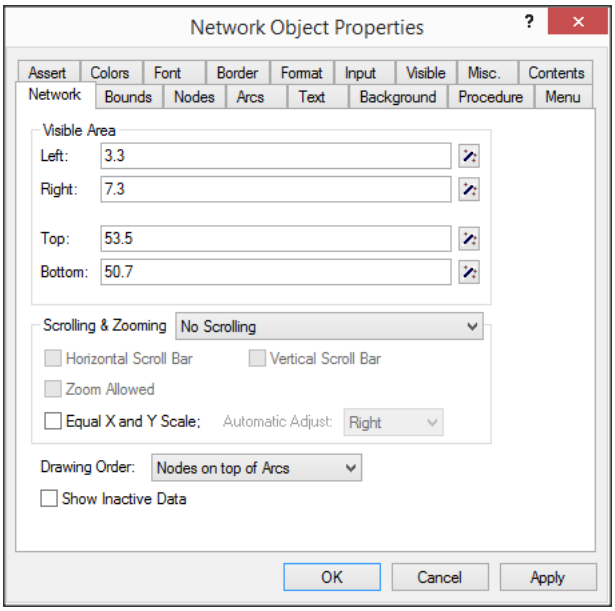


Figure 5.14: The **Network Properties** dialog box

Saving your changes

The asterisk on the left of the tab title in the page indicates that the additions to your page have not yet been saved to disk. To save your work, press the **Save Project** button on the toolbar.

View in **User** *mode*

You are now ready to change the page to user mode by pressing the **Page User Mode** button in the page toolbar. Your final network object should now look like the one in Figure 5.15. Note that the names of the cities are not part of the bitmap, but are superimposed based on the contents of the node set.

Figure 5.15: The final **Network Object**

Chapter 6

Quantities and Time

6.1 Model Structure

Initial tree

The predefined initial model tree is primarily to help students build small models with a single fixed data set. All model declarations can be placed in the single declaration section, the initial data can be entered in the initialization procedure, and the instruction to solve a mathematical program can be placed inside the execution procedure. In this more extensive tutorial you will be asked to structure the entire model tree.

Creating two new sections ...

Whenever you are building an extensive model, it is worthwhile using sections. With sections, you can organize the model in such a way that it is easy to locate relevant portions of your model. Proper organization will also help you and your co-workers maintain the model during its lifespan. In this tutorial, the model representation contains two main model sections: one model section for the overall model to be developed in Parts 4 and 5, and one model section for the user interface to be considered in Part 6. Each of these model sections will, in turn, be subdivided into several subsections to reflect additional structure. In this chapter, the first main model section will be subdivided. To create the two main model sections, you should take the following actions:

- select the root node `Main Softdrink Planning` in the model tree,
- press the **New Section** button  on the toolbar to create a section node in the model tree,
- specify 'The Model' as the name of the section, and press the *Enter* key to register the name,
- once more press the **New Section** button  on the toolbar to create the second section node,
- specify 'The User Interface' as its name, and once more press the *Enter* key.

... and several subsections

The first main section will be subdivided into six smaller subsections. First you need to double-click on the book icon . to open this section. After opening the section, the book icon will be an open book . If, by any chance, you double-clicked on the name of the book section instead of its book icon, you will be in the attribute form of the section. If so, just close that form, and then make sure that you double-click on the book icon. You can now create subsections in exactly the same way as you created the two main sections. At this point you should create a structure of subsections identical to the one in Figure 6.1.

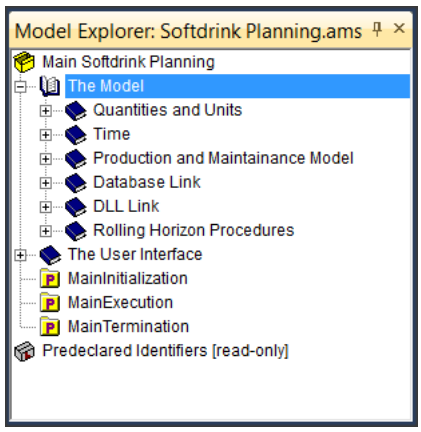


Figure 6.1: The structure of the section `The Model`

6.2 Entering quantity declarations

Creating a declaration section

With the above overall section structure in place, you are ready to specify the first declaration section below the section entitled `Quantities and Units`. To create the declaration section you should take the following actions:

- open the model section `Quantities and Units` by double-clicking on the corresponding book icon ,
- press the **New Declaration** button to create a new declaration section,
- enter 'Quantity Declarations' as the name of this new declaration section, and
- press the *Enter* key to register the name.

Units first

While developing an application, it is not unusual to begin with the declaration of quantities and units. After all, you will need the units later when you complete the declarations of the parameters and variables in your model.

Volume quantity and its base unit

In Chapter 3, volumes were expressed in terms of hectoliters and truckloads. In AIMMS, you first need to declare a volume quantity. Volume is a standard SI quantity (i.e. part of the International System of Units), and is present in the AIMMS SI unit base. The name of the base unit is 'm3', and the units 'hl' (hectoliter) and 'TL' (truckload) are then expressed in terms of this unit.

Declaring the volume quantity

To declare the volume quantity, you should perform the following actions:

- open the declaration section `Quantity Declarations` by double-clicking on the scroll icon ,
- press the **Other**... button on the toolbar (or alternatively, press the *Insert* key),
- select the quantity type in the **Select Type of Identifier** dialog box, and press the **OK** button,
- follow the instruction 'Press enter to select a SI Quantity' in order to choose from a list of predefined SI quantities,
- select the 'SI_Volume' quantity, and press the **OK** button,
- select the second option 'm3' as in Figure 6.2, and
- press the **OK** button.

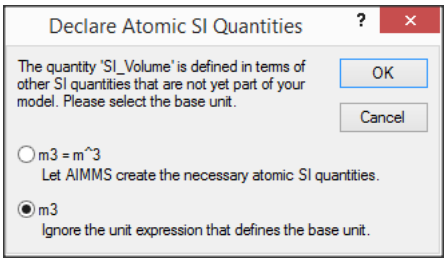


Figure 6.2: The **Ignore Unit Expression** dialog box

Specifying its attributes

You can now open the attribute form of the quantity `SI_Volume` in order to enter the unit conversion factors for the units [hl] and [TL]. The initial attribute form of the quantity `SI_Volume` is shown in Figure 6.3.

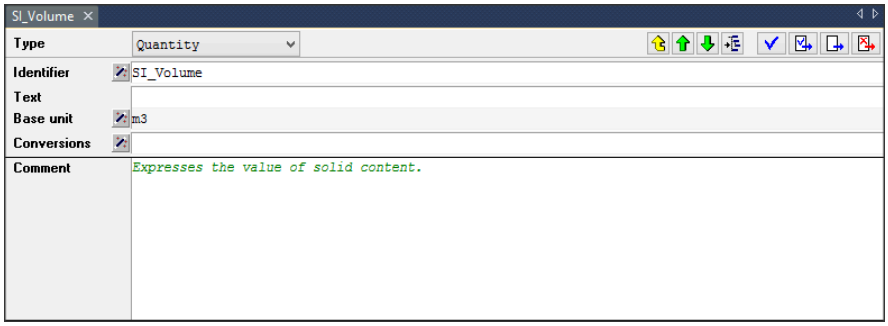


Figure 6.3: The initial attribute form of the quantity `SI_Volume`

Specifying the unit conversion of [hl]

To specify the first unit [hl] (hectoliter), you should perform the following actions:

- open the attribute form of the quantity `SI_Volume` as discussed in the previous paragraph,
- press the **Wizard** button for the **Conversions** attribute,
- select 'l' (which stands for liters) from the 'Derived Units' listbox,
- select 'hecto' from the 'Decimal Scaling' listbox, and
- press the **Transfer** button to accept the definition of the new unit 'hl'.

The initial selection of the derived unit 'l' and the corresponding decimal scaling 'hecto' are shown in Figure 6.4.

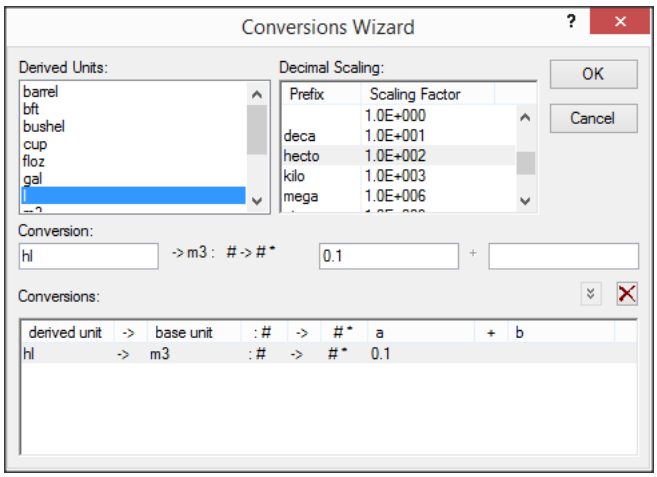


Figure 6.4: The selections in the **Conversions** Wizard

Specifying the unit conversion of [TL]

You are now ready to enter the second unit [TL] (truckload), which was given as 12 cubic meters. Note that [TL] is a self-made unit, and that the two listboxes in Figure 6.4 do not support you in this instance. Execute the following steps:

- consider the edit field under the heading 'Conversion' (containing 'hl'), and change its contents to the letters 'TL' (without quotes),
- consider the edit field to the right (containing '0.1'), and change it to the number '12' (without quotes),
- as before, press the **Transfer** button button to accept the definition of the new unit 'TL', and
- press the **OK** button to complete the specification of the two derived units [hl] and [TL].

The attribute form should now be as shown in Figure 6.5. By pressing the **Check, commit and close** button , you can verify whether AIMMS accepts the attribute form as completed by you. If there are no errors, AIMMS will commit its contents and close the attribute form.

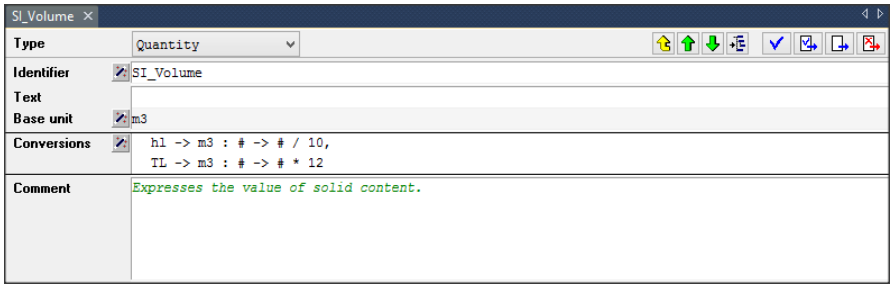


Figure 6.5: The completed attribute form of the quantity SI_Volume

Specifying the currency quantity

To be able to express amounts of money, you need to declare a currency quantity. Currency is not a standard SI quantity, and needs to be specified. In this tutorial you will only use a single base unit '$' without any conversions to other currencies. To declare the currency quantity you should perform the following actions:

- declare a quantity Currency,
- enter '$' (without the quotes) as its **Base Unit** attribute, and
- press the **Check, commit and close** button .

Specifying the time quantity

The final quantity to be introduced is the SI quantity SI_Time_Duration. By default, the base unit of this quantity is set to 's' (seconds). However, the base unit 'day' is more natural for this model. Use the base **Base Unit** wizard on the attribute form to change the base unit from 's' to 'day'. When AIMMS asks you whether you want to retain the data, select 'No'.

Week-to-day conversion

In addition to the base unit 'day' please use the **Conversions** wizard to specify the conversion between 'day' and 'week'. The resulting attribute form is shown in Figure 6.6.

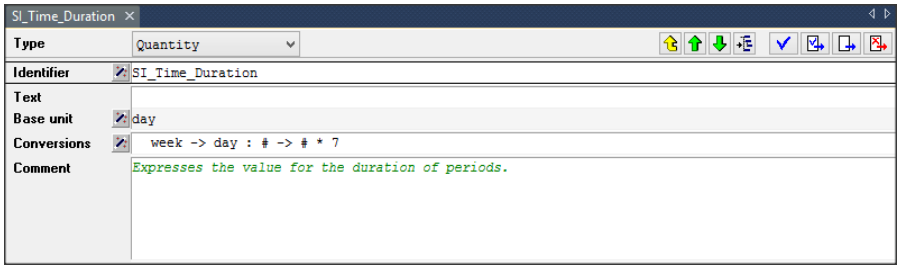


Figure 6.6: The completed attribute form for the quantity `SI_Time_Duration`

Your tree thus far

The model tree so far is shown in Figure 6.7.

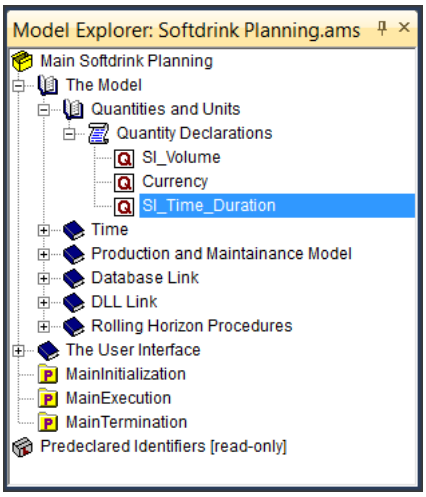


Figure 6.7: The intermediate model tree showing all quantity identifiers

Saving your changes

Again, the asterisk on the left of the model node of the **Model Explorer** indicates that additions to your project have not yet been saved to disk. To save your work, please press the **Save Project** button on the leftmost position on the toolbar.

6.3 Entering time declarations

Special data types

AIMMS offers two special identifier types for time-based modeling applications, namely *calendar* and *horizon*. Calendars and horizons are sets with special features for dealing with time. In this tutorial, both identifier types will be used, and they will be linked through the use of a special indexed set referred to as a *timetable*.

Advanced concepts

Experience with the tutorial has shown that it may take more than one reading of the following paragraphs before one obtains a clear understanding of the advanced concepts presented.

Calendars

A *calendar* is defined as a set of consecutive time slots of unit length covering the complete time frame from the calendar's beginning date to its end date. You can use a calendar to index data defined in terms of calendar time. In this tutorial both a daily and a weekly calendar will be introduced.

Horizons

A *horizon* is basically a set of abstract planning periods to be used inside a mathematical program. The elements in a horizon are divided into three groups, also referred to as time blocks. The main group of elements comprise the *planning interval*. Periods prior to the planning interval form the *past*, while periods following the planning interval form the *beyond*. When variables and constraints are indexed over a horizon, AIMMS automatically restricts the generation of these constraints and variables to periods within the planning interval.

Timetables

A *timetable* is either an indexed set or an indexed element parameter that links model periods in a horizon to time slots in a calendar. Based on a timetable, AIMMS provides functions that let you *aggregate* calendar data into horizon data. Similarly, there are functions to let you *disaggregate* horizon data into calendar data. Figure 6.8 illustrates an example of a timetable linking a horizon and calendar.

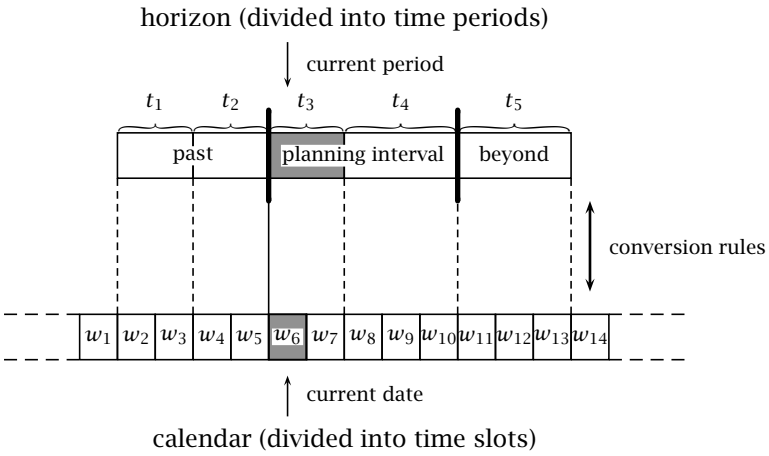


Figure 6.8: Linking a calendar and horizon

The actual timetable corresponding to the example that is shown in Figure 6.8 is shown in Figure 6.9. In this example the timetable is called `TimeslotToPeriod`.

`TimeslotToPeriod`	
`TimeslotToPeriod`(t_1)	$\{w_2, w_3\}$
`TimeslotToPeriod`(t_2)	$\{w_4, w_5\}$
`TimeslotToPeriod`(t_3)	$\{w_6, w_7\}$
`TimeslotToPeriod`(t_4)	$\{w_8, w_9, w_{10}\}$
`TimeslotToPeriod`(t_5)	$\{w_{11}, w_{12}, w_{13}\}$

Figure 6.9: A timetable corresponding to Figure 6.8

Two declaration sections

To group the time-related identifiers in this tutorial you are asked to create two separate declaration subsections within the `Time` model section. Please execute the following actions:

- in the model tree, open the section node `Time`,
- create a new declaration section `Period Declarations`, and
- create a new declaration section `Calendar Declarations`.

6.3.1 Horizon-related declarations

Creating the first parameter `NumberOfPeriods`

To declare the first parameter `NumberOfPeriods` in the section `Period Declarations`, you should execute the following actions:

- open the declaration section `Period Declarations`,
- press the **New Parameter** button on the toolbar to create a new parameter in the model tree,

- specify 'NumberOfPeriods' as the name of this parameter, and
- press the *Enter* key to register the name.

Parameter attributes

To complete the declaration of the parameter NumberOfPeriods you should open its attribute form and perform the following actions:

- enter the integer range '{1..inf}' (without the quotes) as the **Range** attribute,
- select the 'Initial Data' radio button in front of the **Definition/Initial Data** attribute,
- enter the number '10' (without the quotes) as the **Initial data** attribute, and
- press the **Check, commit and close** button to commit your edits.

Note that integer ranges in AIMMS are always enclosed by curly brackets. The square brackets are reserved to represent continuous ranges.

Range checking

The existence of a **Range** attribute enables AIMMS to perform range checking during execution. Since the integer set '{1..inf}' represents the set of all strictly positive integers, AIMMS will report an error when a non-integer, or a value less than one, is assigned to the parameter NumberOfPeriods.

Creating the second parameter

The second parameter NumberOfPeriodsInPlanningInterval can now be declared in a similar fashion. Again, specify '{1..inf}' as the **Range** attribute. Enter '8' (without the quotes) as its **Initial data** attribute.

Declaring a horizon

To declare the horizon, you need to execute the following steps:

- press the **Other**... button on the toolbar,
- select the horizon type H, and press the **OK** button,
- specify 'Periods' as the name, and
- press the *Enter* key to register the name.

Entering the first attributes

Next, open its attribute form and enter both the index and the current period attributes:

- press the *Enter* key again to open the attribute form of Periods,
- position the cursor in the empty edit field next to the **Index** attribute, and type the letter 't' (without quotes), and similarly,
- type 'period-01' (with the quotes) as the **Current period** attribute.

Specifying the **Interval length** *attribute*

Next, consider the **Interval length** attribute. You can use the convenient *name completion* facility in AIMMS to avoid re-typing long identifier names.

- type only the first letter 'N' in the edit field next to the **Interval length** attribute
- use the *Ctrl-Spacebar* key combination to let AIMMS provide you with the list of all identifiers and let AIMMS select the first possible extension of the letter 'N' (see Figure 6.10), and
- select 'NumberOfPeriodsInPlanningInterval' as the identifier name, and press *Enter*.

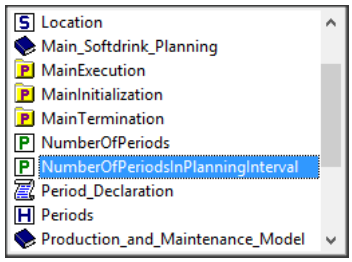


Figure 6.10: The **Name Completion** pop-up menu

Specifying the **Definition** *attribute*

Consider Figure 6.11, and complete the **Definition** attribute. Again, you may want to use the name completion facility to select `NumberOfPeriods` as the second argument in the function `ElementRange`.

- type the definition as in Figure 6.11,
- press the **Check, commit and close** button to commit all your edits.

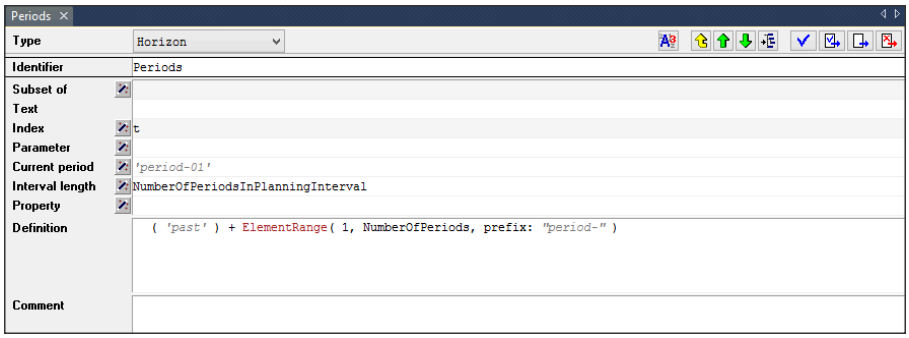


Figure 6.11: The completed attributes of the horizon 'Periods'

Name completion and dragging

The name completion facility can be used to complete any incomplete identifier name. In addition to name completion, you can also drag an identifier name from the model tree to any edit field in your application. Both facilities are there to avoid typing errors, guarantee name consistency and speed up your work.

The function `ElementRange`

The `ElementRange` function allows you to *dynamically* create or change the contents of a set based on integer values. In this tutorial, the elements are 'period-01', 'period-02', etc., up to the value of the parameter `NumberOfPeriods`. The first two arguments are mandatory, and *may* be preceded by their formal argument names 'from' and 'to'. The remaining arguments are optional, and *must* be preceded by their formal argument names when used in a non-default order.

... its arguments

After typing a function name, as soon as you enter the opening bracket (or when you hover with the mouse pointer over the function name), Aimms will pop up a quick info tip window as illustrated in Figure 6.12. This info tip window displays information about the arguments of the `ElementRange` function. The information will remain visible until you enter a closing bracket (or use the mouse to position the cursor outside the argument list).

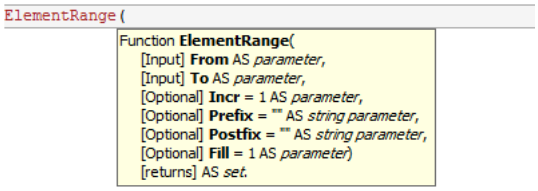


Figure 6.12: The quick info tip window of the 'ElementRange' function

... and its documentation

In AIMMS, you can quickly access information on the type and order of the arguments of a function and/or its documentation from a help file. You can open *The Function Reference* from within AIMMS by performing the following actions:

- use the mouse cursor to position the text cursor on the `ElementRange` keyword,
- use the right-mouse pop-up menu to issue the **Help on** command, and
- select the `ElementRange` entry in the **Help on** submenu (see Figure 6.13).

At this point, Acrobat's PDF viewer will open the *The Function Reference* on the appropriate page.

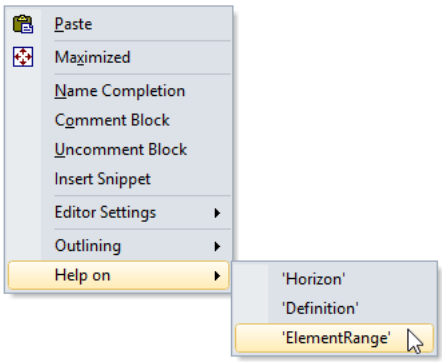


Figure 6.13: A right-mouse pop-up menu

Remaining period declarations

The four remaining period declarations concern three numerical parameters referencing the desired number of days in a period, the desired number of weeks in a period and the actual number of days in a period (reflecting weekends and official holidays), and a so-called *element parameter* denoting the first period in the planning interval. The value of this last element parameter is not a number, but an element of the set `Periods`.

Number of days in a period

The desired number of days in a period is equal to seven. Due to weekend days and official holidays the actual number of days per period will be less than this. To declare the parameter `DesiredNumberOfDaysInPeriod` you should perform the following actions:

- insert a new parameter immediately below the horizon `Periods`,
- specify 'DesiredNumberOfDaysInPeriod(t)' as the name of this new parameter, and press the *Enter* key,
- open its attribute form,
- enter the number '7' (without quotes) as the **Definition** attribute, and
- press the **Check, commit and close** button to commit all your edits.

Number of weeks in period

Because the parameter `DesiredNumberOfWeeksInPeriod` is very similar to the parameter `DesiredNumberOfDaysInPeriod` it is possible to create this identifier declaration from copy of the parameter `DesiredNumberOfDaysInPeriod`. To do so you should execute the following steps:

- select the identifier `DesiredNumberOfDaysInPeriod` in the model tree,
- press the **Copy** button on the toolbar (or alternatively, press the *Ctrl-C* key combination),
- press the **Paste** button on the toolbar (or alternatively, press the *Ctrl-V* key combination),
- press the *F2* key and change the name from `Copy_DesiredNumberOfDaysInPeriods(t)` to `DesiredNumberOfWeeksInPeriod(t)`,
- press the *Enter* key to confirm the name change,

- press the *Enter* key to open its attribute form,
- change the number '7' in the **Definition** attribute to '1' (without the quotes), and
- press the **Check, commit and close** button to commit all your edits.

Name change propagation

Changing the name of an identifier in the model tree will cause AIMMS to change all references to the identifier accordingly.

Declaring the actual period length

To declare the indexed parameter `ActualNumberOfDaysInPeriod(t)`, expressed in terms of days, you should execute the following steps:

- insert a new parameter,
- specify 'ActualNumberOfDaysInPeriod(t)' as the name of this new parameter, and press the *Enter* key,
- open its attribute form, and press the **Wizard** button for the **Unit** attribute,
- select 'SI_Time_Duration' as the quantity and 'day' as the unit,
- press the **OK** button,
- enter the unquoted sentence 'takes into account the weekends and the official holidays' as the **Comment** attribute, and
- press the **Check, commit and close** button to commit all your edits.

The completed attribute form is shown in Figure 6.14.

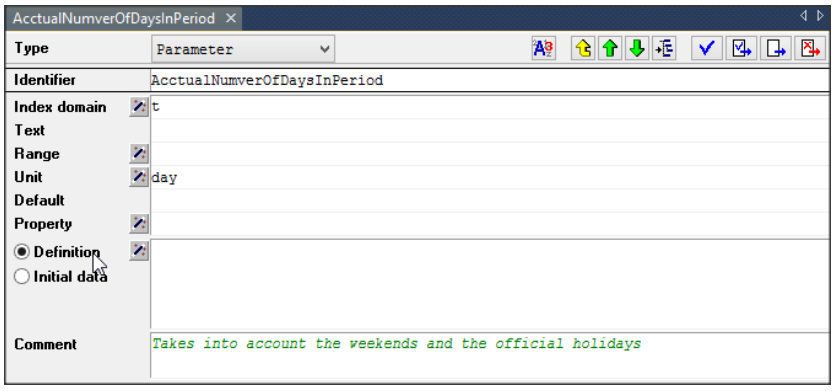


Figure 6.14: The attribute form of the parameter `ActualNumberOfDaysInPeriod`

Declaring a parameter for the first period ...

By declaring a separate element parameter for the first period in the planning interval, instead of simply using the element 'period-1', you promote the important separation between model and data. Please execute the following declaration steps:

- press the **New...** button on the toolbar,
- select the element parameter type , and press the **OK** button,

- specify 'FirstPeriodInPlanningInterval' as the name of the element parameter, and press the *Enter* key to register this name.

... and completing its attributes

The following actions complete the corresponding attribute form:

- press the *Enter* key again to open the attribute form,
- use the **Range** wizard to specify the set `Periods` as the range,
- specify 'first(t | t in Periods.Planning)' (without the quotes) as its definition, and
- press the **Check, commit and close** button to commit all your edits.

The model tree

The model tree up to this point is shown in Figure 6.15.

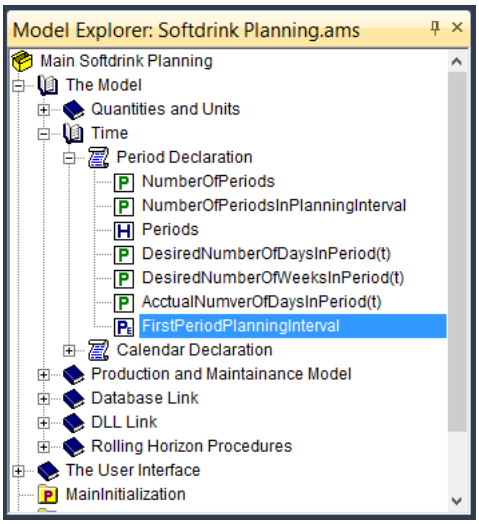


Figure 6.15: All period declarations in the model tree

6.3.2 Calendar-related declarations

Declaring begin and end dates

Two string parameters are introduced to allow you to change the beginning and end dates of both calendars in your model in a dynamic fashion. This is again an example of the separation between model and data. To declare the first of these two string parameters, you should execute the following actions:

- open the `Calendar Declarations` declaration section,
- press the **Other**... button on the toolbar,
- select the string parameter type , and press the **OK** button,
- specify 'BeginDateOfCalendar' as the name of the string parameter, and press the *Enter* key to register this name.

Repeat the last three steps to declare EndDateOfCalendar as the second string parameter.

... completing their attributes

The attribute forms can now be completed as follows:

- select the string parameter BeginDateOfCalendar,
- press the *Enter* key to open its attribute form,
- specify the string "2000-07-01" (don't forget the quotes) as the definition of the beginning date, and
- press the *Ctrl-Enter* key combination as an alternative for the **Check, commit and close** button to commit all your edits.

Repeat these steps for the string parameter EndDateOfCalendar, but use the quoted string "2001-06-30" as its definition. This date format (yyyy-mm-dd), used to represent the beginning and end dates above, is required by AIMMS. The date format of the timeslots in the calendar can be customized to your specification using the **Timeslot format** attribute.

Declaring a calendar

To declare the calendar Days, execute the following steps:

- press the **Other**... button on the toolbar,
- select the calendar type , and press the **OK** button,
- specify 'Days' as the name, and
- press the *Enter* key to register the name.

Specifying the calendar attributes

By now, you should be able to open the attribute form of the calendar and use the wizards to complete the attribute fields as shown in Figure 6.16. When completing the **Begin date** and **End date** attributes, choose the **Select String Parameter**... command from the pop-up menu and select the appropriate string parameter.

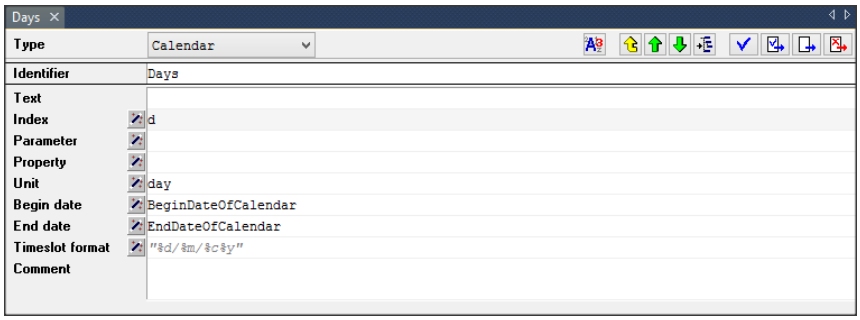


Figure 6.16: The completed attribute form of the calendar 'Days'

Timeslot format

When completing the **Timeslot format** attribute using the wizard you should select the **Select Static String**... command from the pop-up menu. AIMMS will then open a **Timeslot format** wizard to support you in constructing the appropriate timeslot format. Through this wizard, you can not only select from a number of 'Basic Formats' (including the ones defined by the regional settings of your computer), but you also have the possibility of constructing a custom format, observing the result as you proceed. The timeslot format selected in this tutorial is shown in Figure 6.17.

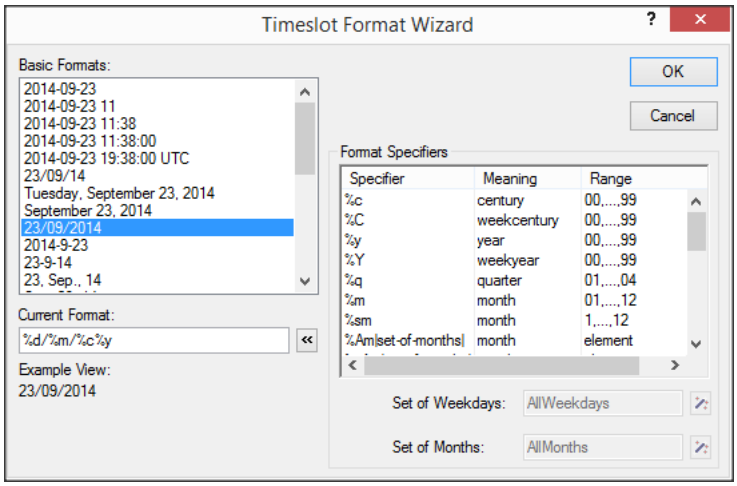


Figure 6.17: The **Timeslot Format** wizard

Declaring subsets ...

Several subsets of the calendar Days will be used throughout the model in this tutorial, and you should be able to enter these sets on the basis of what you have learned so far. Note that, when declaring these subsets, the use of the **Subset of wizard** (see Figure 6.18) is mandatory and you are not allowed to complete the attribute by hand.

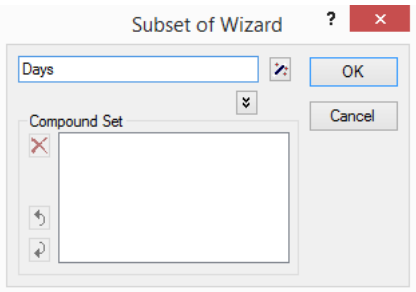


Figure 6.18: The **Subset of** wizard

... of the calendar Days

The names of the subsets are self-explanatory. The subset Mondays will play a role later on when a timetable is constructed to link the horizon Periods and the calendar Days. This subset is used as a function argument, and AIMMS will then begin a new period in the horizon whenever it encounters a Monday. The five subsets to be entered by you in the Calendar Declarations section are as follows:

```
Set WeekendDays {
    SubsetOf    : Days;
    Definition  : {
        { d | TimeslotCharacteristic( d, 'weekday') > 5}
    }
}

Set OfficialHolidays {
    SubsetOf    : Days;
}

Set InactiveDays {
    SubsetOf    : Days;
    Definition  : WeekendDays + OfficialHolidays;
}

Set Mondays {
    SubsetOf    : Days;
    Definition  : {
        { d | TimeslotCharacteristic( d, 'weekday') = 1}
    }
}

Set DaysInPeriod {
    IndexDomain : t;
    SubsetOf    : Days;
}
```

Timeslot characteristics

The predefined function TimeslotCharacteristic determines a numeric value which characterizes the timeslot in terms of its day in the week, its day in the year, etc. In the **Definition** attribute of the set WeekendDays, all days in the week with their numeric value greater than 5 (as weekend days) are selected. Similarly, in the **Definition** attribute of the set Mondays ,this function selects all Mondays (with the numeric value of 1) to be used as delimiter days.

Viewing the weekend days

At this moment the daily calendar is fully defined since the beginning date and end dates are defined as string constants. Similarly, the subset WeekendDays is also fully defined, and its contents can already be viewed as follows:

- select the set WeekendDays in the model tree, and
- select the **Data**... command in the right-mouse pop-up menu (see Figure 6.19).

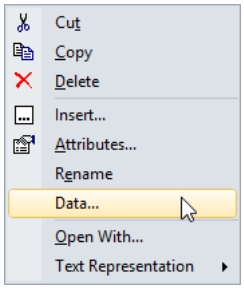


Figure 6.19: A right-mouse pop-up menu

Data page

AIMMS will now display the corresponding data page as shown in Figure 6.20. On the left you see the elements of the set `WeekendDays`. On the right you see these same elements, but then as a subset of the calendar `Days`.

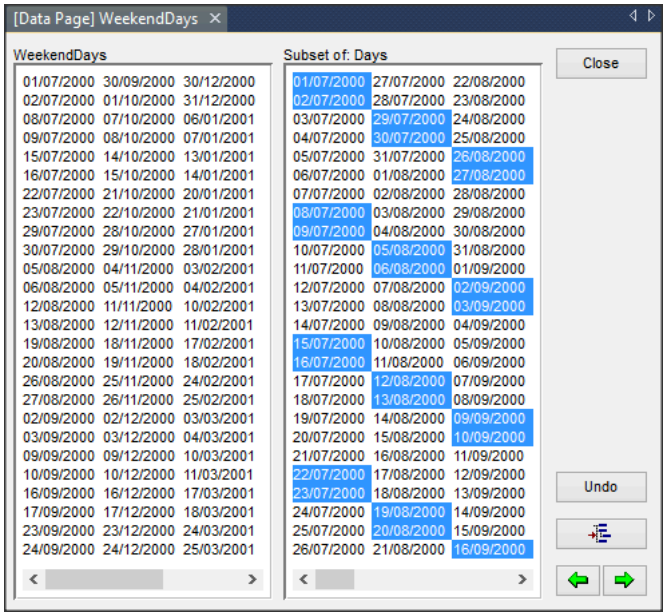


Figure 6.20: Data page for the set `WeekendDays`

Creating a weekly calendar

In addition to the daily calendar, there is also a weekly calendar together with several subsets thereof. You should be able to declare this calendar, called `Weeks`, based on what you have learned so far. We recommend that you specify the **Timeslot format** attribute by hand, because the corresponding format is not predefined. The completed attribute form of `Weeks` is shown in Figure 6.21.

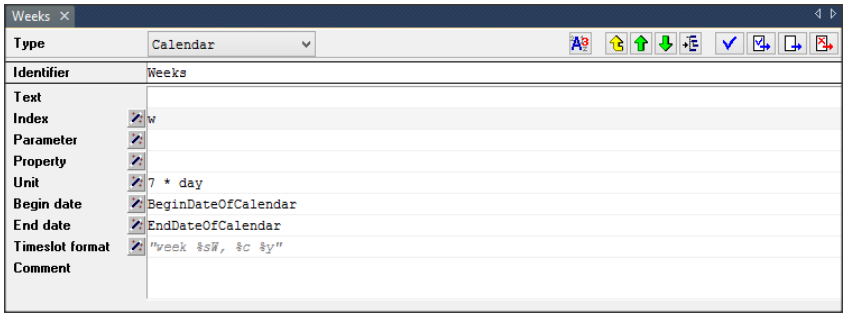


Figure 6.21: The completed attribute form of the calendar 'Weeks'

At this moment if you ask data of `Weeks` calendar, you will get a warning explaining that a weekly calendar for which the start date is not the first day of a week (Monday) is limited in its use. Since the limitations are no issue in this tutorial project and to prevent this warning to pop up again, please switch `off` the option `Warning calendar week begin` that causes this warning, by executing the following actions:

- go to the **Settings** menu and execute the **Project Options** command,
- select the **AIMMS** - **Progress, errors & warnings** - **Warnings** - **Compilation** folder in the option tree (see Figure 6.22),
- click on the Option **Warning calendar week begin** in the rightmost window,
- select on 'Off' value,
- press the **Apply** button on the **AIMMS Options** dialog box, and
- finish by pressing the **OK** button.

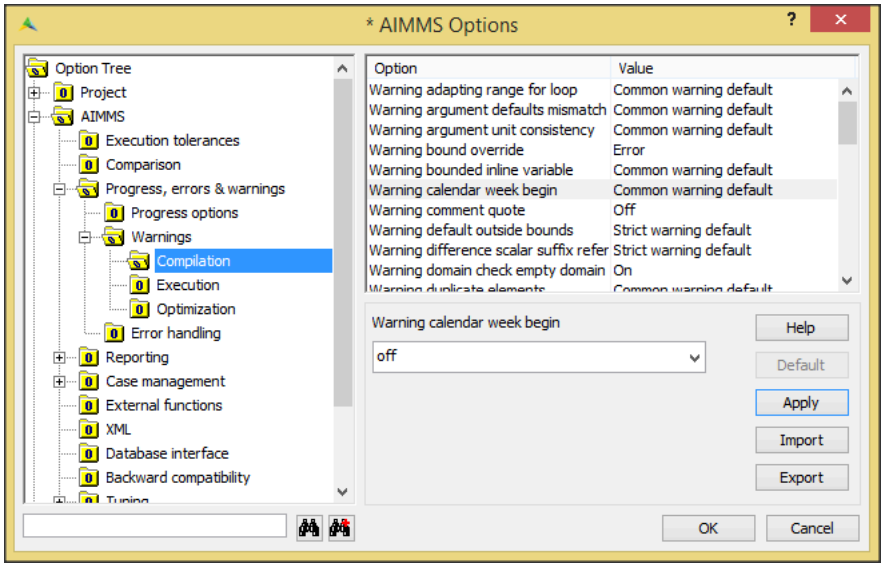


Figure 6.22: The **AIMMS Options** dialog box

Declaring week-related references

As indicated in the previous paragraph, you should have little or no problem entering the following subset and element parameter related to the calendar called `Weeks`.

```
Set InactiveWeeks {
    SubsetOf    : Weeks;
}

ElementParameter WeekInPeriod {
    IndexDomain : t;
    Range       : Weeks;
}
```

Relating days to weeks

The relationship between days and weeks can be captured through an indexed element parameter that contains, for each day in the daily calendar, the corresponding week in the weekly calendar. Please enter the following declaration:

```
ElementParameter DayToWeek {
    IndexDomain : d;
    Range       : Weeks;
    Definition  : {
        first ( w | TimeslotCharacteristic(w, 'week') =
        TimeslotCharacteristic(d, 'week')
        and
        TimeslotCHaracteristic(w, 'year') =
        Timeslotcharacteristic(d, 'year') )
    }
}
```

With the use of the function `TimeslotCharacteristic` it becomes straightforward to verify whether the week number (ranging from 1 to 53) of a week `w` is equal to the week number of a day `d`. The year number can be checked in a similar fashion.

One more identifier

The following calendar-related identifier will be used later. Please enter it now.

```
ElementParameter LastWeekInClaendar {
    Range      : Weeks;
    Definition : last(Weeks);
}
```

Model tree

The part of the model tree containing the calendar declarations is shown in Figure 6.23.

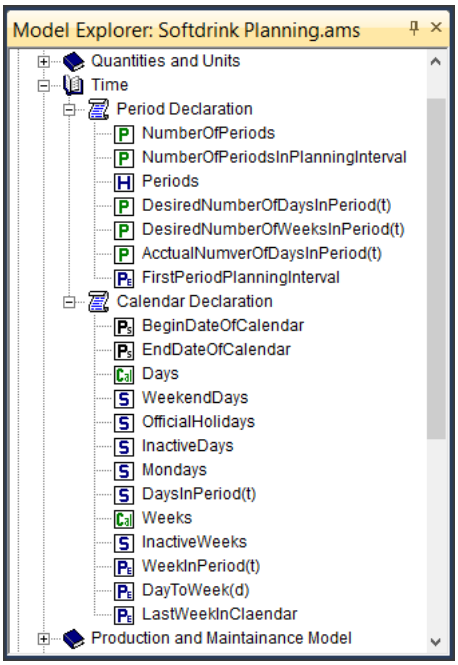


Figure 6.23: All calendar related declarations in the model tree

Chapter 7

Production and Maintenance Model

This chapter

In this chapter you will enter all the model identifiers that are related to the mathematical model described in Chapter 3. As with most realistic models, the number of identifiers is quite large, and it pays to refine the model tree by declaring several additional declaration sections.

7.1 Model structure

Adding to the model tree

Please add the following declaration subsections to the section named `Production and Maintenance Model`:

- `Location Declarations`
- `Scenario Declarations`
- `Production Declarations`
- `Supply and Demand Declarations`
- `Maintenance and Vacation Declarations`
- `Cost Declarations`
- `Mathematical Program Declarations`

The resulting section in the model tree is shown in Figure 7.1.

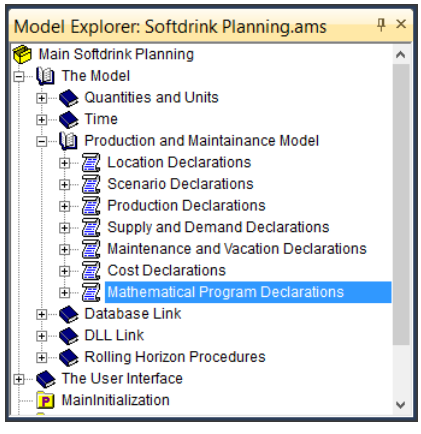


Figure 7.1: Seven declaration sections to increase model structure

Proceed with little assistance ...

Most of the declarations in this chapter are provided in a compact textual format that closely corresponds to the attribute format presented in previous chapters. Instead of providing detailed instructions, you are asked to complete the attribute forms on the basis of what you have learned so far.

... or import all identifier declarations

As explained in Chapter 4, you can avoid entering some, or all, of the declarations in this chapter by importing the model section 'Production and Maintenance Model.amb' into the Production and Maintenance Model section in the model tree. You are advised to at least examine the declarations listed in the remainder of this chapter if you choose the import file option.

7.2 Topology

Moving existing identifiers

In Chapter 5 you already declared the set of locations and their corresponding x- and y-coordinates. You should now move these existing identifiers to their new position in the model tree by performing the following actions:

- open the declaration section Declaration at the end of the model tree,
- select all three identifiers using the *Shift* key in combination with the left-mouse button,
- press the **Cut** button on the toolbar,
- select and open the section named Location Declarations, and
- press the **Paste** button on the toolbar.

As an alternative, you could have dragged the three identifiers to their new position.

Entering location declarations

In the Location Declarations section you can now declare the sets Factories and Centers as subsets of the set of all locations.

```
Set Factories {
    SubsetOf    : Locations;
    Index       : f;
}

Set Centers {
    SubsetOf    : Locations;
    Index       : c;
}
```

7.3 Demand scenarios

Entering scenario declarations

The following two identifiers need to be added to the section Scenario Declarations, and are used to set up the demand scenarios. Note that when a particular set has a fixed number of known elements, you can enter these elements as data in the **Definition** attribute (see the set Scenarios below).

```
Set Scenarios {
    Index       : s;
    Definition  : data { pessimistic, expected, optimistic };
}
Parameter ScenarioProbability {
    IndexDomain : s;
    Definition  : data { pessimistic : 0.25, expected : 0.50, optimistic : 0.25};
}
```

7.4 Production

Data definition and domain condition

An interesting feature of AIMMS is that you can specify a global index domain condition as illustrated in the last three declarations below. In these examples, AIMMS will only consider the (f, p) combinations that exist. All other combinations will be ignored throughout the application. Note that the '|' operator is to be interpreted as the 'such that' operator, and that the Ord(p) operator returns the ordinal position of the corresponding element p within its domain set ProductionLines.

Entering production declarations

Please open the Production Declarations subsection, and enter the following declarations after having read the previous paragraph.

```
Set ProductionLines {
    Index       : p;
    Definition: data { line-01 .. line-04 };
}

Parameter NumberOfProductionLines {
    IndexDomain : f;
}

Set FactoryProductionLines {
    IndexDomain : f;
    SubsetOf    : ProductionLines;
    Definition  : {
        { p | Ord(p) <= NumberOfProductionLines(f) }
    }
}

Parameter DeteriorationLevel {
    IndexDomain : (f,p) | p in FactoryProductionLines(f);
}

Parameter DeteriorationLevelAtStartOfCalendar {
    IndexDomain : (f,p) | p in FactoryProductionLines(f);
}

Parameter MaximumDeteriorationLevel {
    IndexDomain : (f,p) | p in FactoryProductionLines(f);
}
```

Entering unit expressions once

The **Unit** wizard can only complete the **Unit** attribute for you once either the desired unit or the unit expression has been entered. Therefore, when declaring the first of the two parameters below, you should enter the **Unit** attribute [hl/day] manually. When declaring the second parameter, you can use the **Unit** wizard and select the 'Implicit Quantities' entry.

```
Parameter ProductionLineLevelAtStartOfCalendar {
    IndexDomain  : (f,p) | p in FactoryProductionLines(f);
    Unit         : hl/day;
}

Parameter MaximumProductionLineLevel {
    IndexDomain  : (f,p) | p in FactoryProductionLines(f);
    Unit         : hl/day;
}
```

Postponing identifier checking

Once you have entered the declaration listed below, AIMMS still cannot compile the definition of the parameter PotentialProduction. This definition contains a reference to the three identifiers LineInMaintenance, DropDueToVacation and IsVacationPeriod, which have not yet been declared. In such a situation, you should use the **Commit and close** button instead of the **Check, commit and close** button, and AIMMS will not complain (though the identifier names will be colored red). Instructing AIMMS to compile the model will result in errors reporting missing identifiers. The three identifiers will be declared at a later stage.

```
Parameter PotentialProduction {
    IndexDomain  : (f,p,t) | p in FactoryProductionLines(f);
    Unit         : hl;
    Definition   : ActualNumberOfDaysInPeriod(t) *
                   (1- LineInMaintenance(f,p,t) )*
                   (1- DropDueToVacation * IsVacationPeriod(f,t) ) *
                   MaximumProductionLineLevel(f,p);
}
```

Initial overview

An overview of all the declarations entered by you so far is shown in Figure 7.2.

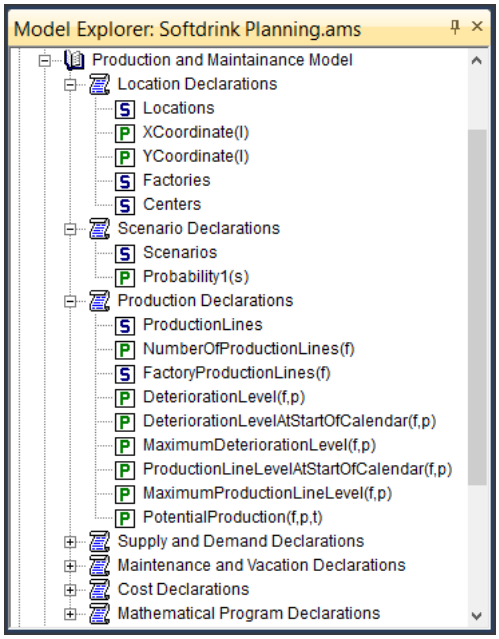


Figure 7.2: All location, scenario and production declarations

7.5 Supply and demand

Entering supply and demand declarations

Use the **Unit** wizard, the **Range** wizard, and the name completion functionality to enter the following supply and demand declarations in the appropriate declaration section of your model tree.

```
Parameter Demand {
    IndexDomain  : (c,t,s);
    Unit         : hl;
}

Parameter MinimumStock {
    IndexDomain  : l;
    Unit         : hl;
}

Parameter MaximumStock {
    IndexDomain  : l;
    Unit         : hl;
}
```

```
Parameter StockAtStartOfCalendar {
    IndexDomain  : l;
    Unit         : hl;
}

Parameter MaximumTransportCapacity {
    IndexDomain  : f;
    Unit         : TL;
}
```

7.6 Maintenance and vacations

Maintenance declaration

As was already mentioned in Chapter 2, most of the computations needed for maintenance planning can be placed outside the mathematical program. All you need to declare at this point is when a particular production line is undergoing maintenance. Please use the `Maintenance and Vacation Declarations` declaration section in your model tree.

```
Parameter LineInMaintenance {
    IndexDomain  : (f,p,t);
    Range        : binary;
}
```

Entering vacation declarations

The management of each factory knows the particular weeks in which a relatively large portion of its personnel will be on leave. During these weeks, production typically drops by 40% of the maximum production capacity. The following two parameters need to be declared.

```
Set VacationWeeks {
    IndexDomain  : f;
    SubsetOf     : Weeks;
}

Parameter DropDueToVacation {
    InitialData  : 0.4;
}

Parameter IsVacationPeriod {
    IndexDomain  : (f,t);
    Range        : binary;
    Definition   : {
        if ( WeekInPeriod(t) in VacationWeeks(f) )
            then 1
            else 0
        endif
    }
}
```

Compiling the entire model

At this point you should be able to compile the entire model, because the three identifiers missing in section 7.4 have now been declared. To compile the entire model you should execute the **Compile All** command from the **Run** menu. Alternatively, you could simply press the *F5* key. Please ignore any warnings concerning data initialization.

7.7 Costs

Entering cost declarations

The total scenario cost is divided into four components, each of which has its own unit cost declaration. The total cost is the weighted sum of the total scenario cost over all scenarios. You should enter the following declarations in the `Cost Declarations` section in your model tree.

```
Parameter UnitTransportCost {
    IndexDomain  : (f,c);
    Unit         : $/TL;
}

Parameter FixedCostDueToLeaveChange {
    Unit         : $;
    InitialData  : 25000;
}

Parameter UnitStockCost {
    IndexDomain  : l;
    Unit         : $/hl;
}

Parameter UnitProductionCost {
    IndexDomain  : f;
    Unit         : $/hl;
}
```

An overview of all supply and demand, maintenance and vacation, and cost declarations entered is shown in Figure 7.3.

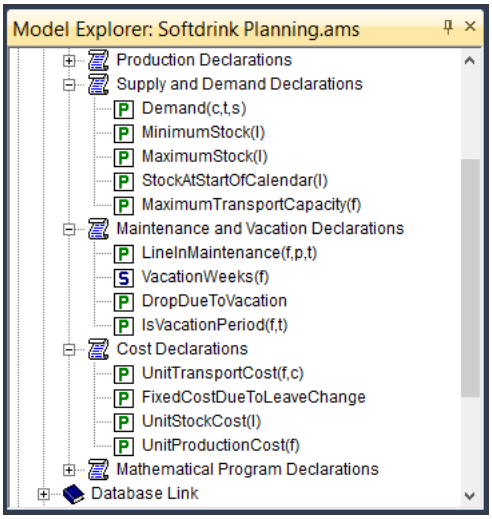


Figure 7.3: All supply, demand, maintenance, vacation, and cost declarations

7.8 Optimization model

Variables

There are seven variables in the formulation of the mathematical program in this tutorial. These should be entered in the declaration section `Mathematical Program Declarations` in your model tree.

```
Variable ProductionLineInUse {
    IndexDomain  : (f,p,t) | p in FactoryProductionLines(f);
    Range        : binary;
}

Variable Production {
    IndexDomain  : (f,t);
    Range        : nonnegative;
    Unit         : hl;
    Definition   : sum[ p, PotentialProduction(f,p,t) * ProductionLineInUse(f,p,t) ];
}

Variable ProductionLineLevelChange {
    IndexDomain  : (f,p,t) | p in FactoryProductionLines(f);
    Range        : [0, 1];
}

Variable Transport {
    IndexDomain  : (f,c,t,s) | UnitTransportCost(f,c);
    Range        : [0, MaximumStock(c)];
    Unit         : TL;
}
```

```
Variable Stock {
    IndexDomain  : (l,t,s);
    Range        : [MinimumStock(l), MaximumStock(l)];
    Unit         : hl;
    Definition   : {
     Stock(l,t-1,s) + Production(l,t) +
        sum[f, Transport(f,l,t,s) ] -
        sum[c,Transport(l,c,t,s) ] - Demand(l,t,s)
    }
}

Variable TotalScenarioCost {
    IndexDomain  : s;
    Range        : free;
    Unit         : $;
    Definition   : {
        sum[ (f,t,p), FixedCostDueToLeaveChange * ProductionLineLevelChange(f,p,t) ] +
        sum[ (f,t), UnitProductionCost(f) * Production(f,t) ] +
        sum[ (l,t), UnitStockCost(l) * Stock(l, t, s) ] +
        sum[ (f,c,t), UnitTransportCost(f,c) * Transport(f,c,t,s) ]
    }
}

Variable TotalCost {
    Range        : free;
    Unit         : $;
    Definition   : sum[ s, ScenarioProbability(s) * TotalScenarioCost(s) ];
}
```

Defined variables

Note that four of the variables have their own definitions. AIMMS will treat these definitions as constraints when generating the corresponding mathematical program.

Constraints

The remaining three constraints in the formulation of the mathematical program are listed below.

```
Constraint ChangeWhenIncrease {
    IndexDomain  : (f,p,t) | p in FactoryProductionLines(f);
    Definition   : {
        ProductionLineLevelChange(f,p,t) >=
        ProductionLineInUse(f,p,t) - ProductionLineInUse(f,p,t-1)
    }
}

Constraint ChangeWhenDecrease {
    IndexDomain  : (f,p,t) | p in FactoryProductionLines(f);
    Definition   : {
        ProductionLineLevelChange(f,p,t) >=
        ProductionLineInUse(f,p,t-1) - ProductionLineInUse(f,p,t)
    }
}

Constraint RestrictTransportCapacity {
    IndexDomain  : (f,t,s);
    Unit         : TL;
    Definition   : sum[ c, Transport(f,c,t,s) ] <= MaximumTransportCapacity(f);
```

```
}
```

Mathematical program

A mathematical program in AIMMS specifies the set of variables and constraints together with the objective, optimization direction and model type that are needed by AIMMS to generate the model. If you do not specify a variable set or a constraint set, AIMMS will assume that all model variables and all model constraints are included in the mathematical program. Please use the **Objective**, the **Direction** and the **Type** wizard to declare the mathematical program `LeastCostPlan` as shown in Figure 7.4.

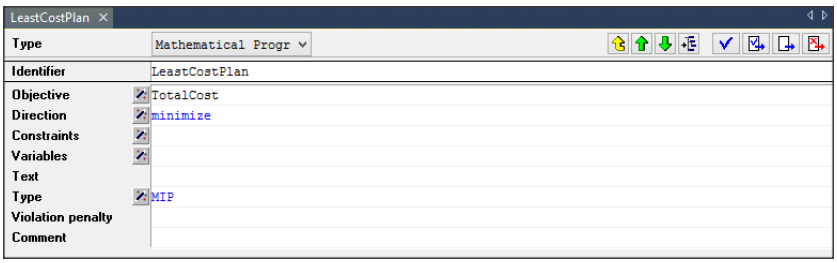


Figure 7.4: Attribute form of the mathematical program

Model tree

All variables and constraints that are declared in the `Mathematical Program Declarations` are shown in Figure 7.5.

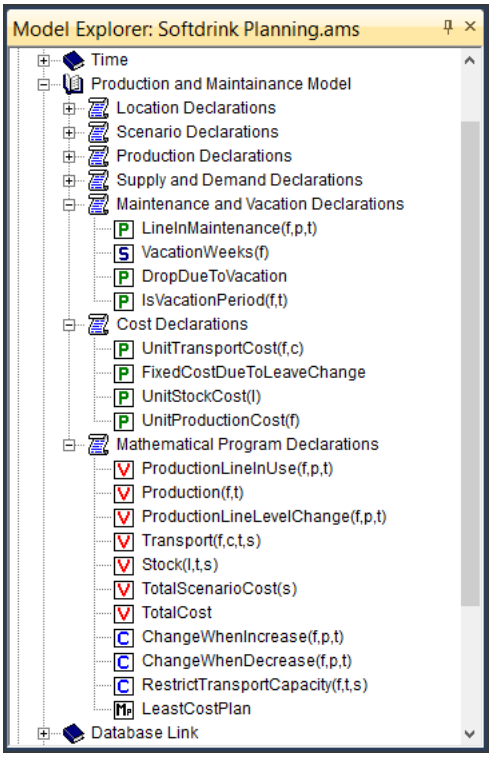


Figure 7.5: The model tree to date

Part III

Model Procedures and Functions

Chapter 8

Linking to the Database

This chapter

In this chapter you will experience how straightforward it is to link your model to a database using the point-and-click database interaction facilities of AIMMS. In addition, the possibility of entering SQL procedures in AIMMS is also illustrated.

Further reading

If you follow the steps in this chapter and you decide that you need to know more about database linkage, please look at the Chapter 'Communicating with Databases' in *The Language Reference*.

8.1 Database tables

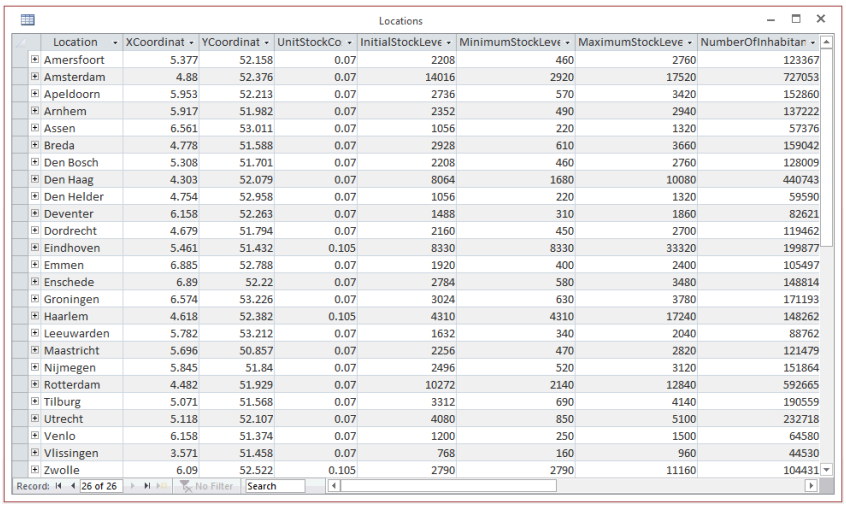
ODBC and MS-Access

The linkage between AIMMS and a database relies on the ODBC (Open DataBase Connectivity) standard. You will need to be aware of the version (32 bit or 64 bit) of the ODBC and Microsoft Office you have installed on your Computer. The Microsoft Access Database Engine 2010 Redistributed will allow you to run different versions on you OCDB and Microsoft Access. More detailed information on how to install the Microsoft Access Database Engine 2010 Redistributed for different versions can be found on this technical blog;

http://techblog.aimms.com/2014/10/27/installing-32-bit-and-64-bit-microsoft-access-driver

Columns and rows

The basic building blocks of a database are database tables containing columns and rows. One or more columns in a particular database table serve as so-called primary key columns. The remaining columns contain data defined over these key columns. The primary key values found in each row uniquely identify that row. For example, the first column in Figure 8.1 is a primary key column and identifies every row uniquely through the name of each location.



Locations

Location	XCoordinat	YCoordinat	UnitStockCo	InitialStockLeve	MinimumStockLeve	MaximumStockLeve	NumberOfInhabitar
Amersfoort	5.377	52.158	0.07	2208	460	2760	123367
Amsterdam	4.88	52.376	0.07	14016	2920	17520	727053
Apeldoorn	5.953	52.213	0.07	2736	570	3420	152860
Arnhem	5.917	51.982	0.07	2352	490	2940	137222
Assen	6.561	53.011	0.07	1056	220	1320	57376
Breda	4.778	51.588	0.07	2928	610	3660	159042
Den Bosch	5.308	51.701	0.07	2208	460	2760	128009
Den Haag	4.303	52.079	0.07	8064	1680	10080	440743
Den Helder	4.754	52.958	0.07	1056	220	1320	59590
Deventer	6.158	52.263	0.07	1488	310	1860	82621
Dordrecht	4.679	51.794	0.07	2160	450	2700	119462
Eindhoven	5.461	51.432	0.105	8330	8330	33320	199877
Emmen	6.885	52.788	0.07	1920	400	2400	105497
Enschede	6.89	52.22	0.07	2784	580	3480	148814
Groningen	6.574	53.226	0.07	3024	630	3780	171193
Haarlem	4.618	52.382	0.105	4310	4310	17240	148262
Leeuwarden	5.782	53.212	0.07	1632	340	2040	88762
Maastricht	5.696	50.857	0.07	2256	470	2820	121479
Nijmegen	5.845	51.84	0.07	2496	520	3120	151864
Rotterdam	4.482	51.929	0.07	10272	2140	12840	592665
Tilburg	5.071	51.568	0.07	3312	690	4140	190559
Utrecht	5.118	52.107	0.07	4080	850	5100	232718
Venlo	6.158	51.374	0.07	1200	250	1500	64580
Vlissingen	3.571	51.458	0.07	768	160	960	44530
Zwolle	6.09	52.522	0.105	2790	2790	11160	104431

Record: 26 of 26 No Filter Search

Figure 8.1: Contents of the table 'Locations'

Four database tables

The database delivered with this tutorial contains four database tables. The first table contains data that are applicable to both factories and distribution centers (e.g. coordinate data and stock level data). The second table provides data that are needed to configure the factories (e.g. production capacity and cost data). Historical data (e.g. demand values over time) have been placed inside the third table, and will be used to initiate the rolling horizon process. Finally, the fourth database table contains the data that are needed to configure the individual production lines (e.g. production line capacities).

8.1.1 Entering the first database table declaration

Database table in AIMMS

You can refer to an external database table within AIMMS by means of a *database table* identifier declaration. As an attribute you can specify the ODBC data source name of the database you want to access, and also the name of the external database table from which you want to read or to which you want to write.

Creating the `LocationTable`

To declare your first database table in AIMMS, you should perform the following actions:

- create a new declaration section named `Database Declarations` under the `Database Link` section of the model tree,
- open the new declaration section,
- press the **Other**... button on the toolbar,

- create a new database table identifier in this new declaration section by selecting the database table icon in the **Select Type of Identifier** dialog box, and
- specify 'LocationTable' as its name.

Specifying the data source attribute

An MS Access database file named '`Softdrink Factory Planning.mdb`' has been supplied with this tutorial. Next, you will make this database available to AIMMS by performing the following actions:

- activate the **Data source** wizard in the attribute form of the database table 'LocationTable',
- choose the **Select File Data Source**... command in the menu that pops up,
- select the file '`Softdrink Planning.dsn`' from the '`Data`' subdirectory, and
- press the **Save** button.

Specifying the table name attribute

Once you have created the data source, you are now ready and able to select a table from this source. Please, execute the following simple steps:

- activate the **Table name** wizard,
- choose the **Select Table/Query Name**... command from the pop-up menu,
- select 'Locations', and
- press the *OK* button.

Look at the external table

If you have not worked with external databases before, it may be of interest to look at the external database table as it appears in the database. For this purpose, you can start MS Access, and inspect the design view of database table `Locations` as shown in Figure 8.2.

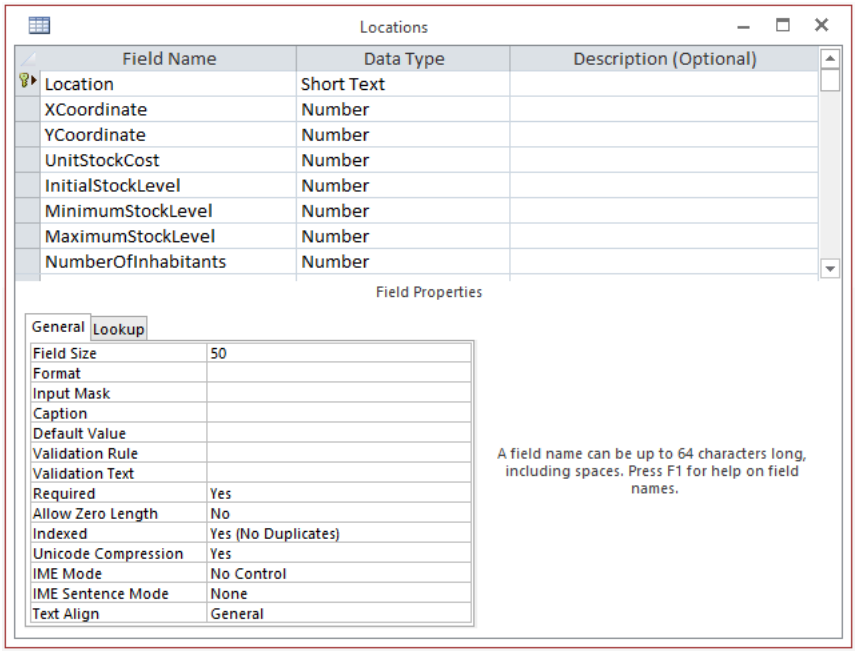


Figure 8.2: The MS Access design view of the Locations table

Specifying the mapping attribute

In general, the naming convention used inside a database table will not be identical to the naming convention used for the corresponding identifiers in AIMMS. That is why a mapping is needed to relate columns in the external database table to identifiers in AIMMS. For example, the mapping between the index identifier l in AIMMS and the column named 'Location' in the database can be specified as follows:

- activate the **Mapping** wizard,
- select the primary key "Location" from the 'Data Column' drop down list (see Figure 8.3),
- press the wizard button to select the index l as the 'AIMMS Identifier',
- press the **Transfer** button to put the specified mapping into the 'Mappings' list, and
- press the **OK** button.

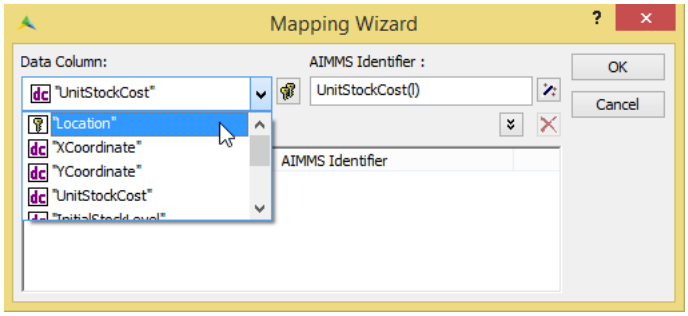


Figure 8.3: The **Mapping** wizard

Completing the mapping

Please look at Figure 8.4, and complete the mapping attribute accordingly using the wizard as explained in the previous paragraph.

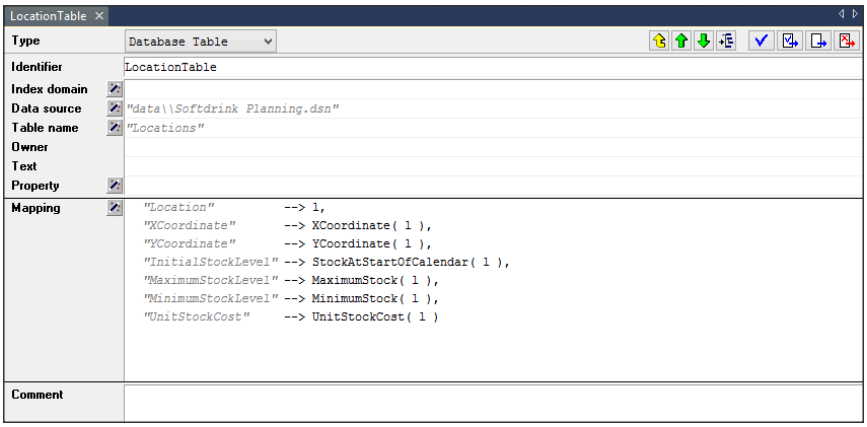


Figure 8.4: Attribute form of the data table 'Locations'

8.1.2 Entering additional database table declarations

Weekly demand data

Once you have completed your first database table declaration as described in the previous section, you can make the remaining three external database tables available to AIMMS. Before entering the corresponding declarations you need to declare two additional model parameters to store the weekly demand data read from the database.

```
Parameter WeeklyDemand {
    IndexDomain : (c,w,s);
    Unit        : hl;
}
```

```
Parameter TotalWeeklyDemand {
    IndexDomain : (w,s);
    Unit        : hl;
}
```

Adding the three database tables

First declare the three additional database table identifiers `FactoryTable`, `CenterTable` and `ProductionLineTable` in the model tree (just below the parameter `TotalWeeklyDemand`). Then consider the attribute descriptions listed below. Next fill in the three attribute forms accordingly, using the **Data source** wizard, the **Table name** wizard, and the **Mapping** wizard.

```
DatabaseTable FactoryTable {
    DataSource : "data\\Softdrink Planning.dsn";
    TableName  : "Factories";
    Mapping    : {
        "Factory"                --> f,
        "UnitProductionCost"     --> UnitProductionCost( f ),
        "MaximumTransportCapacity" --> MaximumTransportCapacity( f )
    }
}

DatabaseTable CenterTable {
    DataSource : "data\\Softdrink Planning.dsn";
    TableName  : "Centers";
    Mapping    : {
        "Center"   --> c,
        "Date"     --> w,
        "Scenario" --> s,
        "Demand"   --> WeeklyDemand( c, w, s )
    }
}

DatabaseTable ProductionLineTable {
    DataSource : "data\\Softdrink Planning.dsn";
    TableName  : "ProductionLines";
    Mapping    : {
        "Factory"               --> f,
        "ProductionLine"        --> p,
        "InitialUsageCount"     --> DeteriorationLevelAtStartOfCalendar( f, p ),
        "InitialProductionLevel" --> ProductionLineLevelAtStartOfCalendar( f, p ),
        "MaximumProductionLevel" --> MaximumProductionLineLevel( f, p ),
        "MaximumUsageCount"     --> MaximumDeteriorationLevel( f, p )
    }
}
```

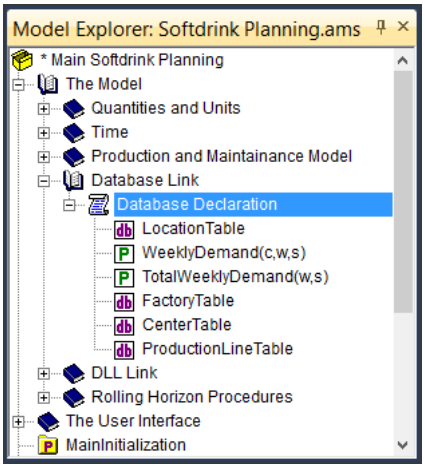


Figure 8.5: The database section of the model tree so far

8.2 Database procedures

Sophisticated control

When transferring data from, or to, a database table, you may need more sophisticated control over the data link than offered by the standard database table interface. AIMMS offers you this additional control by letting you write and execute *SQL* (Structured Query Language) statements, or providing access to *stored procedures* already available inside the database.

8.2.1 SQL queries

A first SQL query . . .

It is possible to access data values in a database that are not directly stored in one of its database tables. Consider, for instance, the database table named "ProductionLines" with the two primary key columns "Factory" and "ProductionLine". In this database table, there is no entry for the number of production lines in each factory. However, this information can be obtained from the database through the following *query* using SQL.

```
SELECT Factory, COUNT(ProductionLine) AS LineCount
FROM ProductionLines GROUP BY Factory
```

This query temporarily creates a new table inside the database consisting of two columns. The first column is a primary key named 'Factory', while the second column is named 'LineCount' and contains the required totals.

... declared in AIMMS

To implement this query in AIMMS, you can create your first database procedure named `NumberOfProductionLinesQuery`. The following steps are required:

- close the declaration section named `Database Declarations` by double clicking on the scroll icon ,
- press the **Other**... button on the toolbar,
- select the database procedure from the **Select Type of Node** dialog box (see Figure 8.6), and press the **OK** button,
- enter 'NumberOfProductionLinesQuery' as the name of the database procedure, and
- press the *Enter* key to register the name.

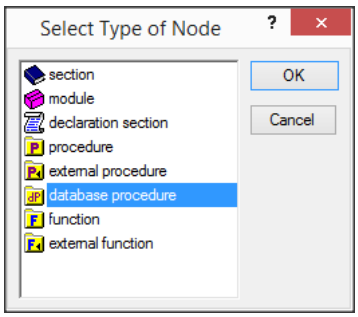


Figure 8.6: The **Select Type of Node** dialog box

Specifying the database procedure attributes

After opening the attribute form of the database procedure, please complete it as shown in Figure 8.7. Note that the SQL text must be in double quotes, and can be split over several "quoted" lines using the + operator and the appropriate use of spaces to ensure that consecutive words are not run together. The specified 'UseResultSet' **Property** attribute enables you to use the database procedure as if it were a database table. Without this property, AIMMS does not allow you to specify the **Mapping** attribute, necessary to read data. Note that the **Mapping** wizard is not available for SQL queries.

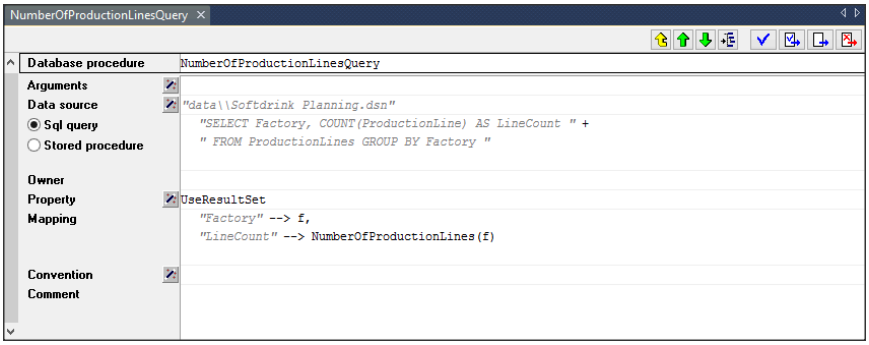


Figure 8.7: A database procedure to execute an SQL command

8.2.2 Stored procedures

Procedures inside the database ...

In the previous subsection, you placed your own SQL query inside an AIMMS database procedure. In this subsection, you will consider a query that already resides inside the database, and that you can also access from within an AIMMS database procedure.

... with or without arguments

A stored procedure can have one or more arguments, and it is straightforward to specify these arguments in an AIMMS database procedure. In this tutorial, however, the stored procedure named `TotalDemand` and `AllCenters` are used, and these procedures happen not to have arguments.

Declaring the database procedures

To declare your second database procedure, please execute the following actions:

- insert a new database procedure in the model tree, and specify 'TotalDemandQuery' as its name,
- open its attribute form,
- use the **Data source** wizard to select 'Softdrink Planning.dsn' as its **Data source** attribute,
- press the radio button in front of the **Stored procedure** attribute,
- activate the **Stored procedure** wizard,
- choose the **Select Stored Procedure Name...** command in the menu that pops up,
- select 'TotalDemand' as the **Stored procedure** attribute,
- complete the attribute form as shown in Figure 8.8, and
- close the attribute form using the **Check, commit and close** button

Figure 8.8: The completed attribute form of the database procedure `TotalDemandQuery`

And to declare your third database procedure with 'AllCentersQuery' as its name, please perform similar steps as mentioned above, only this time select 'AllCenters' as the **Stored procedure** attribute. The completed attribute form should look like the one in Figure 8.9).

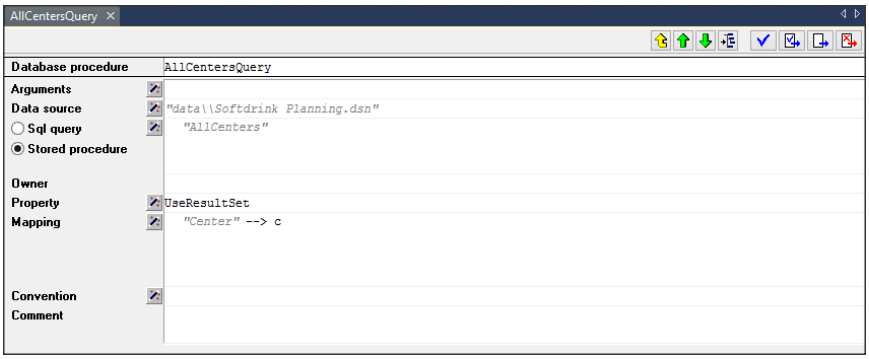


Figure 8.9: The completed attribute form of the database procedure `AllCentersQuery`

The part of the model tree describing the database link is shown in Figure 8.10.

Database declarations so far

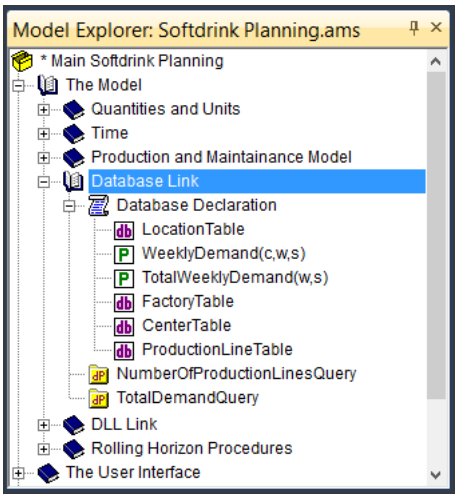


Figure 8.10: An intermediate model tree showing all database identifiers and procedures

Chapter 9

Functions and Procedures

This chapter

In the previous chapter you were introduced to database procedures. In this chapter you will develop several AIMMS procedures to read data and to control the entire rolling horizon process. In addition, you will work with an external procedure that is called from within AIMMS.

Many small procedures

The procedures in this chapter have all been kept small for ease of understanding. The underlying rolling horizon algorithm, however, is not trivial, and results in a multitude of procedures. The chapter is therefore both a tutorial in the use of procedures and a tutorial in the application of a rolling horizon.

9.1 Reading from a database

Reading a database table ...

Reading all the data at once from a database table is quite easy in AIMMS. Consider, for instance, the database table LocationTable declared in the previous chapter. The following statement

```
read from table LocationTable;
```

is an instruction to AIMMS to read all identifiers that have been specified in the **Mapping** attribute of the corresponding database table.

... or a portion thereof

It is also possible to read a *selection* of all identifiers specified in the **Mapping** attribute of a database table. For instance, the following statement

```
read XCoordinate, YCoordinate from table LocationTable;
```

only reads data of XCoordinate and YCoordinate.

Creating a procedure

At this point, you are asked to create a single procedure named ReadFromDatabase to be placed between the Database Declarations node and the NumberOfProductionLinesQuery node in the model tree in the following manner:

- select the Database Link section of the model tree,

- if open, close this section by clicking on the minus sign in front of the icon,
- press the **New Procedure** button button on the toolbar,
- enter 'ReadFromDatabase' as the name of the procedure, and
- press the *Enter* key to register this name.

Completing the **Body** *attribute*

Open the attribute form of the procedure `ReadFromDatabase` by double-clicking on its name, and complete the **Body** attribute as shown in Figure 9.1. Note that the two database procedures `NumberOfProductionLinesQuery` and `TotalDemandQuery` both result in temporary tables inside the database, and that AIMMS acts as if the name of each procedure is the same as the name of the temporary table.

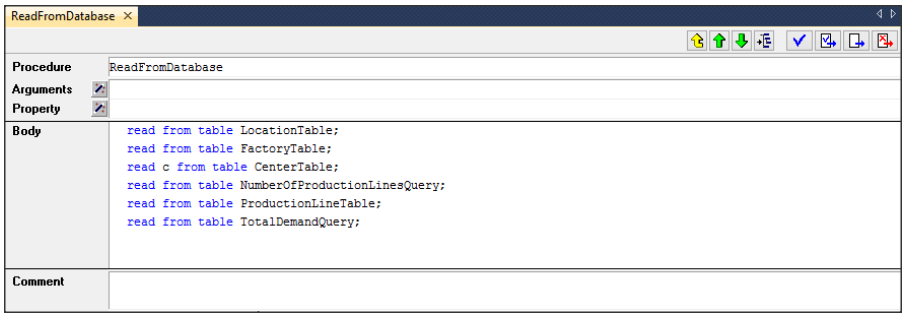


Figure 9.1: The procedure 'ReadFromDatabase'

Running the procedure

After you have completed the **Body** attribute of the procedure `ReadFromDatabase`, close the attribute form using the **Check, commit and close** button . You can now run the procedure by performing the following steps:

- select the procedure `ReadFromDatabase` in the model tree, and
- select the **Run Procedure** command using the right-mouse pop-up menu (see Figure 9.2).

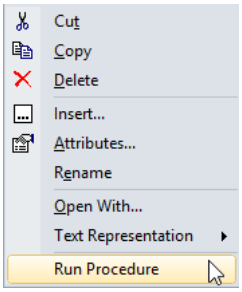


Figure 9.2: The right-mouse menu of the procedure 'ReadFromDatabase'

Finding an identifier . . .

After you have executed the procedure `ReadFromDatabase` you may want to look at the contents of, for instance, the parameter `MaximumProductionLineLevel`. Before you are able to view its data, you need to locate this parameter in the model tree. You can find it in the following manner:

- press the **Find** button on the toolbar,
- enter 'MaximumProductionLineLevel' using the name completion facility (see Figure 9.3), and
- press the **Declaration**. . . button.

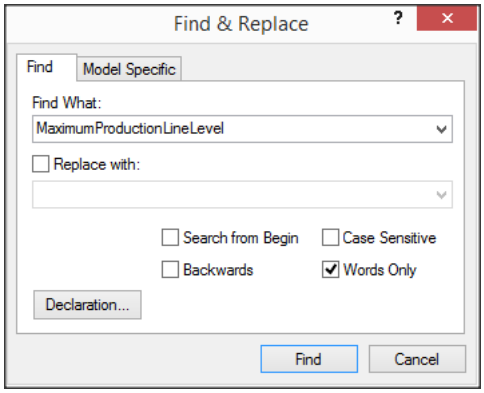


Figure 9.3: The **Find & Replace** dialog box

. . . and inspecting its data

Next, open the data page for the parameter `MaximumProductionLineLevel` by performing the following two steps:

- press the right-mouse button to activate the pop-up menu, and
- select the **Data**. . . command.

The data page on your computer should now look like the one shown in Figure 9.4.

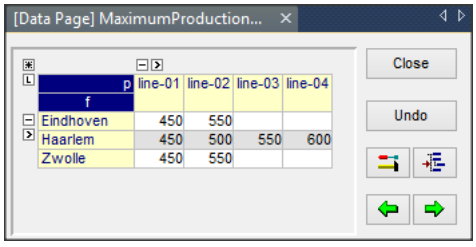


Figure 9.4: The data page for `MaximumProductionLineLevel`

9.2 External DLL functions

External DLL

In this section, you will link an external *Dynamic Link Library* (DLL) named 'External Routines.dll' to your AIMMS model. Inside this DLL, there is a function named DLLUnitTransportCost, that determines the unit transport cost on the basis of the distance between a particular factory and a particular distribution center. Writing your own DLLs is beyond the scope of this tutorial. Chapters 11 and 34 of *The Language Reference*, however, elaborate further on the use of DLLs and the related AIMMS Programming Interface. The source code of 'External Routines.dll' has already been copied to the 'DLL' subdirectory of your project.

DLL function ...

The DLL 'External Routines.dll' exports the following function.

```
double DLLUnitTransportCost( char *from_name, char *to_name )
```

The two input arguments of the function are strings representing the names of the two locations for which the unit transport cost is calculated.

... and its counterpart in AIMMS

For each external DLL function used in an AIMMS application, you must declare a corresponding *external function* in AIMMS. In this tutorial, the external function is named ExternalUnitTransportCost, and has the same number of arguments as its external counterpart.

Declaring an external function

To declare the external function you should perform the following tasks:

- open the DLL Link model section,
- press the **Other**... button on the toolbar,
- select the external function from the **Select Type of Node** dialog box (see Figure 9.5),
- specify 'ExternalUnitTransportCost(factory,center)' as the name of the function, and
- press the *Enter* key to register its name.

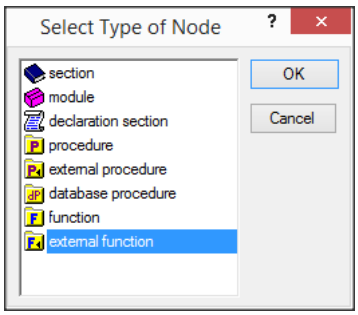


Figure 9.5: The **Select Node Type** dialog box

The **Arguments** *wizard*

Next, AIMMS will automatically open the **Arguments** wizard as shown in Figure 9.6.

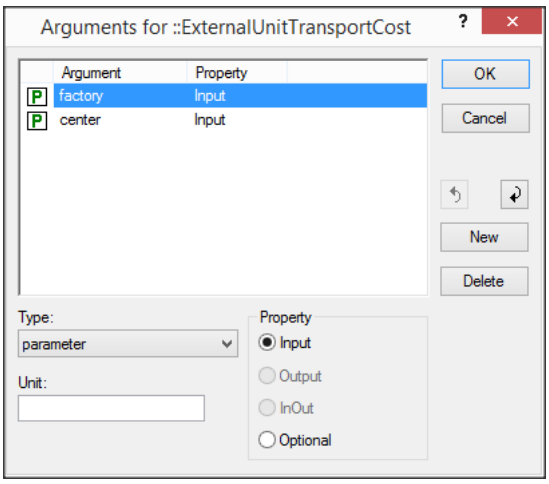


Figure 9.6: The **Arguments** wizard

To complete the **Arguments** wizard, execute the following steps:

- change the type of the currently selected argument `factory` to 'element parameter',
- select `Factories` as its **Range** attribute,
- then click on the second argument `center`,
- change its type to 'element parameter',
- select `Centers` as its **Range** attribute, and
- press the **OK** button.

After completing the **Arguments** wizard, AIMMS will have declared the two input arguments as local element parameters. You may verify that AIMMS has

indeed placed these local parameters in a new declaration section underneath the `ExternalUnitTransportCost` node in the model tree (see Figure 9.7).

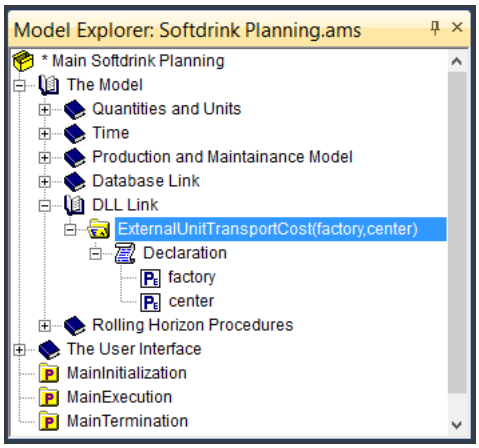


Figure 9.7: The completed `DLL` section of the model tree

Completing the attributes

Using wizards it is now straightforward to complete the **Dll name** and **Return type** attributes of the external function as shown in Figure 9.8.

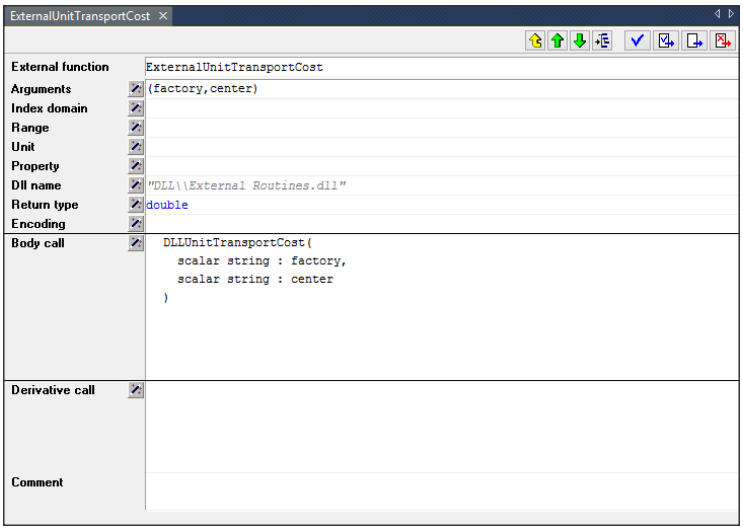


Figure 9.8: The attribute form of the external function `ExternalUnitTransport-Cost`

The **Body call** *attribute*

The **Body call** attribute specifies the actual link between the arguments of the function in AIMMS and in the DLL. There is an extensive **Body call** wizard, as shown in Figure 9.9, which supports several choices in establishing the link. In the **Body call** wizard (see Figure 9.9) you should perform the following actions:

- select 'Scalar' translation type
- press the wizard button to select the element parameter `factory` as the actual argument,
- set the external datatype to 'String',
- press the **Add** button,
- select 'Scalar' translation type
- press the wizard button to select the element parameter `center` as the actual argument,
- set the external datatype to 'String',
- press the **Add** button, and
- press the **OK** button.

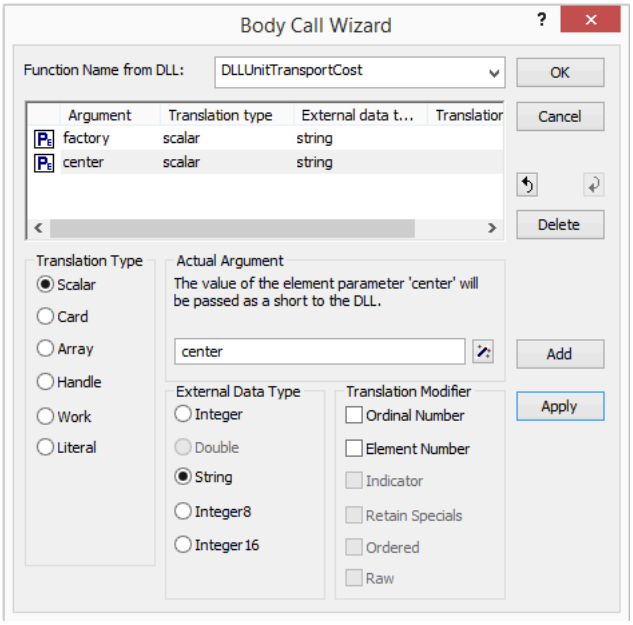


Figure 9.9: The **Body call** wizard

9.3 Specifying the rolling horizon

This section

In this section, you will specify all the procedures that are necessary to describe the rolling horizon process. Once you have implemented the single step contained in this process, it becomes straightforward to describe the overall

process. After proper data initialization you are then ready to run the completed set of rolling horizon procedures.

Structuring the tree

This section is divided into four subsections, as shown in Figure 9.10. You should add these subsections to your own model tree.

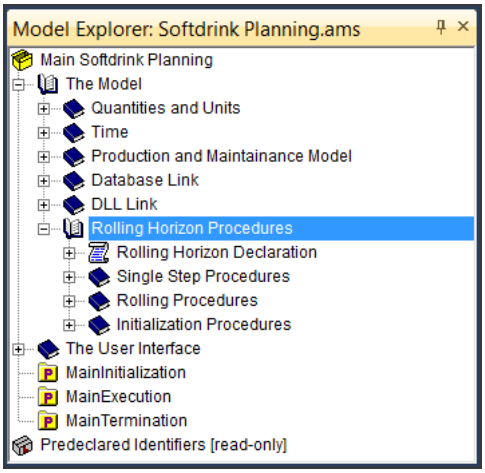


Figure 9.10: The structure of the `Rolling Horizon Procedures` section

9.3.1 Rolling horizon declarations

Horizon identifiers . . .

There are several identifiers that play a role in the rolling horizon process. Their names are mostly self-explanatory, and their contents are specified below. As you will see in the next subsection, these identifiers are used in the formation of timetables, which link the abstract periods in the rolling horizon model to the specific days and weeks in the two calendars.

. . . and their declarations

At this stage, you should enter the following declarations in `Rolling Horizon Declarations`.

```
ElementParameter FirstDayInPlanningInterval {
    Range       : Days;
}

Set WeeksInPlanningInterval {
    SubsetOf    : Weeks;
    Definition  : union[ t, WeekInPeriod(t) ];
}

ElementParameter FirstWeekInPlanningInterval {
    Range       : Weeks;
    Definition  : DayToWeek(FirstDayInPlanningInterval);
```

```
}

ElementParameter LastWeekInPlanningInterval {
    Range       : Weeks;
    Definition  : last(WeeksInPlanningInterval);
}

Parameter LenghtDominatesNotActive {
    IndexDomain : t;
}
```

Additional explanation

The identifier named LengthDominatesNotActive is a required input for the procedure CreateTimeTable discussed in the next subsection. Whenever this identifier assumes its default value of zero, then the desired length of any period may not be achieved due to a delimiter slot being encountered in that period. In the example in this tutorial, this parameter is indeed zero. As a result, the timetable DaysInPeriod will make sure that each period starts on a Monday (the delimiter slot). Even though the desired length of each period has been set to seven days, its actual length is shortened due to weekends and the official holidays (the so-called inactive days).

Registration identifiers

In addition to the five horizon identifiers, you need to enter two registration identifiers. These two identifiers are used to store the overall maintenance and line usage planning. Add the following two parameters at the end of the Rolling Horizon Declarations section:

```
Parameter OverallMaintenancePlanning {
    IndexDomain : (f,p,w) | p in FactoryProductionLines(f);
}

Parameter OverallLineUsagePlanning {
    IndexDomain : (f,p,w) | p in FactoryProductionLines(f);
}
```

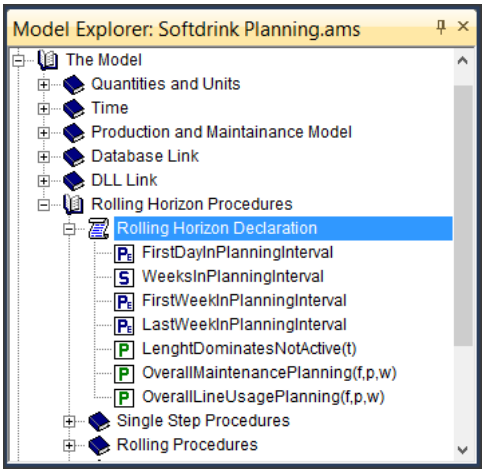


Figure 9.11: The `Rolling Horizon Declarations` section of the model tree

9.3.2 Single step procedures

Tree structure

A single step in the rolling horizon decision process can be divided into several procedures, as shown in Figure 9.12. The implementation of each procedure will be discussed later on in this subsection. Complete your model tree accordingly, but please follow the instructions in the next paragraph when entering the procedure `RegisterInOverallPlanning` with its two arguments named `iw` and `ip`.

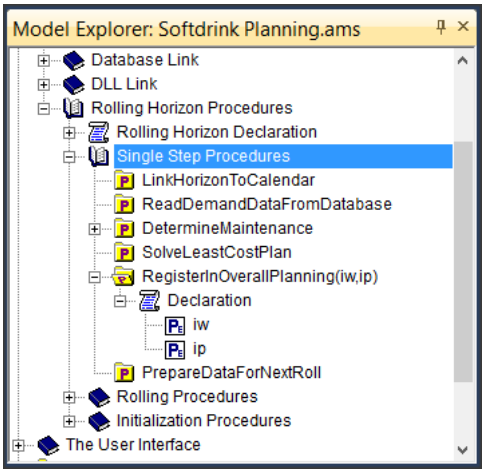


Figure 9.12: The procedures needed to specify a single step

Argument wizard

Once you enter the procedure `RegisterInOverallPlanning(iw,ip)` with its two arguments, AIMMS will automatically open a wizard. To complete this **Argument** wizard for both `iw` (referring to a week) and `ip` (referring to a period), you should execute the following actions:

- change the type of the currently selected argument `iw` to 'element parameter',
- select `Weeks` as its **Range** attribute,
- select 'Input' as its **Property** attribute,
- then click on the second argument `ip` to change the target,
- change its type to 'element parameter',
- select `Periods` as its **Range** attribute, and
- select 'Input' as its **Property** attribute.

At this point, the **Argument** wizard should be the same as the one shown in Figure Figure 9.13.

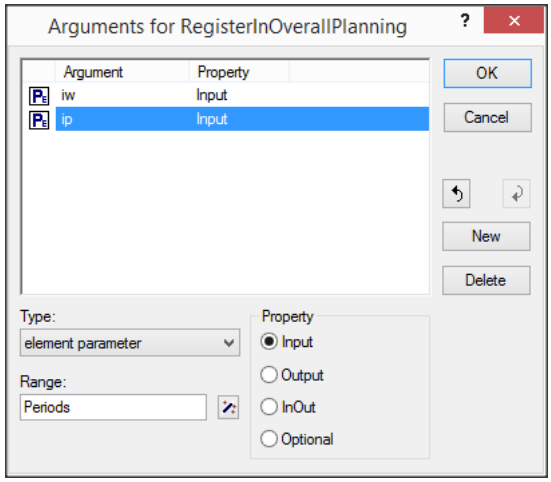


Figure 9.13: **Argument** wizard

Describing a timetable

A *timetable* is either an indexed set or an indexed element parameter, representing the mapping between the periods in the horizon and the timeslots in the calendar. It is an indexed set when the period can contain several time slots as for instance in the timetable `DaysInPeriod`. It can be an indexed element parameter when there is a one-to-one mapping between each period and each time slot as for instance in the timetable `WeekInPeriod`.

Creating a timetable . . .

The quick info tip window of the predefined procedure `CreateTimeTable` are shown in Figure 9.14.

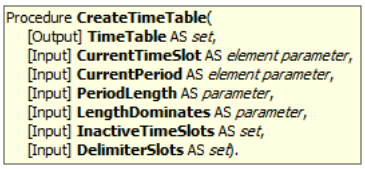


Figure 9.14: Quick info tip window of the function `CreateTimeTable`

Through the arguments you have considerable control over the contents of the timetable. For detailed information see Section 29.4 of *The Language Reference* manual.

... in an AIMMS *procedure*

Go to the **Body** attribute of the procedure LinkHorizonToCalendar, and enter the following statements:

```
CreateTimeTable(
    TimeTable         : DaysInPeriod,
    CurrentTimeSlot   : FirstDayInPlanningInterval,
    CurrentPeriod     : FirstPeriodInPlanningInterval,
    PeriodLength      : DesiredNumberOfDaysInPeriod,
    LengthDominates   : LengthDominatesNotActive,
    InactiveTimeSlots : InactiveDays,
    DelimiterSlots    : Mondays );

ActualNumberOfDaysInPeriod(t) := (card(DaysInPeriod(t))) [day];

CreateTimeTable(
    TimeTable         : WeekInPeriod,
    CurrentTimeSlot   : FirstWeekInPlanningInterval,
    CurrentPeriod     : FirstPeriodInPlanningInterval,
    PeriodLength      : DesiredNumberOfWeeksInPeriod,
    LengthDominates   : LengthDominatesNotActive,
    InactiveTimeSlots : InactiveWeeks,
    DelimiterSlots    : Weeks );
```

Argument names are optional

Note that when calling CreateTimeTable, the arguments are preceeded by their argument names as displayed in Figure 9.14. The use of argument names in function calls is optional in AIMMS. In the above **Body** attribute, the argument names are used to increase the readability.

Overriding units

In order to enforce unit consistency in the above assignment statement, the unitless expression card(DaysInPeriod(t)) is assigned the unit [day]. Such unit casting requires the entire expression to be enclosed between parentheses.

Maximizing attribute fields

You can use the **Maximized** button from the **Edit** menu to temporarily enlarge the size of the **Body** attribute (or any other multi-line attribute) to ease entry. When you have completed the attribute, simply press the **Maximize** button again to restore the original size.

DaysInPeriod ...

The timetable DaysInPeriod contains the working days in a week, and explicitly excludes the inactive days such as the weekends and the official holidays. The sole reason why this timetable is created, is to determine the parameter ActualNumberOfDaysInPeriod needed to establish the correct level of production.

... requires one more initialization

To view the contents of the DaysInPeriod timetable, you should first initialize the element parameter FirstDayInPlanningInterval. All other input arguments have already been initialized. Execute the following steps:

- select the procedure LinkHorizonToCalendar in the model tree,
- press the *Enter* key to open its attribute form,

- position the text cursor somewhere within the string 'FirstDayInPlanningInterval' in the **Body** attribute,
- press the right-mouse button to activate the pop-up menu,
- select the **Data**... command,
- click on the empty right-hand side of the equal sign in the *Data* page,
- specify '03/07/2000' (without the quotes) as the value on the data page, and
- press the **Close** button.

You may re-open the page to verify that AIMMS has accepted your input value. If the input format you entered was incorrect, AIMMS will replace your input with the default empty string.

... is first determined

At this point, you can view the contents of the timetable `DaysInPeriod` by running the procedure and looking at the appropriate data page:

- position the text cursor somewhere within the string 'LinkHorizonToCalendar' in the **Procedure** attribute,
- press the right-mouse button to activate the pop-up menu, and
- select the **Run Procedure** command.

You can ignore all the initialization warnings since the existing default values suffice at this point in the tutorial. Please close the **Errors/Warnings** window and continue.

... and can then be viewed

Next construct the data page corresponding to the timetable `DaysInPeriod` as shown in Figure 9.15 by executing the following steps:

- position the text cursor somewhere within the string 'DaysInPeriod' in the **Body** attribute,
- press the right-mouse button to activate the pop-up menu again, and
- select the **Data**... command.

Note that each period covers exactly five days due to the fact that the weekends are excluded. The default format of this data page requires you to scroll horizontally. You may select a different view by pressing the **Change view** button , and choosing, for instance, 'Sparse List' as the Type of Object.

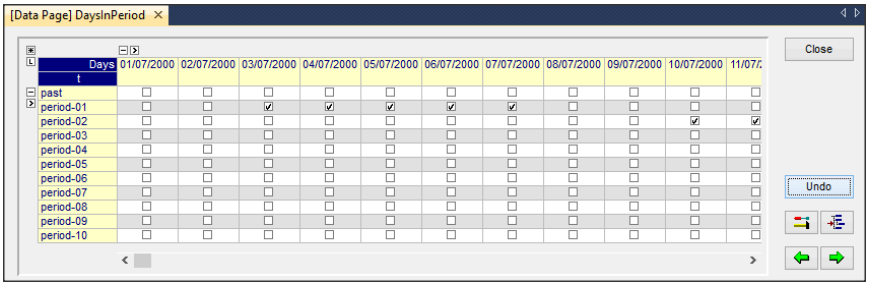


Figure 9.15: The data page of the day-based timetable

Reading just a subset of demand data ...

The weekly calendar in this tutorial spans a period of roughly one year. The planning horizon in a single step of the overall rolling horizon procedure, however, is just a small subset of weeks. That is why the procedure `ReadDemandDataFromDatabase` is introduced to limit the total amount of demand data that is loaded into memory at any given time.

... goes as follows

Prior to each subsequent step of the rolling horizon process, it is recommended that you first emptie the weekly demand data associated with the old planning interval, and then read the demand data for the weeks in the new planning interval. This can be accomplished by entering the following statements in the **Body** attribute of the procedure `ReadDemandDataFromDatabase`.

```
empty WeeklyDemand;
read WeeklyDemand(c,w,s) from table CenterTable
                         filtering w in WeeksInPlanningInterval;
Demand(c,t,s) := WeeklyDemand(c,WeekInPeriod(t),s);
```

Note that the weekly demand is read for only those weeks that are in the current planning interval. Using the timetable `WeekInPeriod`, the weekly demand is then assigned to period demand as required by the mathematical program to be solved.

Determine when under maintenance

The parameter `DeteriorationLevel` registers, for each combination of factory and production line, the amount of time that has elapsed since that line was maintained. Assuming that all lines will be in use for the entire planning interval, it is a straightforward calculation to estimate when a production line should be under maintenance.

At most one line under maintenance in first period

Now comes the slightly tricky requirement: in each factory no more than one production line can be maintained in the first period. If there is more than one candidate, you should maintain just one line, and delay the maintenance of the other candidate(s) to the next period. The final result is then stored in the parameter `LineInMaintenance` declared for each factory, production line

and period. This parameter is one of the determinants of the production level of a line when in use (see the definition of the parameter PotentialProduction).

Entering local declarations

Before specifying the **Body** attribute of the procedure DetermineMaintenance, you need to declare the following two local identifiers in a new declaration section within the procedure node DetermineMaintenance.

```
ElementParameter EstimatedMaintenancePeriod {
    IndexDomain : (f,p);
    Range       : Periods;
}

Set LinesInMaintenanceInFirstPeriod {
    IndexDomain : f;
    SubsetOf    : ProductionLines;
}
```

Figure 9.16 shows the local declaration section of the procedure DetermineMaintenance.

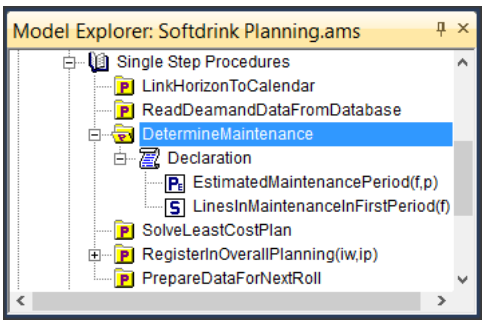


Figure 9.16: The local declaration of the procedure DetermineMaintenance

Entering maintenance calculations

The following statements in AIMMS have been discussed in the previous paragraph. Please enter them in the **Body** attribute of the procedure DetermineMaintenance.

```
EstimatedMaintenancePeriod(f,p) :=
   Element( Periods, max( MaximumDeteriorationLevel(f,p) -
                          Floor(DeteriorationLevel(f,p)) + 2, 2 ) );

LinesInMaintenanceInFirstPeriod(f) :=
   { p | EstimatedMaintenancePeriod(f,p) = FirstPeriodInPlanningInterval };

EstimatedMaintenancePeriod( (f,p) |
   Ord(p,LinesInMaintenanceInFirstPeriod(f)) >= 2 ) += 1;

empty LineInMaintenance;
LineInMaintenance((f,p,EstimatedMaintenancePeriod(f,p)) |
   EstimatedMaintenancePeriod(f,p) in Periods.Planning ) := 1;
```

Entering a 'solve' procedure ...

Having completed the first three single step procedures, you are now ready to enter the procedure in which the single step mathematical program is solved. Please enter the following statements in the **Body** attribute of the procedure SolveLeastCostPlan.

```
solve LeastCostPlan;
halt with "Least cost mathematical program is not optimal.\nCheck "
        + "input data for infeasibilities."
     when ( LeastCostPlan.ProgramStatus <> 'Optimal' );
```

... with a halt statement

Note that the second statement illustrates the use of the halt statement in AIMMS. Once the program halts, it will provide a two-line message as indicated by the special character '\n'. By using the + notation in the **Body** attribute, you may divide a single quoted string into several pieces. In the conditional when part of the halt statement, there is a reference to a property of the mathematical program, namely the *program status*, using the 'dot' notation (see Section 15.2 in *The Language Reference*).

Register overall planning

Following the solution of the single step mathematical program, the results associated with just the first period are kept as 'definite'. In this tutorial, only the overall planning of maintenance and the overall planning of production line usage are kept. The overall planning is registered in terms of calendar weeks, which implies that period data must be translated into week data. Such translation is achieved with the following two statements, to be added to the **Body** attribute of the procedure RegisterInOverallPlanning:

```
OverallMaintenancePlanning(f,p,iw) := LineInMaintenance(f,p,ip);
OverallLineUsagePlanning(f,p,iw)   := ProductionLineInUse(f,p,ip);
```

Preparing data for next step

Once the overall planning has been registered, all that remains is to prepare several data items for the next step. First of all, the first day in the planning interval must be moved forward seven days to the next Monday. Then the current first-period stock and production solution data must become historic data. Finally, the deterioration level of all the production lines must be properly adjusted upwards or downwards. All these assignments are captured in the following **Body** attribute of the procedure PrepareDataForNextRoll.

```
FirstDayInPlanningInterval += 7;

Stock(l,'past',s) :=
   Stock(l,FirstPeriodInPlanningInterval,s);
ProductionLineInUse(f,p,'past') :=
   ProductionLineInUse(f,p,FirstPeriodInPlanningInterval);

DeteriorationLevel(f,p) +=
   0.75 * ProductionLineInUse(f,p,FirstPeriodInPlanningInterval) + 0.25;
DeteriorationLevel( (f,p) |
   LineInMaintenance(f,p,FirstPeriodInPlanningInterval) ) := 0;
```

Updating deterioration level

Note that the deterioration level of a productive line is updated by 1 reflecting that the line was in use during the first period in the planning interval. Otherwise, the deterioration level is increased by only 0.25 to reflect that the line remained idle for that week. Of course, if a line is under maintenance during the first period, its deterioration level is reset to zero.

9.3.3 Rolling Procedures

Tree structure

Two rolling horizon procedures can be considered. One of them captures all the procedures needed to execute a single step in the rolling horizon process. You may execute this procedure sequentially by using the corresponding right-mouse action, and examine the results as they are found. The second procedure executes all the remaining single steps in one go. Please update the section `Rolling Procedures` in your tree structure as shown in Figure 9.17.

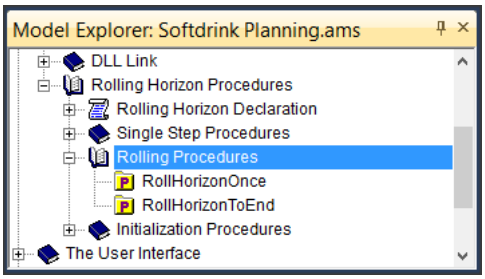


Figure 9.17: The structure of the `Rolling Procedures` section

Roll horizon once

The following sequence of statements carries out a single step in the rolling horizon process. Please enter them in the **Body** attribute of the procedure `RollHorizonOnce`. Note that each of the statements is a call to a procedure that was developed in the previous subsection.

```
LinkHorizonToCalendar;
ReadDemandDatafromDatabase;
DetermineMaintenance;
SolveLeastCostPlan;
RegisterInOverallPlanning(FirstWeekInPlanningInterval,FirstPeriodInPlanningInterval);
PrepareDataForNextRoll;
```

Rolling horizon to end

The following procedure completes the rolling horizon process starting from the current point in the calendar as determined by the element parameter `FirstWeekInPlanningInterval`. In the next subsection, you will encounter a procedure that will allow you to start the rolling horizon process from the beginning of the calendar. Please enter the following statements in the **Body** attribute of the procedure `RollHorizonToEnd`.

```
while ( LastWeekInPlanningInterval < LastWeekInCalendar ) do
    RollHorizonOnce;
endwhile;

for (t | t > FirstPeriodInPlanningInterval) do
    RegisterInOverallPlanning(WeekInPeriod(t),t);
endfor;
```

Complete overall planning

Note that the maintenance and line usage planning of the final planning interval is not only registered for the first period through the procedure `RollHorizonOnce`, but also for the remaining periods through the execution of the `for` statement.

9.3.4 Initialization procedures

Tree structure

There are three initialization procedures to be considered. One of them is the system-supplied procedure `MainInitialization` that is executed every time a project is started. The other two initialization procedures have been embedded in `MainInitialization`, but can also be called separately. Please update your tree structure as shown in Figure 9.18. Be sure not to create a `MainInitialization` procedure, because one is already present in your model. Simply move it from the end of the model tree to its desired position (using either the cut-and-paste or the drag-and-drop facility in AIMMS).

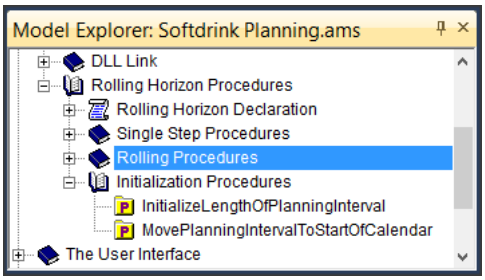


Figure 9.18: The structure of the `Initialization Procedures` section

Initializing Periods

In the procedure `InitializeLengthOfPlanningInterval`, two crucial parameters in the rolling horizon are set. Their values determine the amount of time considered in a single step of the rolling horizon process. You may change these values if you want to consider different planning intervals. Please enter the following statements into the **Body** attribute.

```
NumberOfPeriods                  := 10;
NumberOfPeriodsInPlanningInterval := 8;
```

Starting at the beginning

The procedure MovePlanningIntervalToStartOfCalendar first empties any existing overall maintenance and line usage planning, and then assigns all starting values known at the beginning of the calendar to the appropriate variables and parameters. This procedure can be called at any time, causing any activated rolling horizon procedures to start at the beginning of the calendar. Please enter the following statements into the **Body** attribute.

```
empty OverallMaintenancePlanning, OverallLineUsagePlanning;

Stock(l,'past',s)               := StockAtStartOfCalendar(l);
ProductionLineInUse(f,p,'past') := 1 onlyif ProductionLineLevelAtStartOfCalendar(f,p);

DeteriorationLevel(f,p)         := DeteriorationLevelAtStartOfCalendar(f,p);
FirstDayInPlanningInterval      := first( Mondays );
WeekInPeriod(t)                 := Element( Weeks, Ord(t) );
```

MainInitialization

The procedure MainInitialization, executed by AIMMS at the start of each run, is a natural starting point for reading data, initializing various parameters and starting other procedures that also initialize your model data. In this tutorial, the procedure MainInitialization reads essentially all the problem data from the database tables. The only exception is the demand data, which are read one section at a time for the current planning horizon from within the procedure RollHorizonOnce. Following this, the unit transport costs are obtained by calling the external function developed in Section 9.2. Finally, the data initialization required for the rolling horizon procedures is completed by calling the two procedures described above. Please replace the content of the MainInitialization procedure by the following statements.

```
ReadFromDatabase;
UnitTransportCost(f,c) := (ExternalUnitTransportCost(f,c)) [\$/TL];
InitializeLengthOfPlanningInterval;
MovePlanningIntervalToStartOfCalendar;
empty LengthDominatesNotActive, InactiveWeeks;
```

FormatString and unit casting

Note that the unit [$/TL] is attributed to the output of the external function. This requires you to place the parentheses around the function call as illustrated above.

9.4 Running the model

Closing your project

As indicated previously, the statements that you entered in the MainInitialization procedure are executed when the project is opened. Even though you could run this procedure directly using the right-mouse **Run Procedure** command, you may as well try out the default action by first closing the project and then re-opening it. To do so, execute the following steps to close your project:

- select the **Close Project** command from the AIMMS **File** menu,
- answer 'No' when being asked to compile your model before closing the project,
- answer 'No' in the dialog box that asks whether you want to save your data (see Figure 9.19),
- answer 'Yes' to save the changed project.

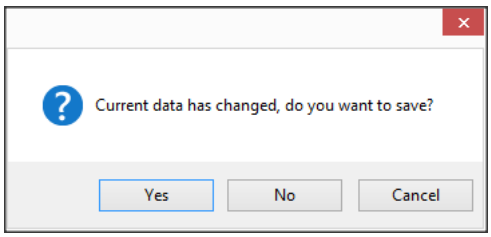


Figure 9.19: The **Save Changes** dialog box

Opening the project

Opening a project that you have just closed, is straightforward. AIMMS keeps track of the last five projects opened. Just select the 'Softdrink Planning' project from project list displayed in the AIMMS **Start Page**. Alternatively, you can select the recent project from the **File** menu (see Figure 9.20).

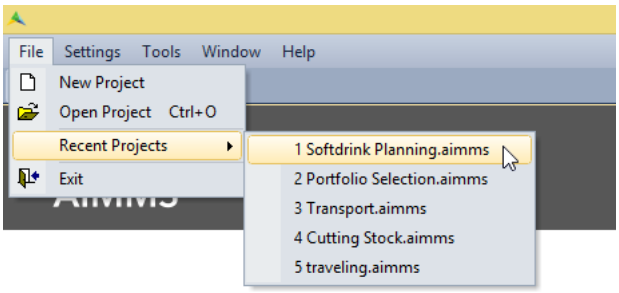


Figure 9.20: The **File** menu of AIMMS

Running a procedure

You are now ready to test the rolling horizon process starting from the beginning of the calendar. To run the procedure `RollHorizonOnce` you should perform the following actions:

- select the procedure `RollHorizonOnce` in the model tree, and
- in the right-mouse menu select the **Run Procedure** command (see Figure 9.21).

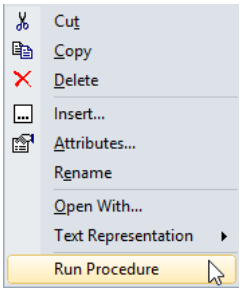


Figure 9.21: The right-mouse menu of the procedure RollHorizonOnce

Monitoring the progress

The **Progress** window lets you to monitor the progress made by AIMMS and the solver during the generation and solution of a mathematical program. By pressing the *Ctrl-P* key combination, the **Progress** window as shown in Figure 9.22 will appear. Once the solution has been found, AIMMS will again display warnings about data not yet initialized. These warnings can be ignored at this stage of the tutorial.

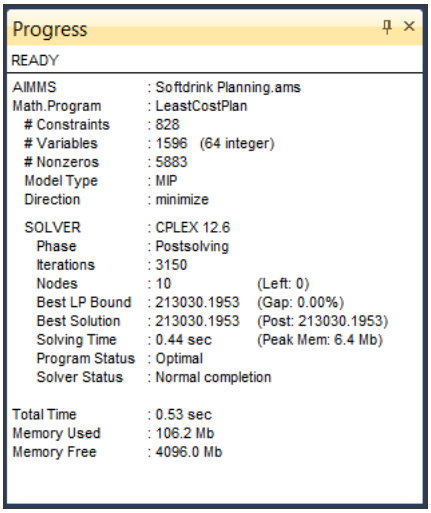


Figure 9.22: The **Progress** window

Viewing the solution

Once the procedure RollHorizonOnce has finished, you can view the results. For instance, you could open the data page associated with the variable TotalCost, and compare its value to the one in the **Progress** window in Figure 9.22. Similarly, you can inspect the value of any of the decision variables. For example, the optimal values for the variable Production are displayed in Figure 9.23

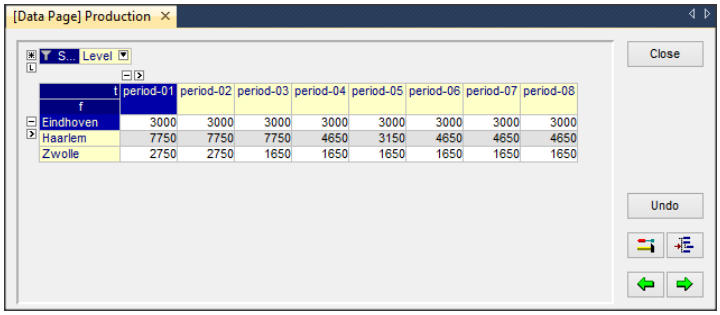


Figure 9.23: The data page of the variable `Production`

Additional information in a pivot table

By default AIMMS will display non-scalar data in a pivot table. For variables and constraints, additional information (e.g. marginal values, basic status) will also be shown in the pivot table when available. Notice that in the data page of the variable `Production` the basic status is displayed.

Ready for GUI building

At this point in the tutorial, you have reached a major milestone in that the complete model description of a rolling horizon application has been completed. In the next part of this tutorial, you will concentrate on building a graphical user interface for the end-user of this application.

Part IV

Building an End-User Interface

Chapter 10

Management of Pages and Templates

This chapter

Following this chapter, you will set up the structure of your end-user interface using the **Page Manager**. In addition, you will specify the style of your end-user interface using the **Template Manager**. At the end of this chapter you will make a startup page that will contain references to all the other pages.

Iterative design process

Designing an effective end-user interface is an iterative process that requires interaction with the end-users. Chapter 12 of the *The User's Guide* contains several design principles. In this tutorial, however, you will build the specified interface without any redesign.

10.1 Page management

The AIMMS **Page Manager**

In AIMMS, *pages* correspond to windows of information visible to the end-user. Pages are managed using the **Page Manager**, which allows you to organize all your end-user windows in a tree-like fashion. The organization of pages in the page tree defines the navigation structure of the end-user interface. Relative to a particular page in the page tree, the positions of the other pages define relationships such as *parent* page, *child* page, *next* page or *previous* page, which can be used with navigation controls such as buttons and menus. Figure 10.1 shows the navigation structure that you will use in your end-user application.

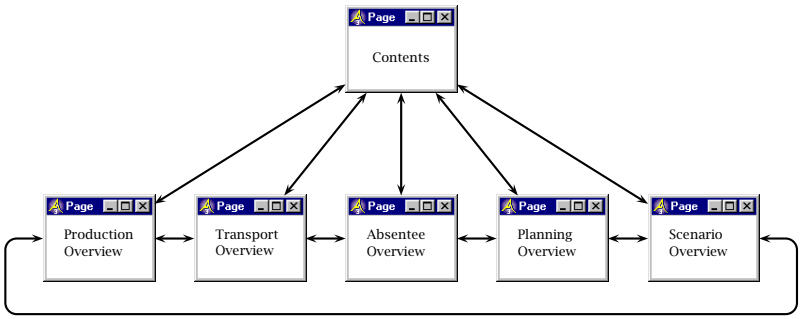


Figure 10.1: The navigation structure to be implemented

Opening the **Page Manager**

To create the desired page structure, you should first open the **Page Manager** by selecting it from the AIMMS **Tools** menu, or alternatively by pressing the *F9* key. A page tree is shown in Figure 10.2. Note that the trial page created in Chapter 5 was automatically added to the **Page Manager**. If you previously saved a changed Data Page, a parent page named 'All Data Pages' is added as well, containing the saved Data Page.

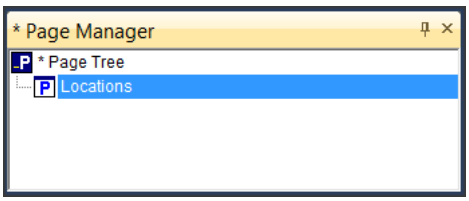


Figure 10.2: A **Page Manager** with one page

Creating a new page

You have already created a new page in Chapter 5:

- press the **New Page** button on the toolbar to create a new page, or alternatively press the *Insert* key,
- specify 'Contents' as the name of this new page, and
- press the *Enter* key to register the page.

Creating a child page

To create a child page of the *Contents* page you should execute the following steps:

- open the *Contents* page by double-clicking on its icon,
- press the **New Page** button on the toolbar to create a new page,
- specify 'Production Overview' as the name of this new page, and
- press the *Enter* key to register the page.

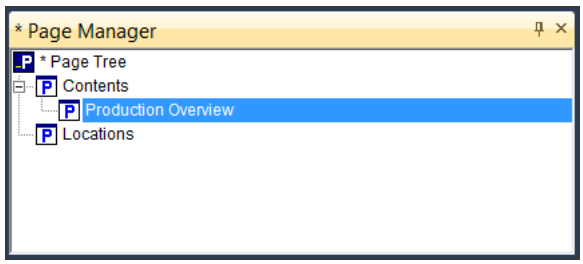


Figure 10.3: The intermediate page tree

You should now complete the structure of the page tree to match Figure 10.4.

Completing the page navigation structure

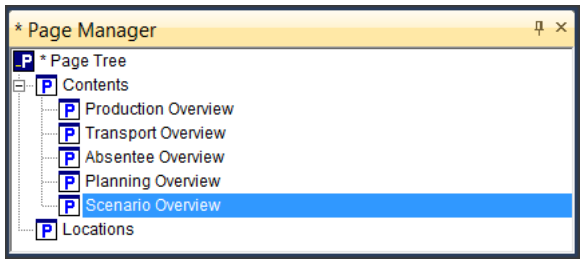


Figure 10.4: The final page navigation structure

Saving your changes

The asterisk at the left side of the title bar indicates that changes to your project have not yet been saved to disk. Save your work by pressing the **Save Project** button on the toolbar, or alternatively, pressing the *Ctrl-S* key combination.

Describing the six pages

The intended contents of each of the six pages are described below.

- **Contents:** The *Contents* page will be created as a means of navigating to the other pages.
- **Production Overview:** The *Production Overview* page will contain the optimal production levels and maintenance schedule for the current planning interval.
- **Transport Overview:** The *Transport Overview* page will contain the optimal transport values for the factories and centers plus their corresponding stock levels for the current planning interval.
- **Absentee Overview:** The *Absentee Overview* page will provide an interactive facility to specify holidays and vacation periods in a convenient manner.
- **Planning Overview:** The *Planning Overview* page will display the overall production and maintenance planning for the portion of the entire calendar under consideration.
- **Scenario Overview:** The *Scenario Overview* page will display the demand figures for the different scenarios in the database.

10.2 Template management

The AIMMS **Template Manager**

Using the **Template Manager**, you can ensure that all end-user pages are the same size and possess the same look and feel. You can accomplish this effect by creating so-called page templates, which define page properties and objects common to a group of end-user pages. These page templates can be nested inside the tree of page templates. In addition, you need to position all your

end-user pages as child pages beneath the page templates so that the objects on the template pages become visible on the end-user pages.

Common page components

Typical page objects and page properties that are inherited by end-user pages from page templates are:

- background color or background bitmap,
- a logo,
- navigation buttons,
- page menubar and toolbar,
- header and footer areas, and
- page size and resize behavior.

In this tutorial exercise, there will be one template for the background color, and one template containing shared navigation buttons.

Opening the **Template Manager**

To create the desired page templates you should first open the **Template Manager** by selecting it from the AIMMS **Tools** menu, or alternatively by pressing the *Alt+F9* key. The initial template tree is shown in Figure 10.5. Note that the initial template tree automatically contains all the pages that you previously created inside the **Page Manager**.

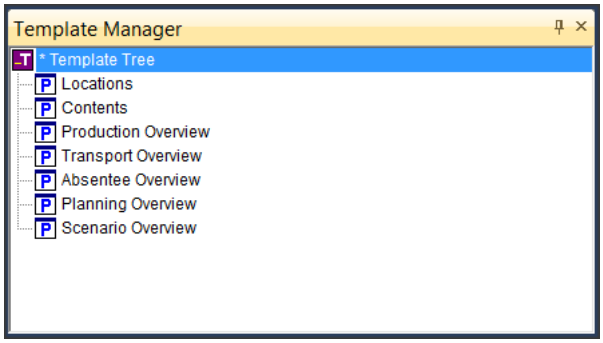


Figure 10.5: The **Template Manager** with initial template tree

Creating two page templates

Next, you need to create one page template for the background color and one for the navigation buttons:

- select the root node in the template tree,
- press the **New Template** button on the toolbar,
- specify 'Background Bitmap' as the name of this new template, and
- press the *Enter* key to register the template.

Position the second page template as a child of the first page template as shown in Figure 10.6:

- open the *Background Bitmap* template by double-clicking on its icon,
- press the **New Template** button on on the toolbar,
- specify 'Navigation Buttons' as the name of this new template, and
- press the *Enter* key to register the template.

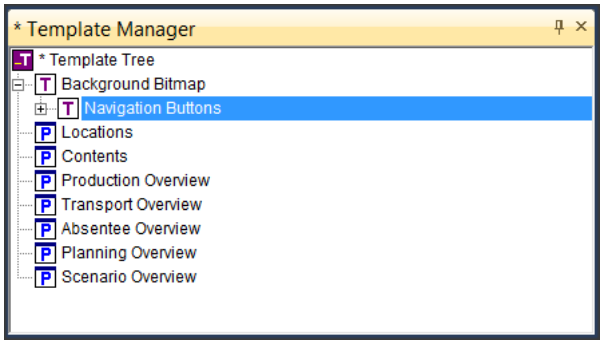


Figure 10.6: The **Template Manager** with intermediate template tree

Moving pages underneath templates

The six pages created in the **Page Manager** appear automatically in the **Template Manager**. You should move the *Contents* page so that it inherits the bitmap background as indicated in Figure 10.7:

- select the *Contents* page in the template tree, and
- drag the page to the *Background Bitmap* template.

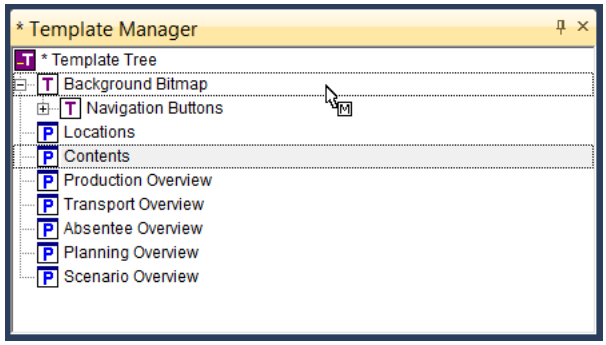


Figure 10.7: The **Template Manager** while moving the *Contents* page

Next, you should move the remaining five overview pages so that they inherit both the bitmap background and the navigation buttons as illustrated in Figure 10.8:

- open the *Navigation Buttons* template by double-clicking on its icon,

- select all five overview pages in the template tree using the *Shift* key together with the mouse, and
- drag the selected pages to below the *Navigation Buttons* template.

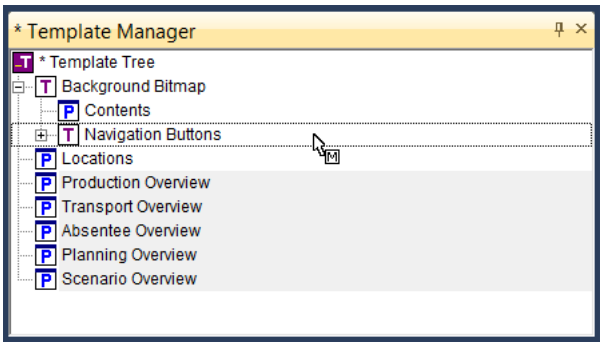


Figure 10.8: The **Template Manager** while moving overview pages

Final template tree

The final template tree should be as shown in Figure 10.9.

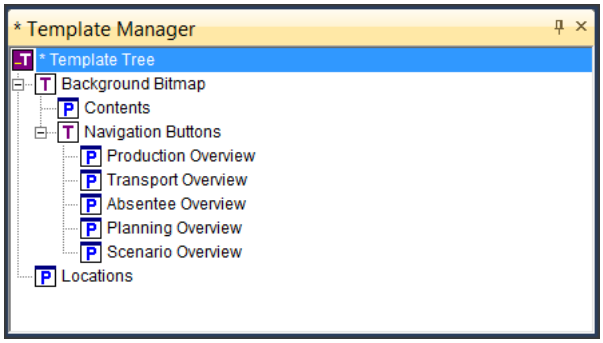


Figure 10.9: The **Template Manager** after moving pages

Background bitmap specification

The *Background Bitmap* template is designed to provide a uniform background for your entire end-user interface. You can specify this template in the following manner:

- select the *Background Bitmap* template in the template tree,
- open the template by clicking on the **Open in Edit Mode** button on the toolbar,
- select the **Picture** command from the **Object** menu,
- position the mouse cursor at the upper left corner of the template,
- depress the left-mouse button and drag the mouse cursor to the lower right corner of the template, and
- release the mouse button.

At this point you need to complete the **Picture Properties** dialog box:

- press the **Wizard** button on the right of the 'File Name' edit field,
- select the **Select File Name**... command in the right-mouse pop-up menu,
- select the bitmap file '`Bitmaps\Background.bmp`',
- press the **Open** button,
- select the 'Fill with Multiple Pictures' display option, and
- press the **OK** button.

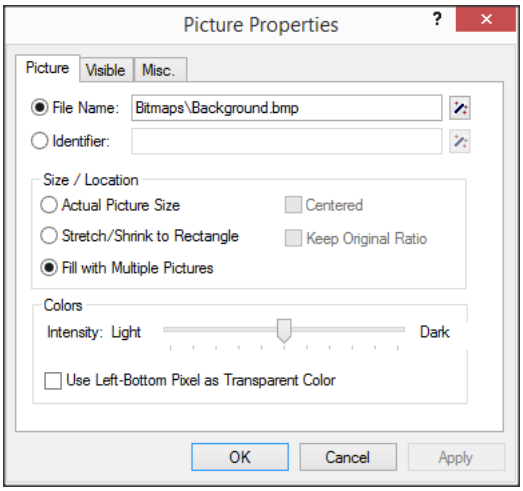


Figure 10.10: The **Picture Properties** dialog box

Viewing the result

By selecting the option 'Fill with Multiple Pictures', as shown above in Figure 10.10, you instruct AIMMS to replicate the small bitmap contained in the file '`Background.bmp`'. As a result, the entire screen should now be filled with a blue pattern as displayed in Figure 10.11.

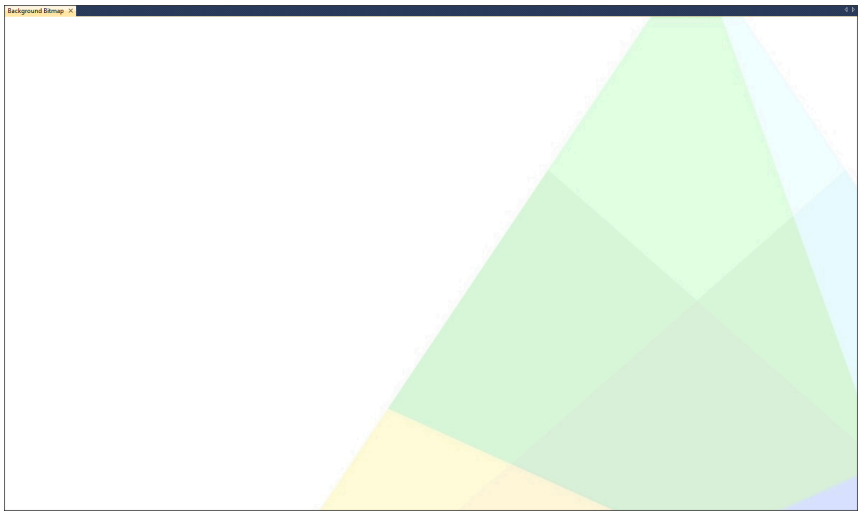


Figure 10.11: The *Background Bitmap* template

Saving your changes

The asterisk on the left of the title bar on the template page indicates that additions to your project have not yet been saved to disk. Save your work by pressing the **Save Project** button on the toolbar.

Verifying template inheritance

You can check whether the *Background Bitmap* template is correctly inherited by performing the following actions:

- press the *F9* key to open the **Page Manager**, and
- open, for instance, the *Production Overview* page by double-clicking on it.

The *Production Overview* page should look the same as the *Background Template* page. Once you have verified this action, you may close this page by clicking the cross at the upper right corner of the page.

Navigation buttons

The second template provides a dedicated area with navigation buttons for the overview pages. You will place three buttons for easy access to:

- the next page,
- the previous page, and
- the contents page.

Creating a 'Next Page' button

To create a button that allows you to go to the next page with a single click, you should perform the following actions:

- open the *Navigation Buttons* template in **Edit** mode,
- press the **New Button** button on the toolbar,

- use the mouse to draw a small rectangle at the lower right corner of the page,
- select the 'Bitmap Button' option in the **Button Properties** dialog box,
- use the wizard to select the **Select File Name**... command from the right-mouse pop-up menu,
- select the file '`Bitmaps\Button Next.bmp`', and
- press the **OK** button.

Next, you need to open the **B**utton Properties dialog box again and complete the **Actions** tab as shown in Figure 10.12.

- select the **Actions** tab,
- select a 'Goto Page' action,
- press the **Add** button which selects the default 'Go to Previous Page' action,
- select the 'Next Page' option,
- press the **Apply** button to get the new 'Go to Next Page' action, and
- press the **OK** button.

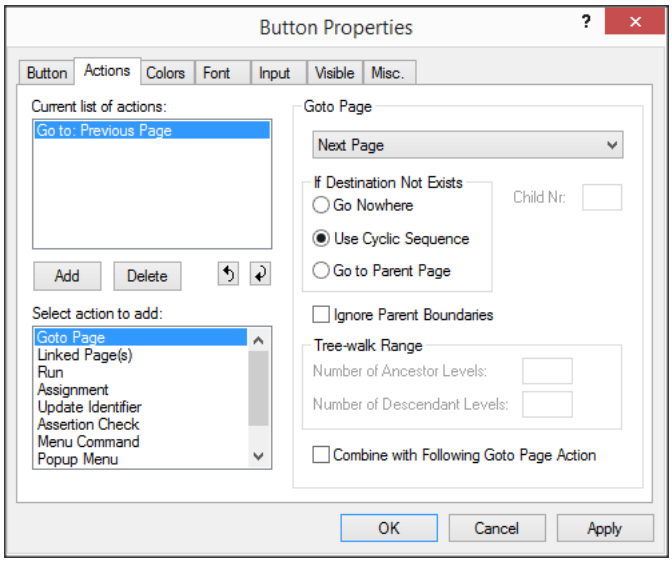


Figure 10.12: The **Button Properties** dialog box

Inspecting the button

On your screen you should see a button containing a small grey box. By pressing the **Page User Mode** button on the left of the tool bar, the grey box changes into the bitmap with an arrow pointing to the right. By again pressing the **Page Edit Mode** button on the left of the tool bar, you are back in object **Edit** mode and can create the remaining two buttons as shown in Figure 10.13.

Creating the remaining two buttons

The bitmap on the button with the left arrow corresponds with the bitmap file 'Bitmaps\Button Prev.bmp'. This button reflects the action 'Go to Previous Page'. The remaining button corresponds with the file 'Bitmaps\Button Up.bmp', and reflects the action 'Go to Parent Page'. Again, you can inspect the three buttons by changing into **User** mode as described in the previous paragraph.

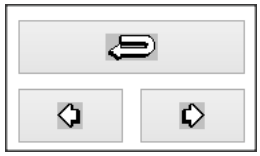


Figure 10.13: The three buttons on their page template

10.3 The Contents page

Referencing the overview pages

The *Contents* page is the parent page in the hierarchy of pages within the **Page Manager**. From this page you should be able to reference each of the five overview pages. For this purpose, AIMMS provides you with a so-called navigation object. The contents of such a navigation object can change dynamically depending on the page structure in the **Page Manager**.

Creating a navigation object

To create a new navigation object on the *Contents* page you should perform the following steps:

- open the *Contents* page,
- make sure that this page is in **Edit** mode,
- press the **New Navigation Object** button on the toolbar,
- use the mouse to draw a rectangle in the center of the page, and
- press the **OK** button.

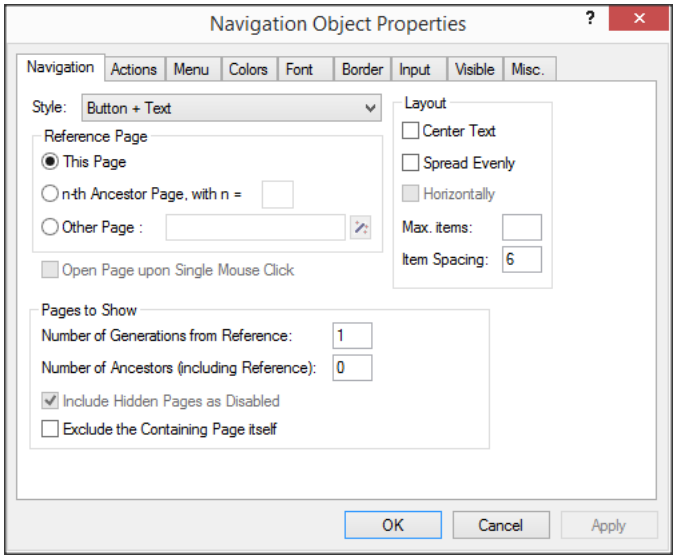


Figure 10.14: The **Navigation Object Properties** dialog box

Default settings

As you can see in Figure 10.14, the default settings in the **Navigation Object Properties** dialog box are such that only child pages of the current reference page will be shown. By changing the 'Number of Generations from Reference' parameter and/or the 'Number of Ancestors (including Reference)' parameter, you can adjust the contents of the navigation object.

Changing the font

You might have thought that the default font size in the navigation object is rather small. To change the font size you should open the **Navigation Properties** dialog box using either the right-mouse to select **Properties**... command, or clicking on the **Properties** button on the tool bar. Once you are in the dialog box, you should execute the following steps:

- select the **Font** tab,
- press the **Add** button,
- select 'Bold' as the 'Font Style',
- select '20' as the 'Font Size',
- press the **OK** button,
- specify 'Navigation Object Font' as the name of the new font, and
- press the **OK** buttons.

The font selections are shown in Figure 10.15, and they should be visible in the navigation object on your screen.

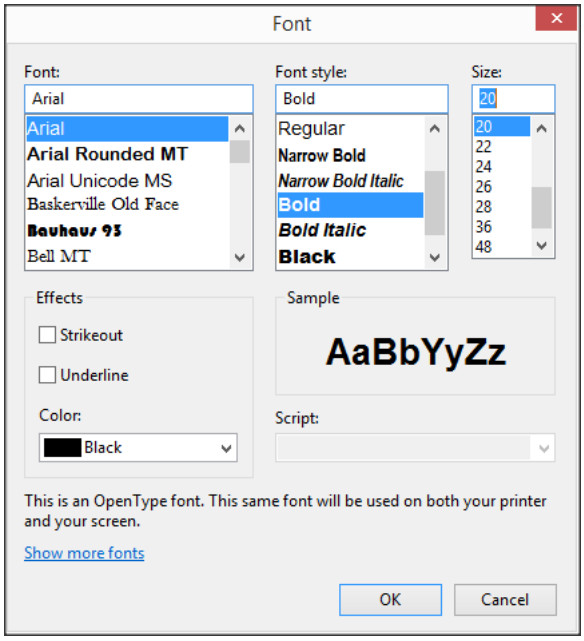


Figure 10.15: The **Font** dialog box

Changing the color

Figure 10.16 indicates how to set the foreground color to navy blue. Please execute the following steps.

- re-open the **Navigation Properties** dialog box,
- select the **Colors** tab,
- select 'Transparent' in the dropdown list of the background color
- select 'User' as the provider of the foreground color,
- set the foreground color to navy blue, and
- press the **OK** button.

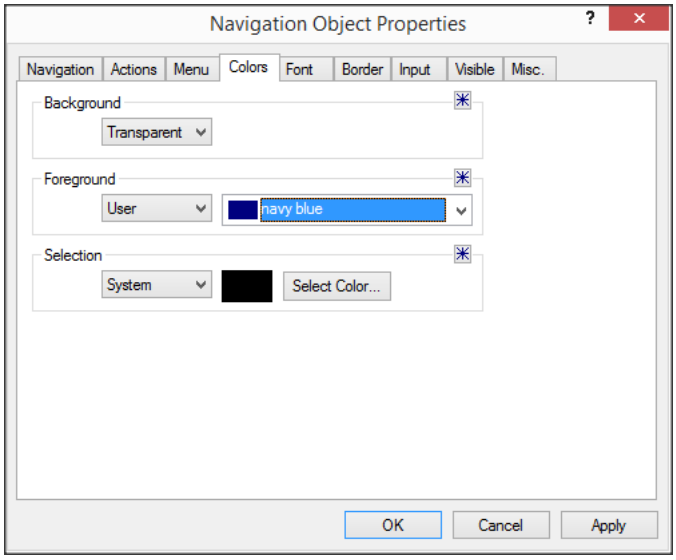


Figure 10.16: The **Colors** tab of the **Navigation Properties** dialog box

Putting a logo on the page

In many applications you will want to put a logo on a page. In this tutorial the AIMMS logo will be used by executing the following steps:

- open the *Contents* page in edit mode,
- select the **Picture** command from the **Object** menu,
- use the mouse to draw a rectangle in the upper right corner of the page,
- press the **Wizard** button to the right of the 'File Name' edit field,
- select the **Select File Name** command from the right-mouse pop-up menu,
- select the file '`Bitmaps\AIMMS Logo.bmp`' in the **Picture Properties** dialog page,
- press the **Open** button to return to the **Picture Properties** dialog box, and
- press the **OK** button.

The *Contents* page should now look like the one shown in Figure 10.17.

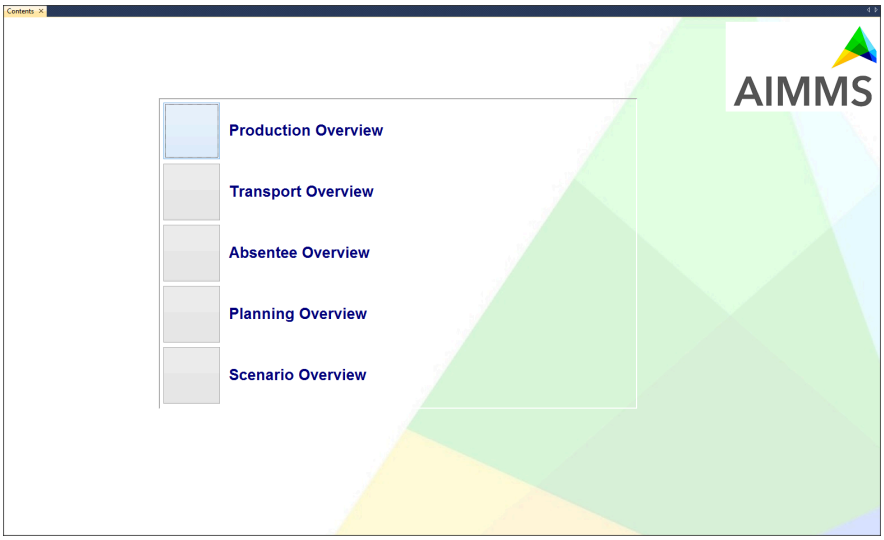


Figure 10.17: The *Contents* page

Testing the initial interface

Once you have pressed the **Page User Mode** button, you can press any of the five buttons on the *Contents* page. AIMMS will automatically open the corresponding child page. You can then use the 'Previous', 'Next' or 'Up' buttons to navigate to another page.

Specifying a startup page

In AIMMS you can specify a startup page. This page is automatically shown when the underlying application is opened. To make the *Contents* page the default startup page of your application, you should execute the following actions:

- select the **Project Options** command from the **Settings** menu,
- set the 'Startup page' as shown in Figure 10.18, and
- press the **OK** button.

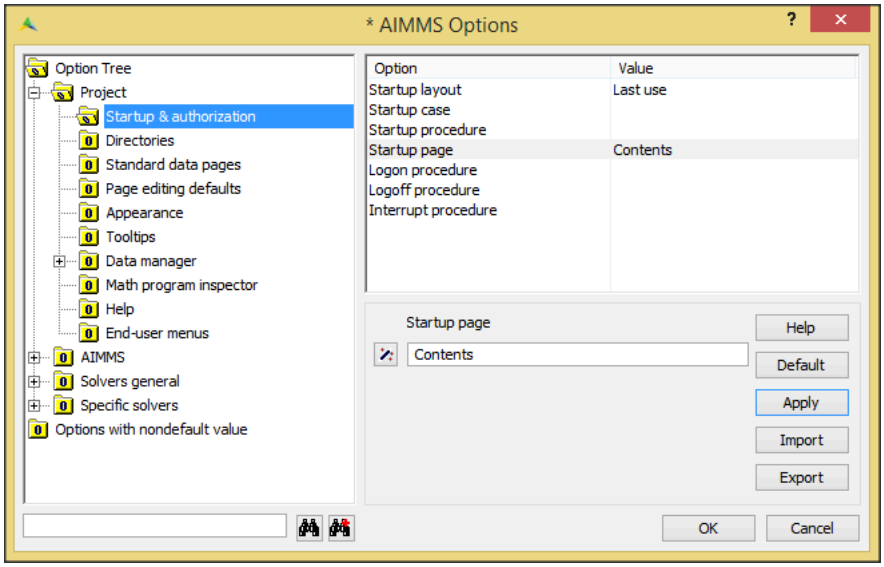


Figure 10.18: The **AIMMS Options** dialog box

Saving your changes

The asterisk at the left of the title bar of the AIMMS window indicates that recent changes to your project have not yet been saved to disk. Save your work by pressing the **Save Project** button on the toolbar.

Closing and re-opening the project

After having saved your project, you can close and subsequently re-open the project to verify that the *Contents* page is displayed automatically. The process of closing and re-opening a project has already been discussed in detail at the end of Chapter 9.

Chapter 11

Production and Transport Overviews

This chapter

In this chapter you will build two end-user pages that display the solution corresponding to a single 'roll' in the rolling horizon process. The first page, the *Production Overview* page, concentrates on the optimal production and maintenance schedule for every period in the current planning horizon. The second page, the *Transport Overview* page, provides not only the optimal transport patterns from the factories to the distribution centers, but also the corresponding stock overviews for all locations considered.

11.1 Extending the model tree

Needing additional identifiers

Whenever you build a professional user interface, it is quite natural to introduce additional identifiers to support such an interface. For instance, an element parameter defined over the predefined set of `AllColors` can be used to change the color of numbers when they drop below a particular threshhold value. Another possibility is the introduction of parameters to control the scrolling mechanism of a Gantt chart. Yet another option is an identifier to control whether or not a particular object appears at all depending on data elsewhere in your application.

Introducing extra model sections

You should now introduce five extra sections in your model tree corresponding to the five end-user overview pages already introduced in the **Page Manager**. All new page-specific identifiers introduced can then be inserted into the appropriate section. The updated tree structure is shown in Figure 11.1.

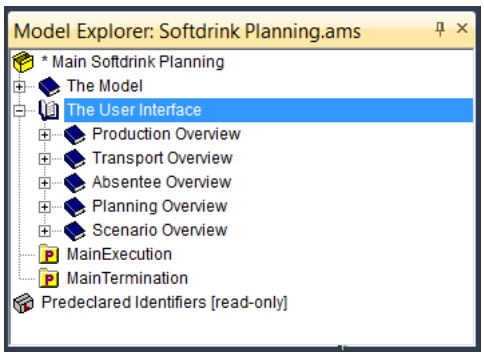


Figure 11.1: Subdividing the `The User Interface` section

11.2 The Production Overview page

In this section you will construct the entire page as shown in Figure 11.2. Each page object will be treated in a separate subsection.

Viewing the entire page

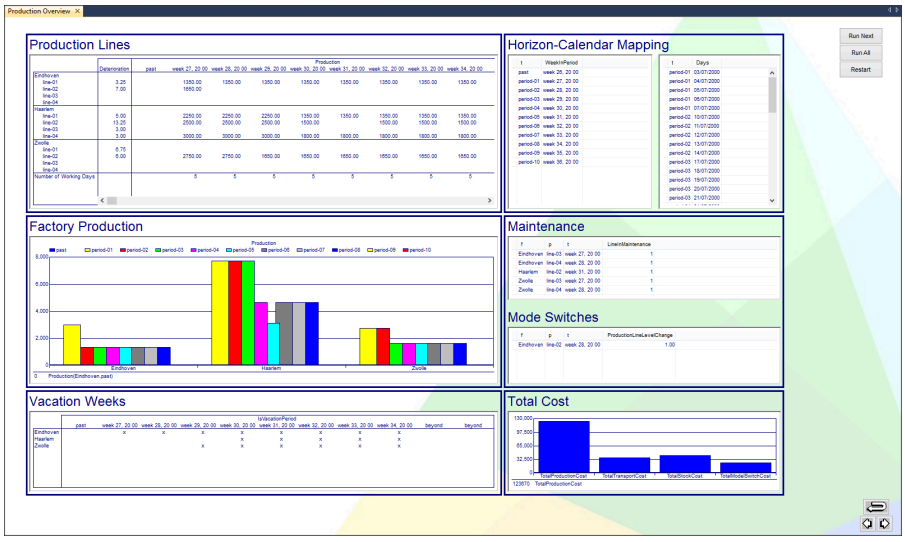


Figure 11.2: The completed *`Production Overview`* page

11.2.1 Execution buttons

The **Run Next** *button*

The first execution button you will add is designed to execute a single step in the rolling horizon process. This allows you to track the behavior of the model step by step. To create the **Run Next** button you should perform the following actions:

- open the *Production Overview* page in **Edit** mode,
- press the **New Button** button on the toolbar,
- drag and create a small rectangle in the upper right corner of the page,
- specify "Run Next" (with the quotes) in the 'Title' edit field,
- press the **Actions** tab,
- select the 'Run' action,
- press the **Add** button,
- select the 'Procedure' option (not the 'Page Procedure' option),
- use the **Wizard** button to select the procedure `RollHorizonOnce`,
- press the **Finish** button, and
- press the **OK** button.

The **Run All** *button*

The second execution button to be added is designed to execute the entire rolling horizon process from the current point forward. Just repeat the steps in the previous paragraph while creating the **Run All** button, but select the procedure `RollHorizonToEnd`.

The `Restart` *button*

The third execution button is the **Restart** button which activates the procedure `MovePlanningIntervalToStartOfCalendar`. Following the execution of this procedure you can use either of the previous two execution buttons to execute part or all of the rolling horizon process. Instead of creating the button from scratch, as in the previous two paragraphs, you could use the 'copy and paste' facility as described in the following steps:

- in **Edit** mode, select the **Run All** button by clicking on it,
- press the **Copy** button on the toolbar,
- press the **Paste** button on the toolbar (the mouse cursor will change as shown in Figure 11.3) ,
- use the mouse cursor to position the new button underneath the **Run All** button,
- click the left-mouse button to confirm the position of the new button,
- double-click the left-mouse button to open the **Button Properties** dialog box of the new button, and
- modify the button properties as appropriate.

Figure 11.3: The mouse cursor after having pressed the **Paste** button

11.2.2 The production lines table

Three identifiers in one table

In the first table on the *Production Overview* page you will include three identifiers, namely:

- the actual production level by factory, production line and time period,
- the number of working days in each week, and
- the current deterioration level associated with each production line.

Actual production level

The actual level of production will be equal to potential production whenever a production line is in use. Create a new declaration section `Production Overview Declaration` in the `Production Overview` section, and insert the following parameter declaration:

```
Parameter ActualProduction {
    IndexDomain : (f,p,t);
    Unit        : hl;
    Definition  : PotentialProduction(f,p,t)*ProductionLineInUse(f,p,t);
}
```

Creating a table

The first part of the table can be created by executing the following steps:

- ensure that the *Production Overview* page is in **Edit** mode,
- press the **New Table** button on the toolbar,
- drag and create a rectangle that matches the desired table size on your page,
- in the **Identifier** wizard select the parameter `ActualProduction(f,p,t)`,
- press the **Next** button, and
- press the **Finish** button.

Adding an identifier

To add the identifier `DeteriorationLevel(f,p)` as the first column of this new table you should perform the following actions:

- select the existing table object,
- press the **Properties** button on the toolbar,
- select the **Contents** tab,
- press the **Add** button,
- select the identifier `DeteriorationLevel(f,p)` using the **Identifier** wizard,
- press the **Next** button,

- uncheck the 'Automatic split row/column' checkbox,
- select the 'split line' entry that pops up in the listbox (see Figure 11.4),
- press the **Down** button,
- press the **Finish** button,
- press the **Up** button to display the identifier `DeteriorationLevel` as the *first* column, and
- press the **OK** button.

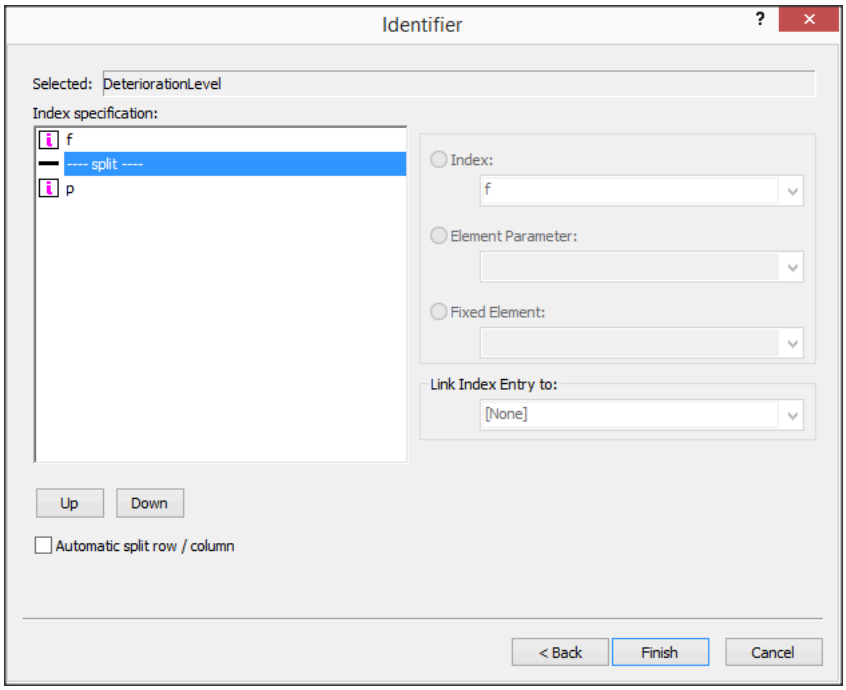


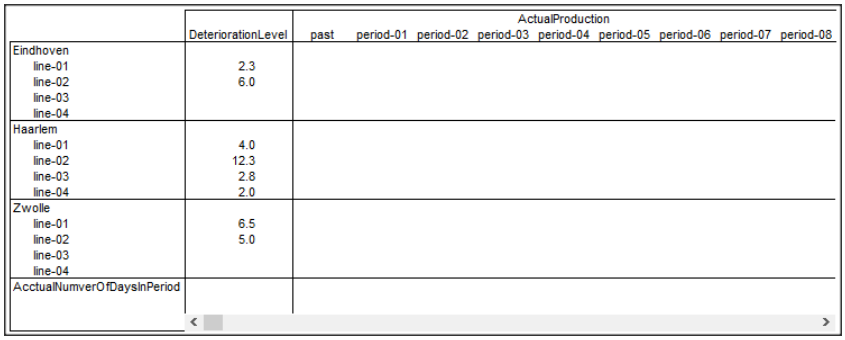
Figure 11.4: Specifying the row and column domain

Moving the split line

If you had not moved the split line, AIMMS would have used the index `f` for rows and the index `p` for columns. However, by moving the split line, both indices can be used as row indices conforming to Figure 11.5.

Adding another identifier

Following the routine specified above, you should now add the identifier `ActualNumberOfDaysInPeriod(t)` as a new row in the table. The table on your screen should then look like the one shown in Figure 11.5.



	DeteriorationLevel	ActualProduction past	period-01	period-02	period-03	period-04	period-05	period-06	period-07	period-08
Eindhoven										
line-01	2.3									
line-02	6.0									
line-03										
line-04										
Haarlem										
line-01	4.0									
line-02	12.3									
line-03	2.8									
line-04	2.0									
Zwolle										
line-01	6.5									
line-02	5.0									
line-03										
line-04										
AcctualNumverOfDaysInPeriod										

Figure 11.5: The initial production overview table

Creating week labels ...

The 'period' references in the table are somewhat abstract and not meaningful. In AIMMS you can change these references using a string parameter. You should first create this string parameter in the section `Production Overview Declarations`.

```
StringParameter PeriodDescription {
    IndexDomain : t in Periods;
    Definition  : {
        if ( t in Periods.past) then
            "past"
        elseif ( t in Periods.beyond) then
            "beyond"
        else
            FormatString("%e", WeekInPeriod(t))
        endif
    }
}
```

The predefined function `FormatString` allows you to compose a string that is built up from a combination of numbers, strings and set elements (see Chapter 5 of *The Language Reference*).

... as part of the table

The above string parameter `PeriodDescription(t)` can be used as element text in the table after executing the following steps:

- open the **Table Properties** dialog box of the table,
- select the **Element Text** tab (see Figure 11.6),
- select the index `t`,
- press the **Modify** button,
- select the identifier `PeriodDescription(t)`,
- press the **Next** button,
- press the **Finish** button, and
- press the **OK** button.

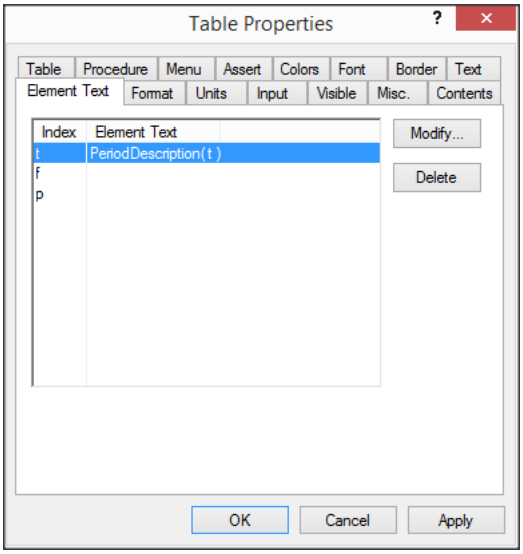


Figure 11.6: The **Element Text** tab of the **Table Properties** dialog box

Viewing the result

If the table does not show the constructed period descriptions, and you receive an initialization warning, you should press the **Run Next** button once and the period descriptions should then appear.

Specifying the number format ...

AIMMS chooses a default number format when displaying identifiers in a table. However, you might want to change the number of digits and/or the number of decimals. For example, the parameter `ActualNumberOfDaysInPeriod` should be an integer, and the values of the parameter `ActualProduction` are too large for the default format.

... first for actual production

You can execute the following steps to change the number format of `ActualProduction` to a width of 8 digits with 2 decimals:

- open the **Table Properties** dialog box of the table,
- select the **Format** tab (see Figure 11.7),
- select the element `ActualProduction(f,p,t)` from the drop-down listbox,
- enter the number '8' (without quotes) in the 'Width' field,
- enter the number '2' (without quotes) in the 'Decimals' field, and
- press the **Apply** button.

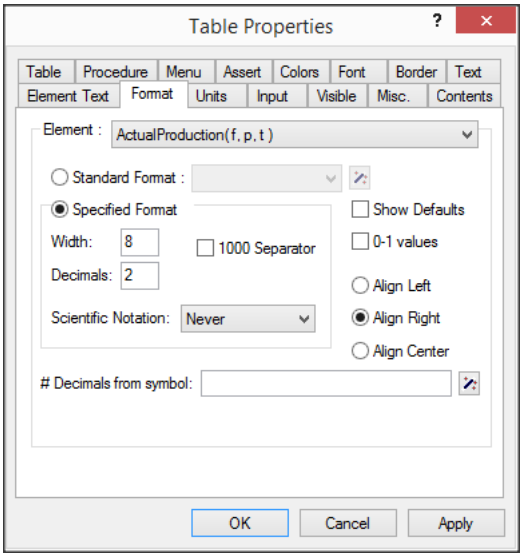


Figure 11.7: The **Format** tab of the **Table Properties** dialog box

... and then for the other two identifiers

Next, you should change the format of the parameter `DeteriorationLevel` to a width of 5 with 2 decimals, and also adjust the number format of the parameter `ActualNumberOfDaysInPeriod` to a width of 5 with 0 decimals. An instance of the completed table is shown in Figure 11.8.

	Deterioration	Production past	week 27, 20 00	week 28, 20 00	week 29, 20 00	week 30, 20 00	week 31, 20 00	week 32, 20 00	week
Eindhoven									
line-01	2.25		1350.00	1350.00	1350.00	1350.00	1350.00	1350.00	
line-02	6.00		1650.00	1650.00	1650.00	1650.00	1650.00	1650.00	
line-03									
line-04									
Haarlem									
line-01	4.00		2250.00	2250.00	2250.00	1350.00	1350.00	1350.00	
line-02	12.25		2500.00	2500.00	2500.00	1500.00		1500.00	
line-03	2.75								
line-04	2.00		3000.00	3000.00	3000.00	1800.00	1800.00	1800.00	
Zwolle									
line-01	6.50								
line-02	5.00		2750.00	2750.00	1650.00	1650.00	1650.00	1650.00	
line-03									
line-04									
Number of Working Days			5	5	5	5	5	5	

Figure 11.8: The completed production line table

11.2.3 The factory production bar chart

Creating a bar chart

The production lines table displays a production overview for each individual production line. The following bar chart will provide a similar overview at the factory level. To create this bar chart you should perform the following actions:

- make sure that the *Production Overview* page is opened in **Edit** mode,
- press the **New Bar Chart** button on the toolbar,
- drag and create a rectangle underneath the *Production Lines* table with the same width, and
- select the variable `Production(f,t)` using the **Identifier** wizard.

Creating week labels

As before, you should change the abstract period references into week references using the string parameter `PeriodDescription`. The resulting bar chart is shown in Figure 11.9.

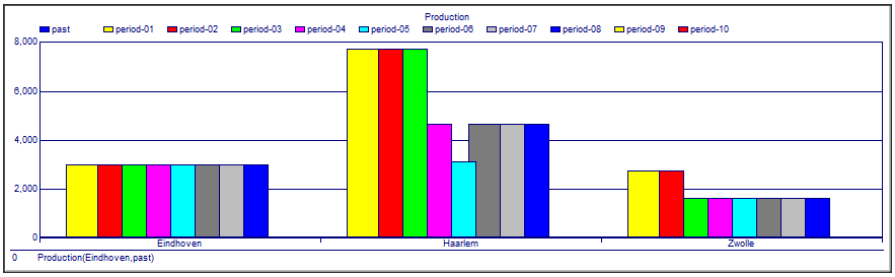


Figure 11.9: The completed factory production bar chart

11.2.4 The vacation table

Creating the table

The created table will display all the weeks that correspond to a vacation period with a 40% drop in production. To create this table you should complete the following sequence of steps:

- make sure that the *Production Overview* page is in **Edit** mode,
- press the **New Table** button on the toolbar,
- drag and create a rectangle below the factory production bar chart with the same dimensions,
- select the parameter `IsVacationPeriod(f,t)` using the **Identifier** wizard, and
- change the element text of the index `t` to the string parameter `PeriodDescription(t)`.

Displaying nonzero values as crosses

The identifier `IsVactionPeriod(f,t)` is a binary parameter. A value of zero means 'no vacation period', while a value of one indicates a 'vacation period'. The chosen value of one is somewhat arbitrary, and for this reason you might prefer to display a cross instead of a one. This minor modification can be accomplished as follows:

- open the **Table Properties** dialog box of the table,
- select the **Format** tab (see Figure 11.10),

- check the '0–1 values' check box, and
- press the **OK** button.

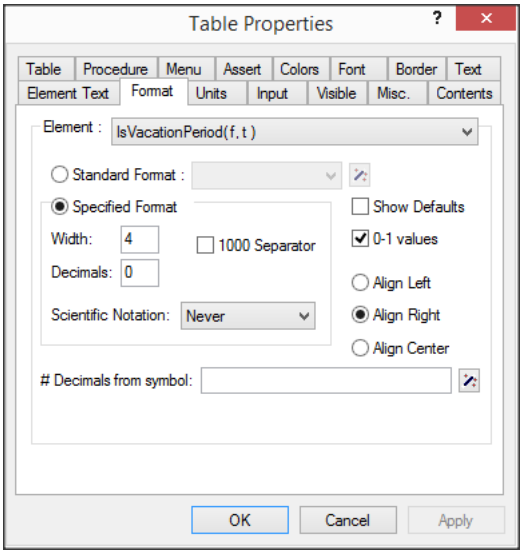


Figure 11.10: The **Format** tab of the **Table Properties** dialog box

Note that at this point the table is still empty since no vacation weeks have yet been specified. Later, you will specify these vacation weeks using a Gantt chart object on the *Absentee Overview* page.

11.2.5 The horizon-calendar tables

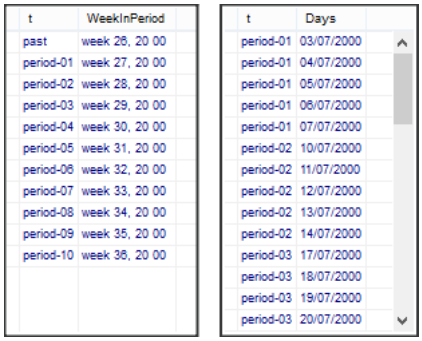
Creating your first composite table

In this subsection you will create two composite tables that establish the relationship between the abstract horizon periods and the weekly and daily calendar periods. Composite tables in AIMMS resemble the structure of relational database tables, and you can adjust the width of columns from within the graphical interface. To create your first composite table, you should execute the following steps:

- press the **New Composite Table** button on the toolbar,
- draw a rectangle on the page,
- select the parameter `WeekInPeriod(t)`,
- press the **Next** button, and
- press the **Finish** button.

Creating the second table

For the second composite table you should select the indexed set `DaysInPeriod(t)`. The two composite tables should look similar to the ones shown in Figure 11.11.



t	WeekInPeriod
past	week 26, 20 00
period-01	week 27, 20 00
period-02	week 28, 20 00
period-03	week 29, 20 00
period-04	week 30, 20 00
period-05	week 31, 20 00
period-06	week 32, 20 00
period-07	week 33, 20 00
period-08	week 34, 20 00
period-09	week 35, 20 00
period-10	week 36, 20 00

t	Days
period-01	03/07/2000
period-01	04/07/2000
period-01	05/07/2000
period-01	06/07/2000
period-01	07/07/2000
period-02	10/07/2000
period-02	11/07/2000
period-02	12/07/2000
period-02	13/07/2000
period-02	14/07/2000
period-03	17/07/2000
period-03	18/07/2000
period-03	19/07/2000
period-03	20/07/2000

Figure 11.11: The mapping between horizon and calendars

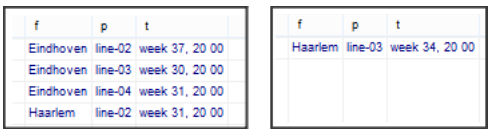
11.2.6 The maintenance and mode switches tables

Creating the maintenance table

As with vacation periods and holidays, maintenance periods also cause a decrease in production. Therefore, a maintenance overview can also contribute to the interpretation of the results in the production line table and factory production bar chart. By now you should be able to create the maintenance table without guidance. This composite table needs only the identifier `LineInMaintenance(f,p,t)` as its domain, and the table will immediately contain the required three columns. To complete the table you should again change the abstract period references by specifying that the string parameter `PeriodDescription(t)` is used as the element text of the index `t` (as you did previously).

Creating the mode switch table

The last composite table on the *`Production Overview`* page will display all the optimal mode switches for the current planning horizon. It can be specified in the same way as the table in the previous paragraph. The identifier `ProductionLineLevelChange(f,p,t)` is used to specify the domain of the table. The two composite tables are shown in Figure 11.12.



f	p	t
Eindhoven	line-02	week 37, 20 00
Eindhoven	line-03	week 30, 20 00
Eindhoven	line-04	week 31, 20 00
Haarlem	line-02	week 31, 20 00

f	p	t
Haarlem	line-03	week 34, 20 00

Figure 11.12: The maintenance (left) and mode switch (right) tables

11.2.7 The total costs bar chart

Declaring auxiliary parameters

The final data object on this page will display the four cost components that together determine the overall total cost, in an aggregated way. As of yet, there are no identifiers that contain the values of these four components. Therefore, you must first declare four new parameters describing the aggregated production, transport, stock and mode-switch costs which are to be placed at the end of the `Production Overview` section. Note that the aggregated transport and stock costs are expected costs.

```
Parameter TotalProductionCost {
    Unit        : $;
    Definition  : sum[ (f,t), UnitProductionCost(f) * Production(f,t) ];
}

Parameter TotalTransportCost {
    Unit        : $;
    Definition  : sum[ (f,c,t,s), ScenarioProbability(s) * UnitTransportCost(f,c)
                                  * Transport(f,c,t,s) ];
}

Parameter TotalStockCost {
    Unit        : $;
    Definition  : sum[ (l,t,s), ScenarioProbability(s) * UnitStockCost(l) * Stock(l,t,s) ];
}

Parameter TotalModelSwitchCost {
    Unit        : $;
    Definition  : sum[ (f,p,t), FixedCostDueToLeaveChange
                                * ProductionLineLevelChange(f,p,t) ];
}
```

Creating a bar chart

Following the declaration of the above four identifiers, you can now create a bar chart object with as its first identifier `TotalProductionCost`. You can then open the **Bar Chart Properties** dialog box and use the **Contents** tab to add the remaining three identifiers (see Figure 11.13). You can ignore all the initialization warnings.

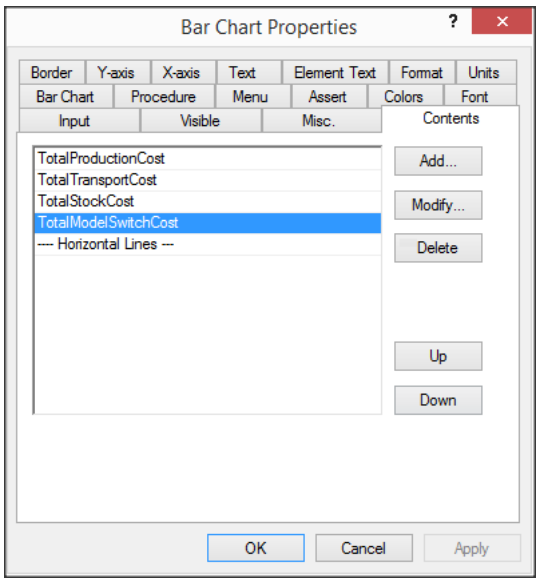


Figure 11.13: The **Contents** tab of the **Bar Chart Properties** dialog box

Viewing the result

The completed total costs bar chart should look like the one shown in Figure 11.14.

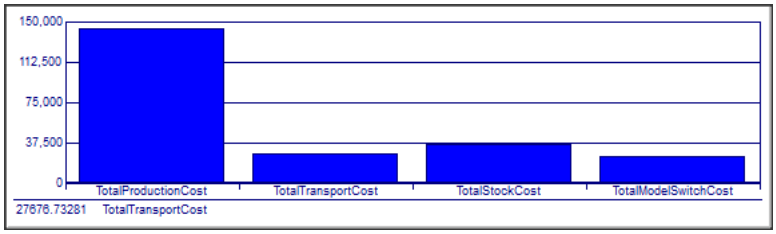


Figure 11.14: The completed total cost bar chart

11.2.8 Completing the page

Changing fonts

One way to display more information within objects on a page is to reduce the size of the font used. To create a new, small, font for use with all data objects you should execute the following actions:

- make sure that the *Production Overview* page is in **Edit** mode,
- select a table, and then
- select the remaining seven tables and bar charts while keeping the *Shift* key pressed,
- press the **Properties** button on the toolbar,

- select the **Font** tab, and
- press the **Add** button,
- enter '7' as the 'Font Size' (see Figure 11.15),
- press the **OK** button,
- specify 'Data Font' as the name of the new font, and
- press the **OK** button twice.

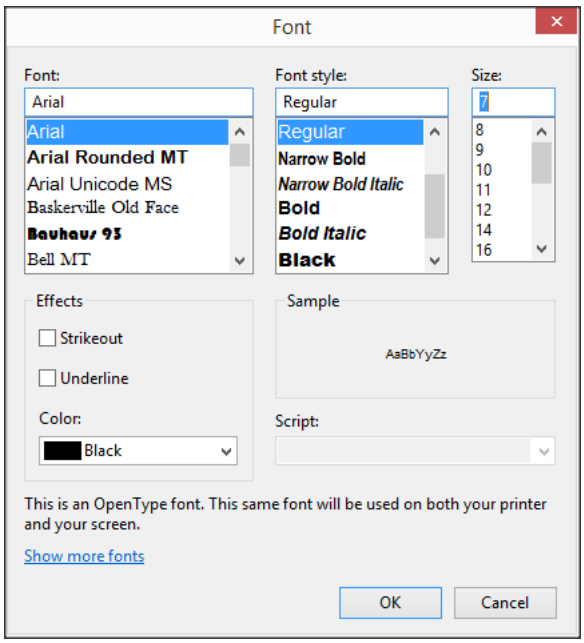


Figure 11.15: The specification of a new font

Alignment of objects

Several tables, bar charts and composite tables have been placed on the *Production Overview* page. To complete the page you should first align and resize the page objects in order to create a structured and attractive composition. For this purpose AIMMS offers several alignment tools that are accessible through the **Alignment** submenu of the **Edit** menu. The following alignment options are supported:

- aligning objects to the *left*, *right*, *top* or *bottom*,
- centering objects *horizontally* or *vertically*,
- spreading objects *horizontally* or *vertically*, and
- making object size equal in *width* or *height*.

Aligning the `Production Overview` *page*

You should now use the alignment tools described in the previous paragraph to align all the page objects as shown in Figure 11.16. Remember, if you need to select several objects at once, you should keep the *Shift* key pressed.

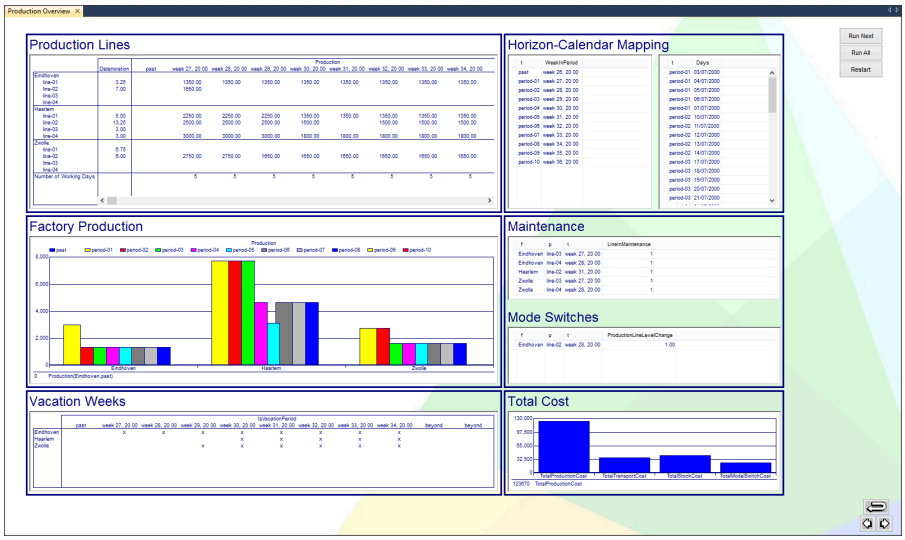


Figure 11.16: Aligned objects on the *Production Overview* page

Creating the text objects

Adding text to objects will help the end-user of your application. In this paragraph you will create a text object, and in the next paragraph you will change the font associated with this text. Consider first the production line table in the upper left corner, and add a line of text by following these steps:

- make sure that the *Production Overview* page is in **Edit** mode,
- select the **Text** command from the **Object** menu,
- draw a rectangle above the production line table,
- enter 'Production Lines' (without quotes) in the edit field (see also Figure 11.17), and
- press the **OK** button.

You should now create six more text objects as shown in Figure 11.2 at the beginning of this chapter.

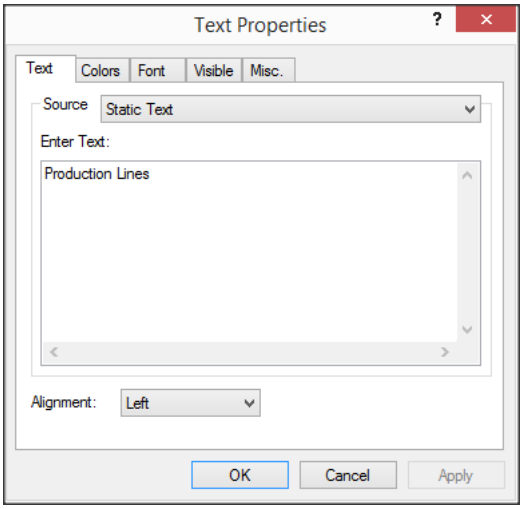


Figure 11.17: The **Text** tab of the **Text Properties** dialog box

Changing the text font

To change the font size of the text objects referred to in the previous paragraph, first select all of them using the *Shift* key, and create a new font named 'Title Font' with 'Font Size' 18. Again, you are referred to the text objects as shown in Figure 11.2.

Creating the rectangles

To improve the structure of your page even further, you can enclose one or more page objects within a rectangle. The following steps are required:

- make sure that the *Production Overview* page is in **Edit** mode,
- select the **Rectangle** command from the **Object** menu, and
- draw the rectangle around an object on your page.

Again, you should try to match the six rectangles as shown in Figure 11.2.

Rectangle line size

To embolden your rectangles you can enlarge their line thickness by executing the following actions:

- make sure that the *Production Overview* page is in **Edit** mode,
- select all rectangles using the *Shift* key,
- press the **Properties** button on the toolbar,
- complete the **Rectangle** tab of the **Rectangle Properties** dialog box as shown in Figure 11.18, and
- press the **OK** button.

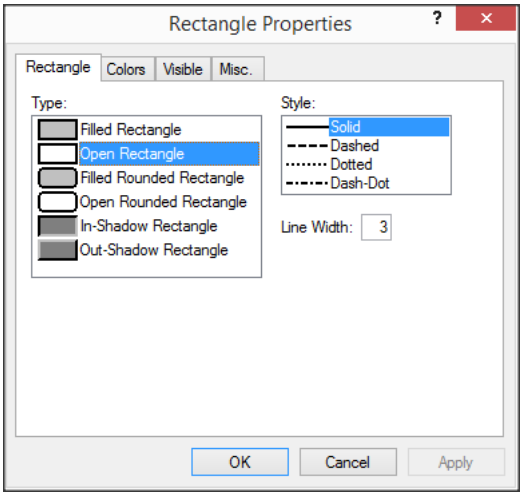


Figure 11.18: The **Rectangle** tab of the **Rectangle Properties** dialog box

Changing the foreground color

To change the default foreground color of all objects on the page from black to navy blue, you need to execute the following steps:

- make sure that the *Production Overview* page is in **Edit** mode,
- press the *Ctrl-A* key combination to select all objects on the page,
- unselect the three execution button using the *Shift* key,
- press the **Properties** button on the toolbar,
- select the **Colors** tab,
- select 'User' as the determinant of the 'Foreground' color,
- select the color 'Navy Blue' from the drop-down list, and
- press the **OK** button.

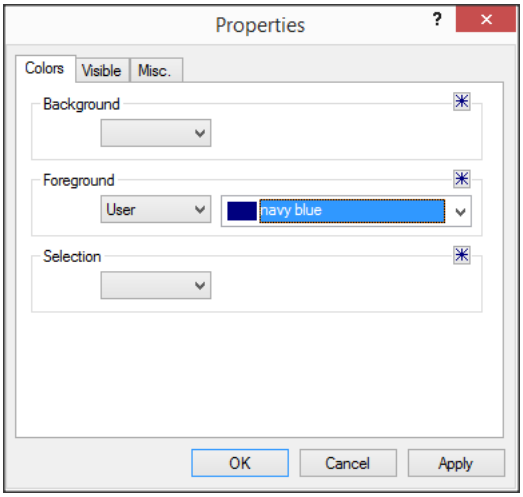


Figure 11.19: The **Colors** tab of the **Properties** dialog box

Changing text inside objects

By default, AIMMS will display the identifier names inside data objects. If this default name needs to be changed for your end-user, you can enter your own preferred string. You can even enter a string parameter, so that you can serve end-users with different language needs. As an illustration, please change the default representation of the identifier `ActualNumberOfDaysInPeriod` to the string 'Number of working days' by performing the following steps:

- select the production lines table,
- open its **Table Properties** dialog box,
- select the **Text** tab,
- select the identifier `ActualNumberOfDaysInPeriod(t)`,
- select 'Other' from the drop-down list in the 'Title' section,
- specify "Number of working days" (in quotes) as the new title (see Figure 11.20), and
- press the **OK** button.

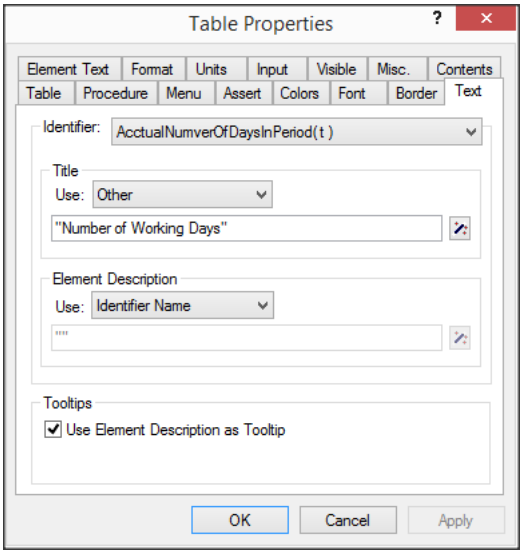


Figure 11.20: The **Text** tab of the **Table Properties** dialog box

Coloring data entries

In AIMMS it is even possible to color the individual data entries in tables. For instance, you might want to display the deterioration levels in red instead of blue whenever these levels have reached their maximum. To do this, you should first create a so-called *color parameter*. Such a parameter is an element parameter in the predefined AIMMS set `AllColors`. The contents of this set can be inspected or changed using the **User Colors** command from the **Tools** menu.

Creating a color parameter ...

As an example, please declare the following color parameter in the `Production Overview Declarations` section:

```
ElementParameter DeteriorationColor {
    IndexDomain : (f,p) | p in FactoryProductionLines(f);
    Range       : AllColors;
    Definition  : {
        if (DeteriorationLevel(f,p) > MaximumDeteriorationLevel(f,p) ) then
            'red'
        else
            'navy blue'
        endif
    }
}
```

... and linking it to model data

To specify the actual link between the color parameter and the data in the table you should perform the following actions:

- open the **Table Properties** dialog box of the production lines table,
- select the **Colors** tab,
- select the identifier `DeteriorationLevel(f,p)` in the 'Identifier' section (at the bottom),
- select 'Model' as the color determiner,
- press the **Wizard** button (see Figure 11.21) to select the identifier `DeteriorationColor(f,p)`, and
- press the **OK** button.

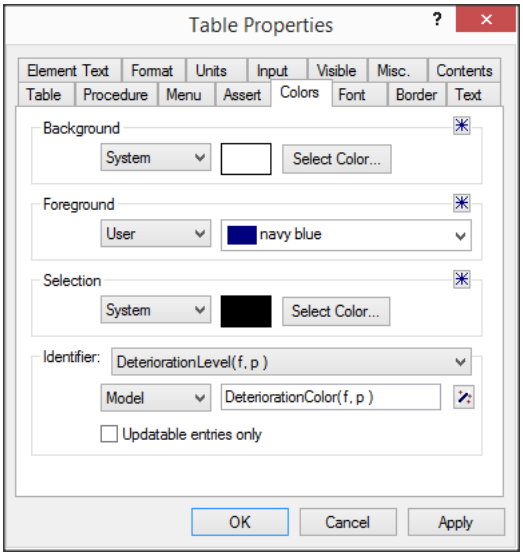


Figure 11.21: The **Colors** tab of the **Table properties** dialog box

The completed page

The completed *Production Overview* page is repeated in Figure 11.22, so that you can compare it with the contents of your screen.

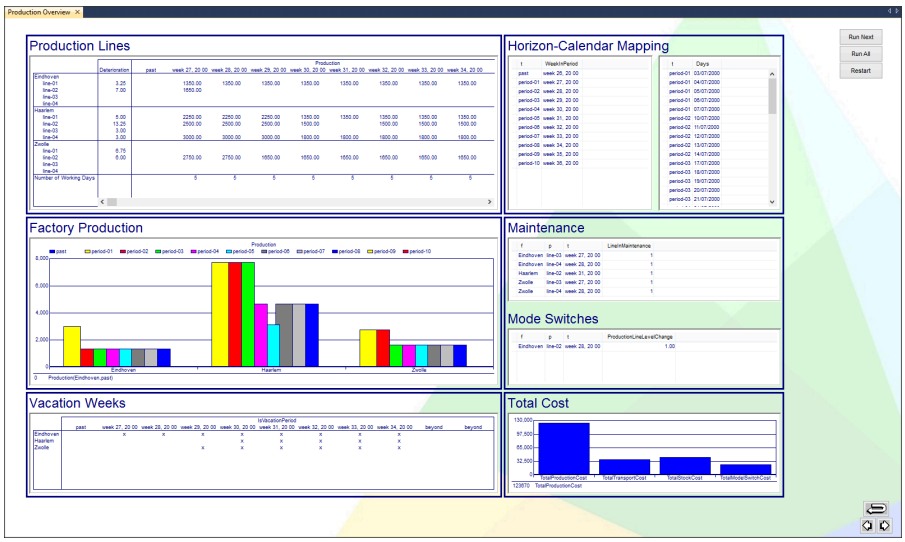


Figure 11.22: The completed *Production Overview* page

11.3 The Transport Overview page

Viewing the entire page

In this section you will construct the entire *Transport Overview* page as shown in Figure 11.23. Each page object is covered by a separate subsection.

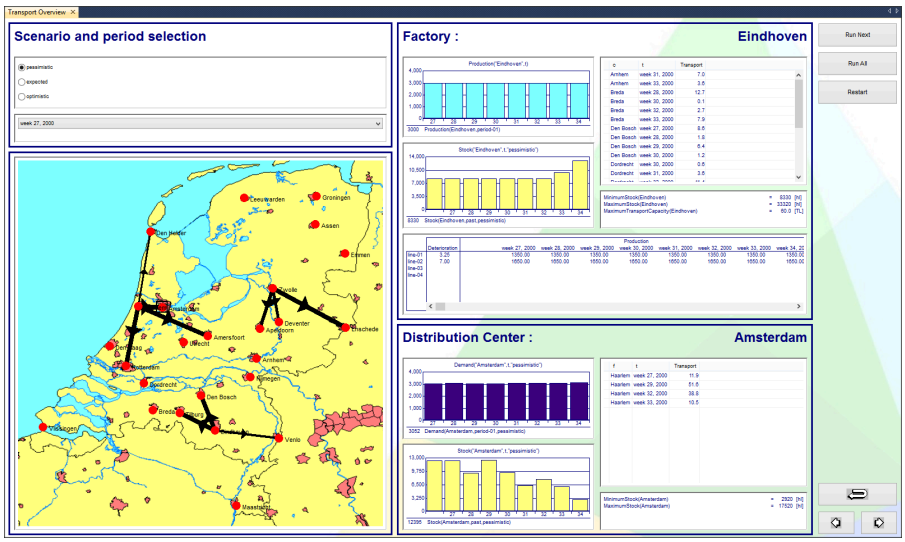


Figure 11.23: The completed *Transport Overview* page

11.3.1 Scenario selection object

Scenario dependency

The values of the identifiers Transport and Stock are different for each demand scenario. Displaying these values for all scenarios on a single page would overload the page. Therefore, the displayed information will be limited to one scenario, and the end-user will be able to switch between scenarios. AIMMS provides a *selection object* for this purpose.

Creating a scenario parameter

In the model section Transport Overview you should first create a new declaration section Transport Overview Declarations containing the following element parameter:

```
ElementParameter DisplayScenario {
    Range       : Scenarios;
}
```

Creating a selection object

The value of this element parameter is then determined by linking it to a selection object through the following steps:

- open the *Transport Overview* page in **Edit** mode,
- press the **New Selection Object** button on the toolbar,
- drag and create a small rectangle in the upper left corner,
- select 'Radio Buttons' from the 'Single Item Selection' options,

- select 'Element Parameter' as the 'Type of Data',
- press the **Wizard** button next to the 'Element' field (see Figure 11.24),
- select the element parameter `DisplayedScenario`,
- press the **Finish** button, and
- press the **OK** button.

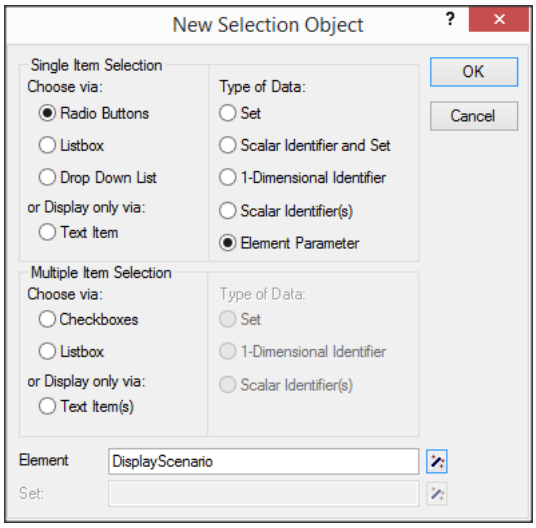


Figure 11.24: The **New Selection Object** dialog box

Using the selection object

The selection object that you have created is shown in Figure 11.25. Selecting a radio button in the selection object will set the corresponding value of the element parameter `DisplayedScenario`. As you will see later in this section, other page objects will be defined over this element parameter, and their data will adjust accordingly.

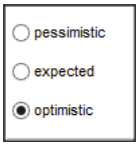


Figure 11.25: The scenario selection object

11.3.2 Period selection object

Creating a period parameter

As with the element parameter `DisplayedScenario`, you can introduce another element parameter to support the selection of a particular period. Please

declare the following element parameter at the end of the section `Transport Overview`.

```
ElementParameter DisplayedPeriod {
    Range       : Periods;
}
```

Creating the drop down list

When creating the selection object that sets the element parameter `DisplayedPeriod`, you should select the 'Drop Down List' option rather than the 'Radio Buttons' option (see Figure 11.26).

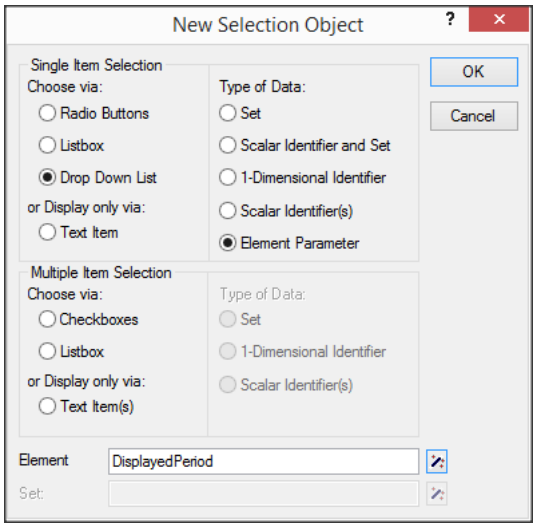


Figure 11.26: The **New Selection Object** dialog box

Specifying element text

Once you have created the drop down list, you can open its **Selection Object Properties** dialog box (either by double-clicking or using the right-mouse pop-up menu), and change the element text from abstract period references to specific week references. You can accomplish this change by selecting the **Element Text** tab, and specifying the string parameter `PeriodDescription(t)` as the element text of the index `Periods`.

Initializing element parameters

To initialize the two element parameters `DisplayedScenario` and `DisplayedPeriod` you should temporarily change the page mode to **User** mode, and use the two selection objects to select 'optimistic' as the displayed scenario and 'week 27, 2000' as the displayed period.

11.3.3 Transport network object

Copying the network object

The third object to be created on the transport page is a network object displaying the optimal transports for a given scenario and a given period in the planning interval. In Chapter 5 you created a network object displaying all locations and this will be used to create the new network object. To copy the existing network from the *Locations* page to the *Transport Overview* page you should perform the following steps:

- open both the *Locations* and the *Transport Overview* pages in **Edit** mode,
- select the *Locations* page tab,
- select the network object on the *Locations* page,
- press the **Copy** button on the toolbar,
- close the page by clicking on the cross in the upper right corner,
- select the *Transport Overview* page tab,
- press the **Paste** button,
- position the network object underneath the selection object, and
- press the left-mouse button.

Adding arcs to the network

The network object that you created in Chapter 5 only showed the locations. You can now add arcs to the network object to represent the optimal transport between the factories and the distribution centers for a given period and a given scenario. To add these arcs, you should take the following actions:

- select the network object in **Edit** mode,
- open its **Network Object Properties** dialog box,
- select the **Contents** tab,
- select the '----- Arcs -----' entry from the listbox,
- press the **Add** button,
- select the variable `Transport(f,c,t,s)`, and
- press the **Next** button.

Next you need to specify that the indices `t` and `s` will assume the values of the element parameters `DisplayedPeriod` and `DisplayedScenario` respectively:

- select the index `t` from the list box,
- select the 'Element Parameter' radio button,
- select the element parameter `DisplayedPeriod` from the drop-down list,
- select the index `s` from the list box,
- select the 'Element Parameter' radio button,
- select the element parameter `DisplayedScenario` from the drop-down list,
- press the **Finish** button (see Figure 11.27), and
- press the **OK** button.

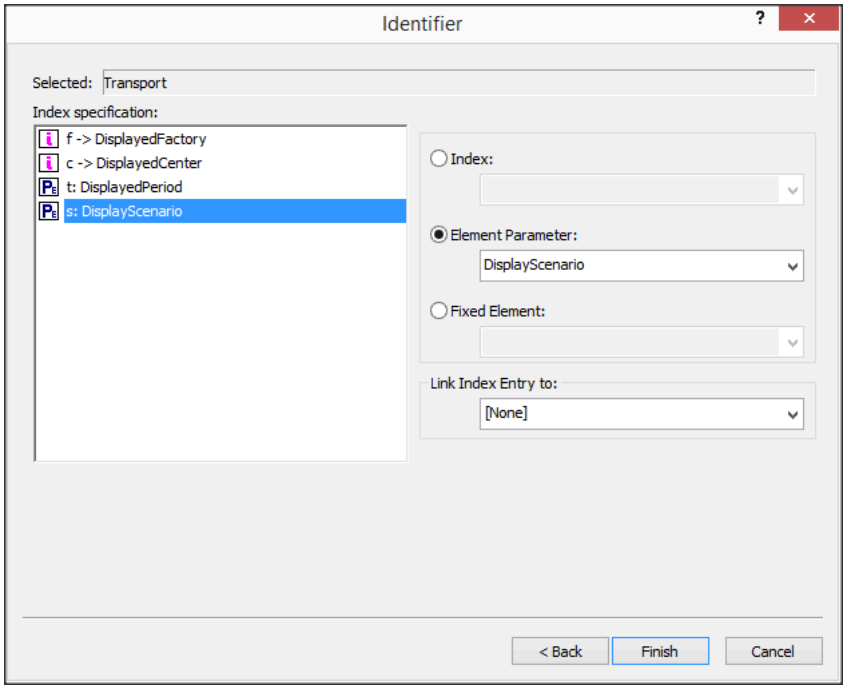


Figure 11.27: Fixing indices of the variable `Transport`

Specifying arc thickness

The network object will display arcs for all transport values that have a non-zero value. To distinguish between small and large transport values the thickness of the arc can be varied depending on the transport value. To achieve this you should execute the following actions:

- select the network object in **Edit** mode,
- open its **Network Object Properties** dialog box,
- select the **Arcs** tab,
- press the **Wizard** button to the right of the 'Size' field,
- select the identifier `Transport(f,c,t,s)`,
- press the **Next** button,
- link the index `t` to the element parameter `DisplayedPeriod`,
- link the index `s` to the element parameter `DisplayedScenario`,
- press the **Finish** button, and
- press the **OK** button.

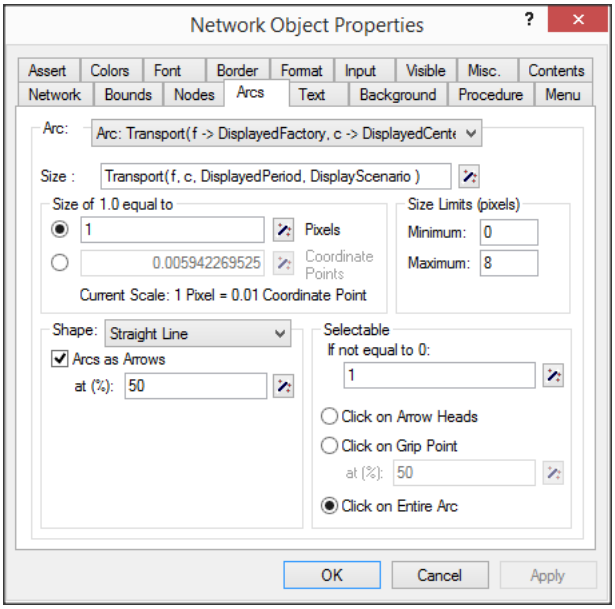


Figure 11.28: The **Arcs** tab of the **Network Properties** dialog box

Viewing the arcs

Assuming that you have already solved the model for the first step, the arcs in the network object should now have different widths as shown in Figure 11.29.

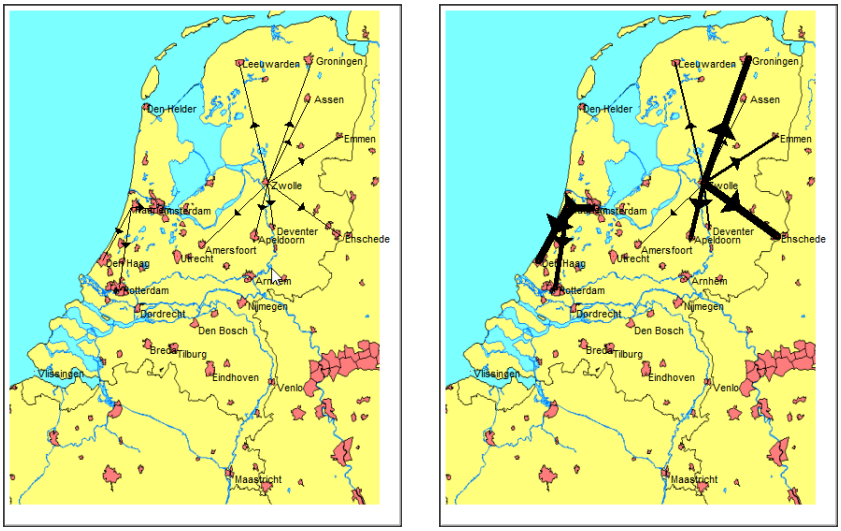


Figure 11.29: Using arc thickness to illustrate transport volumes

Node and arc dependent information

AIMMS has facilities to display node and arc dependent information whenever an end-user selects a node or an arc in the network object. Consider, for instance, Figure 11.23. The data block displayed in the lower right corner of that page deals with a particular distribution center, while the data block displayed in the upper right corner deals with a particular factory. In the following paragraphs you will specify how the selection of a particular arc will update both of these data blocks, while the selection of a particular node will update one of these data blocks.

Declaring location identifiers

The following two element parameters will be needed to hold the current choice of factory and distribution center. Please add their declarations to the Transport Overview Declarations.

```
ElementParameter DisplayedFactory {
    Range       : Factories;
}

ElementParameter DisplayedCenter {
    Range       : Centers;
}
```

Specifying arc dependency

Arc dependency can then be specified with the aid of the above two element parameters. Whenever an arc is selected, the locations of the corresponding two end nodes should become the current values of DisplayedFactory and DisplayedCenter. As soon as their values change, the data blocks in Figure 11.23 will be updated accordingly. To implement this action, you should execute the following steps:

- select the network object in **Edit** mode,
- open its **Network Object Properties** dialog box,
- select the **Contents** tab,
- select the arc Transport(f,c,DisplayedPeriod,DisplayedScenario),
- press the **Modify** button,
- press the **Next** button,
- select the index f from the 'Index specification' list box,
- use the drop-down list under 'Link Index Entry' to select the element parameter DisplayedFactory,
- repeat the previous two steps to link the index c the element parameter DisplayedCenter,
- press the **Finish** button, and
- press the **OK** button.

By simply linking an index to an element parameter as shown in Figure 11.30 you have specified the linkage between a selection and a data block. This powerful facility is also available for other data objects in AIMMS.

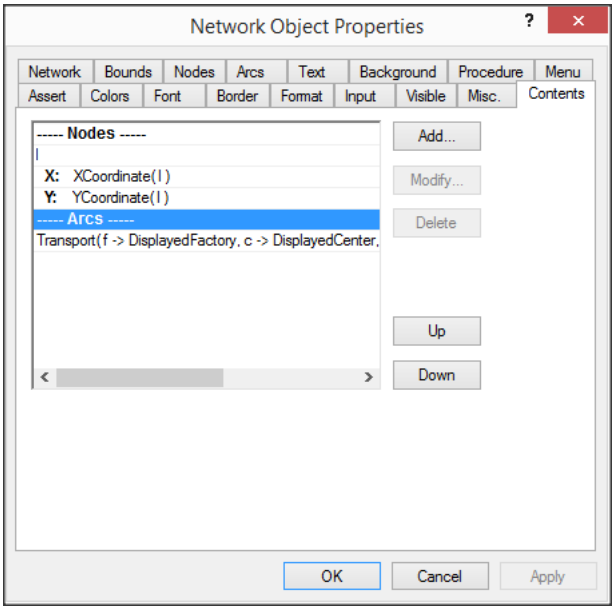


Figure 11.30: The **Contents** tab of the **Network Object Properties** dialog box

Specifying node dependency

Specifying node dependency is not as straightforward as with arc dependency, because a node is a location that can be either a factory or a distribution center. This makes the linkage between a node and one of the data blocks less trivial to specify. A straightforward procedure, however, can resolve this choice. Once you have specified such a procedure, it is then straightforward to link it to the network object.

The selection procedure ...

Create a procedure `SelectLocationInNetwork(SelectedLocation)`, where the argument `SelectedLocation` is declared as a local element parameter with **Range** attribute `Locations` and with the **Property** attribute 'Input' as shown in Figure 11.31. The following conditional statement will constitute the **Body** attribute of this procedure:

```
if ( SelectedLocation in Factories )
    then  DisplayedFactory := SelectedLocation ;
    else  DisplayedCenter  := SelectedLocation ;
endif;
```

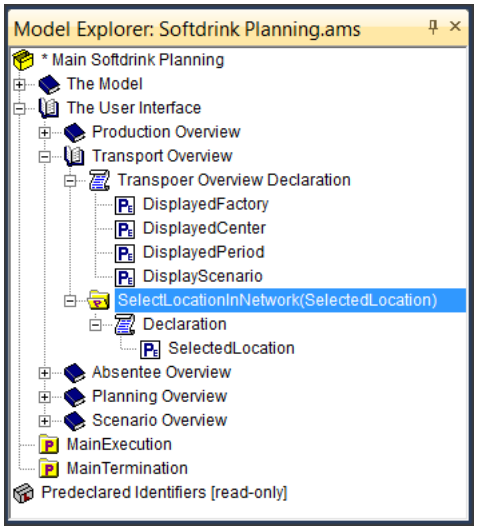


Figure 11.31: The contents of the `Transport Overview` section

... linked to the network object

The above procedure will be linked to the network object as a *procedure upon selection* by executing the following steps:

- select the network object in **Edit** mode,
- open its **Network Properties** dialog box,
- select the **Procedure** tab,
- verify that 'Node: l' is selected as the 'Identifier',
- select the procedure `SelectLocationInNetwork` as the 'Upon Selection' procedure,
- press the **Next** button,
- select the 'Index' radio button,
- select the index `l` from the 'Index' drop-down list (see Figure 11.32),
- press the **Finish** key, and
- press the **OK** button.

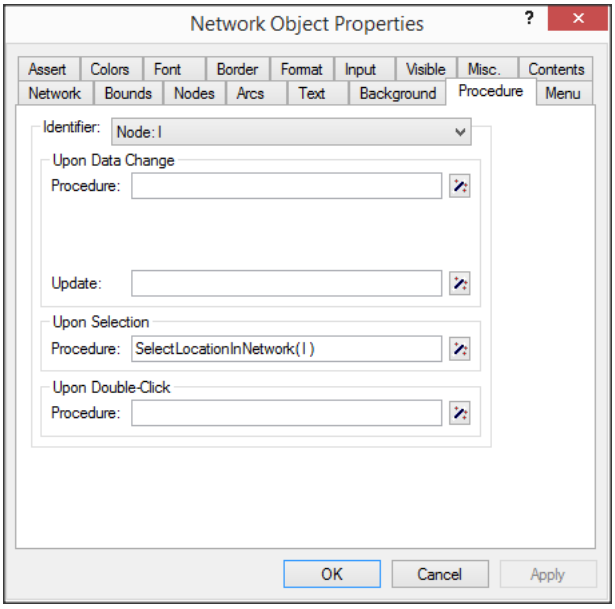


Figure 11.32: The **Procedure** tab of **Network Object Properties** dialog box

Increasing the node size

In order to see nodes in the network more clearly, you can increase their size by changing the **Nodes** tab of the network object as shown in Figure 11.33. If you want, you can also change their color using the **Colors** tab.

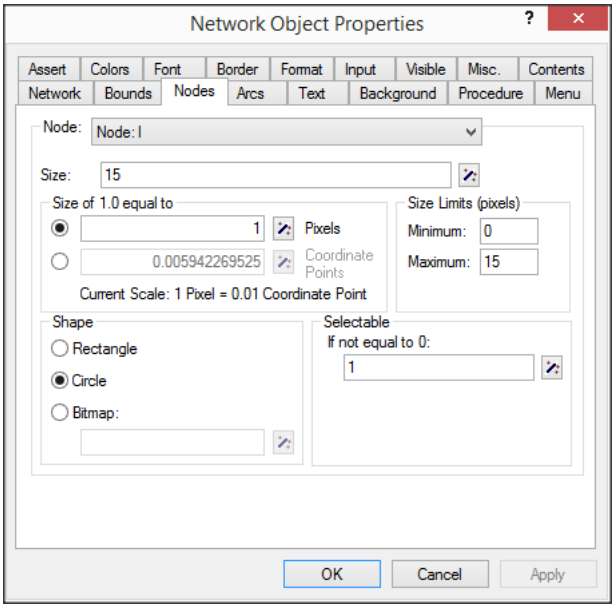


Figure 11.33: The **Nodes** tab of the **Network Object Properties** dialog box

Viewing the result

Once you have increased the node size, the network object should look like the one shown in Figure 11.34. If you had used separate node sets for factories and centers, different icons could have been used to represent them in the network object.

Figure 11.34: The network object with increased node size

11.3.4 Factory text object

Factory description

The upper right data block in the *Transport Overview* page contains data pertaining to a particular factory. The name of that factory is displayed at the top of this block using a text object. The following string parameter is needed to fill this object:

```
StringParameter FactoryDescription {
    Definition  : FormatString( "%e", DisplayedFactory );
}
```

You should add this declaration at the end of the `Transport Overview Declarations` section.

Creating a text object

You should now create a text object that will display the contents of the string parameter you have just declared. Try to create the text object on your own. To display the string parameter `FactoryDescription` you should complete the **Text** tab of the **Text Properties** dialog box as shown in Figure 11.35. You can also try changing its color and font size.

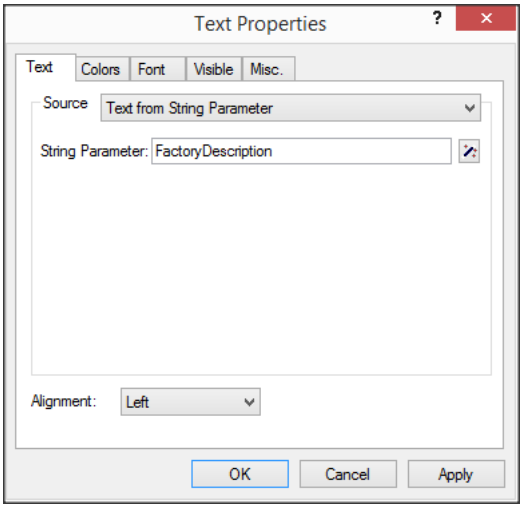


Figure 11.35: The **Text** tab of the **Text Properties** dialog box

11.3.5 The factory production bar chart

Creating the bar chart

You will begin by creating a bar chart containing the production data corresponding to the currently selected factory. The name of this factory is the value of the element parameter `DisplayedFactory`. You should execute the following steps:

- make sure the *Transport Overview* page is opened in **Edit** mode,
- press the **New Bar Chart** button on the toolbar,
- drag a rectangle underneath the factory description text object,
- select the variable `Production(f,t)` in the **Identifier** wizard,
- press the **Next** button,
- link the index `f` to the element parameter `DisplayedFactory`, and
- press the **Finish** button.

Adjusting the element text

The period references along the x-axis are probably too long to fit. The `PeriodDescription` parameter contains even longer strings. To create short references you should now create the following string parameter:

```
StringParameter ShortPeriodDescription {
    IndexDomain : t;
    Definition  : {
        if (WeekInPeriod(t) )
            then FormatString( "%n" , TimeslotCharacteristic( WeekInPeriod(t), 'week' ) )
            else ""
        endif
    }
}
```

You should change the element description of the period index `t` to be the string parameter `ShortPeriodDescription` using the **Element text** tab of the **Bar Chart Properties** dialog box.

The page so far

At this point, the page on your screen should resemble the partially completed *Transport Overview* page shown in Figure 11.36.

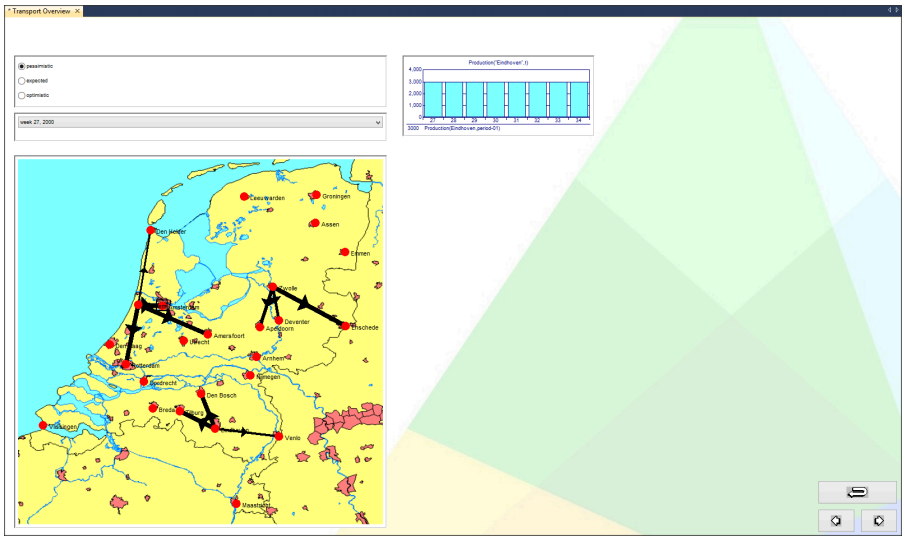


Figure 11.36: The current *Transport Overview* page

11.3.6 The factory stock bar chart

Copying the previous bar chart

To create a bar chart containing the stock values for the currently selected factory, you can make use of the following copy, paste and adjust actions:

- select the production bar chart you have just created,
- press the **Copy** button on the toolbar,
- press the **Paste** button on the toolbar,

- position and drop the new bar chart underneath the production bar chart,
- press the **Properties** button on the toolbar,
- select the **Contents** tab,
- select the identifier `Production(DisplayedFactory,t)` from the listbox,
- press the **Modify** button,
- select the identifier `Stock(l,t,s)`,
- press the **Next** button,
- link the index `l` to the element parameter `DisplayedFactory`,
- link the index `s` to the element parameter `DisplayedScenario`,
- press the **Finish** button, and
- press the **OK** button.

11.3.7 Factory transport composite table

Specifying the table ...

The network object only displays transport values for the selected period. To view the transport values for all periods in the planning interval you can create a composite table by executing the following steps:

- press the **New Composite Table** button on the toolbar,
- draw a rectangle on the page,
- select the variable `Transport(f,c,t,s)`,
- press the **Next** button,
- link the index `f` to the element parameter `DisplayedFactory`,
- link the index `s` to the element parameter `DisplayedScenario`,
- press the **Finish** button,

... and improving its appearance

You can improve the overall appearance of the table by taking the following actions:

- specify the string parameter `PeriodDescription(t)` as the element text of the index `t`, and
- change the font to the 'Data Font' that you specified in Subsection 11.2.8.

The resulting table should now look like the one shown in Figure 11.37.

c	t	Transport
Arnhem	week 20, 20 01	2.2
Arnhem	week 21, 20 01	8.1
Arnhem	week 23, 20 01	4.1
Breda	week 20, 20 01	9.7
Breda	week 22, 20 01	5.1
Breda	week 23, 20 01	5.0
Breda	week 24, 20 01	5.4
Breda	week 25, 20 01	5.0
Breda	week 26, 20 01	5.1

Figure 11.37: The factory transport composite table

11.3.8 Factory properties scalar object

To be able to view the minimum and maximum stock levels as well as the maximum transport capacity for the selected factory, you should first create a scalar object with the first of these identifiers:

- create a scalar object,
- select the identifier `MinimumStock(l)`, and
- link its index `l` to the element parameter `DisplayedFactory`.

Next, you should add the remaining two identifiers to the scalar object by performing the following actions:

- open the **Properties** dialog box,
- select the **Contents** tab,
- press the **Add** button,
- select the identifier `MaximumStock(l)`,
- press the **Next** button,
- link the index `l` to the element parameter `DisplayedFactory`,
- press the **Finish** button,
- press the **Add** button,
- select the identifier `MaximumTransportCapacity(f)`,
- press the **Next** button,
- link the index `f` to the element parameter `DisplayedFactory`,
- press the **Finish** button, and
- press the **Apply** button.

Identifier `MinimumStock(l)` and `MaximumStock(l)` have different unit from `MaximumTransportCapacity(f)`. The unit of each identifier will be shown by the following steps:

- select the **Units** tab of the **Properties** dialog box,
- it shows the setting of the first identifier `MinimumStock(DisplayedCenter)`,
- select the **Show per Value** radio button under **Display** as Figure 11.38,
- click the drop down list on top of the dialog,
- select the second identifier `MaximumStock(DisplayedCenter)`,
- again, select the **Show per Value** radio button under **Display**,
- repeat this for `MaximumTransportCapacity(DisplayedCenter)` as well, and
- press the **OK** button.

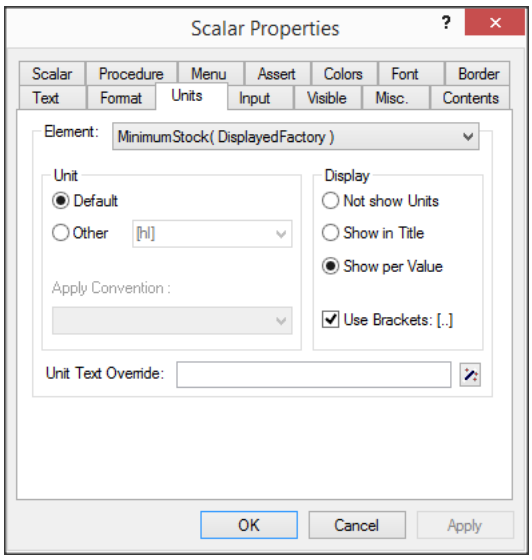


Figure 11.38: The **Property** dialog of factory scalar object

The resulting table should look like the one shown in Figure 11.39 including the appropriate values.

Figure 11.39: The factory scalar object containing factory limitations

11.3.9 Factory production line table

Copying the production line table

The factory production line table is essentially the same as the production line table on the *Production Overview* page with the exception that the index `f` is replaced by the element parameter `DisplayedFactory`. The following steps involve copying the table from one page to the next:

- open both the *Production Overview* and the *Transport Overview* page in **Edit** mode,
- select the *Production Overview* page tab,
- select the production line table,
- press the **Copy** button on the toolbar,
- close the page,
- select the *Transport Overview* page tab,
- press the **Paste** button ,

- position the object underneath the other factory information objects, and
- press the left-mouse button.

Changing table properties

The following changes are required to display only the information for the currently selected factory:

- open the **Properties** dialog box of the new table,
- select the **Contents** tab,
- select the `DeteriorationLevel(f,p)` entry in the list,
- press the **Modify** button,
- press the **Next** button,
- link the index `f` to the element parameter `DisplayedFactory` and close the wizard,
- select the `ActualProduction(f,p,t)` entry in the list,
- press the **Modify** button,
- press the **Next** button,
- link the index `f` to the element parameter `DisplayedFactory` and close the wizard,
- select the `ActualNumberOfDaysInPeriod(t)` entry from the list,
- press the **Delete** button, and
- press the **Apply** button.

An error dialog will appear due to the fact that on the **Colors** tab there is still reference to the index f. By pressing the **Ok** on the dialog window, AIMMS will get rid of the index reference (i.e. removing the `DeteriorationColor(f,p)`). Therefore, you have to specify color for the `DeteriorationLevel(f,p)` again and change the index reference. This can done by executing the following steps:

- select the **Colors** tab,
- in the 'Identifier' section select 'Model' as the color determiner,
- press the **Wizard** button again to select the identifier `Deterioration-Color(f,p)`
- link the index `f` to the element parameter `DisplayedFactory`,
- press the **Finish** button, and
- press the **OK** button.

The resulting table is shown in Figure 11.40.

	Deterioration	Production past	week 27, 20 00	week 28, 20 00	week 29, 20 00	week 30, 20 00	week 31, 20 00	week 32, 20 00	week 33, 20 00	wee
line-01	2.25		1080	1350		1350	1350	1350		
line-02	6.00		1320	1650	1320	1650	1650	1650		
line-03										
line-04										

Figure 11.40: The factory production line table

Arranging the factory objects

At this stage you should use the aligning and resizing facilities that were discussed in Subsection 11.2.8 to rearrange the composition objects as shown in Figure11.41. Once the factory data block is neatly organized, you can copy it in its entirety to create a similar data block for distribution centers.

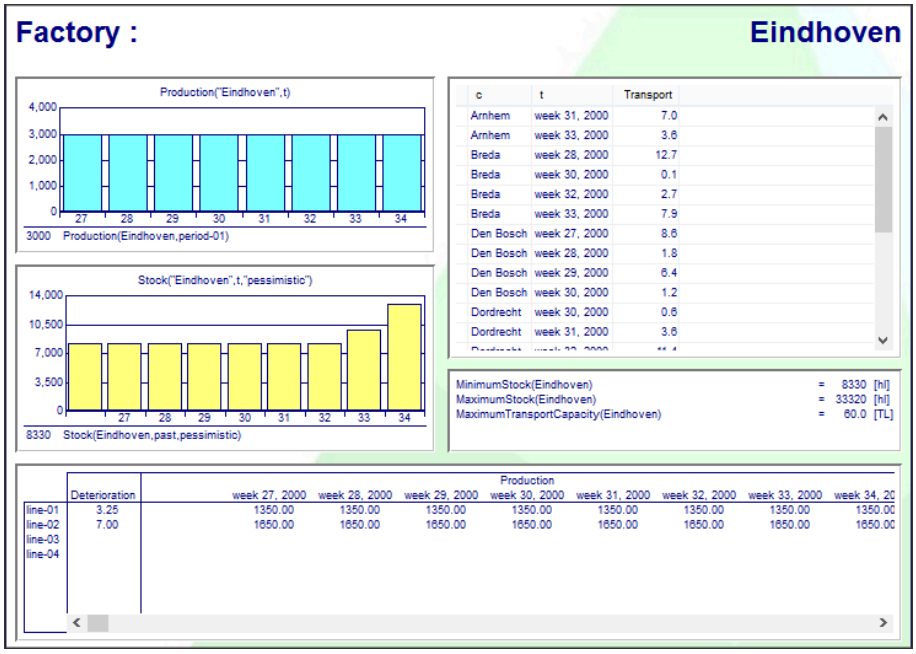


Figure 11.41: The factory data block

11.3.10 The distribution center data block

To create the four page objects for a particular distribution center you should execute the following steps:

- select all objects in the factory data block except for the production lines table at the bottom using the *Shift* key,
- press the **Copy** button on the toolbar,
- press the **Paste** button on the toolbar,
- position the five objects underneath the factory information area (see Figure 11.42), and
- press the left-mouse button.

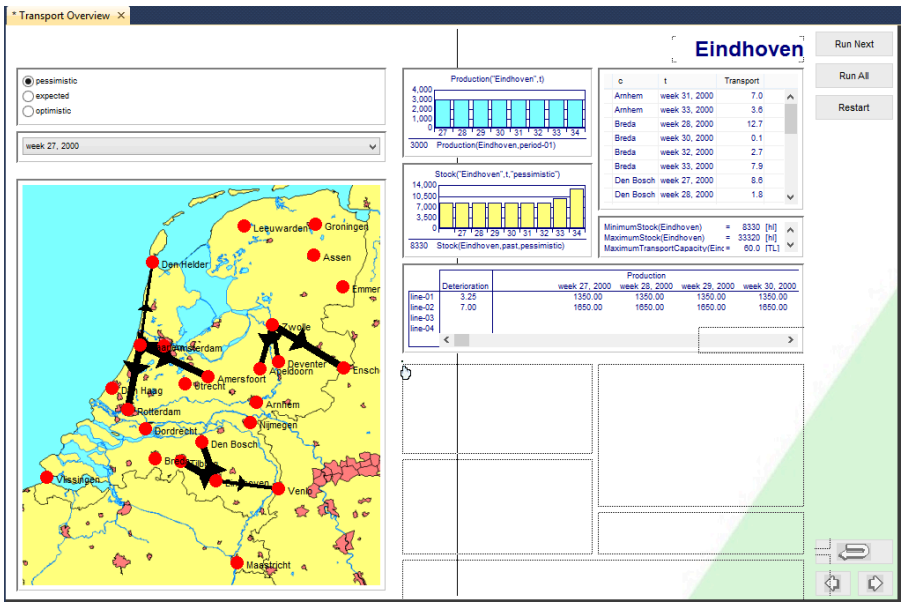


Figure 11.42: The copy and paste process illustrated

Making the required modifications

By now you should have enough experience to make a series of modifications to transform the factory data block into a distribution center data block. First add the following declaration at the end of the `Transport Overview Declarations` section.

```
StringParameter CenterDescription {
    Definition  : FormatString( "%e" , DisplayedCenter );
}
```

The following list of actions now needs to be executed, using the detailed knowledge gained so far:

- change the string parameter `FactoryDescription` to the string parameter `CenterDescription` using the **Text** tab of the **Text Properties** dialog box of the copy of the text object,
- remove `MaximumTransportCapacity(DisplayedFactory)` from the **Contents** tab of the scalar object,
- find `Production(DisplayedFactory,t)` on the **Contents** tab of the production bar chart,
- change this to `Demand(DisplayedCenter,t,DisplayedScenario)`,
- find `Transport(DisplayedFactory,c,DisplayedPeriod,DisplayedScenario)` on the **Contents** tab of the factory transport composite table,

- change this to `Transport(f,DisplayedCenter,DisplayedPeriod,DisplayedScenario)`,
- open, in sequence, the **Contents** tab of the **Properties** dialog box associated with the table, the scalar object and the two bar charts, and
- replace all references to the element parameter `DisplayedFactory` with one to the element parameter `DisplayedCenter`.

11.3.11 Completing the page

Copying the execution buttons

At this point you should copy the three execution buttons (**Run Next**, **Run All** and **Restart**) from the *Production Overview* page, and paste them at the same position on the *Transport Overview* page. You could introduce a new template page for this purpose.

Beautifying the page

Finally, you could enhance the page by adding rectangles, changing text color and sizes as discussed in Subsection 11.2.8. Figure 11.43 will serve as a guide while completing the *Transport Overview* page on your screen.

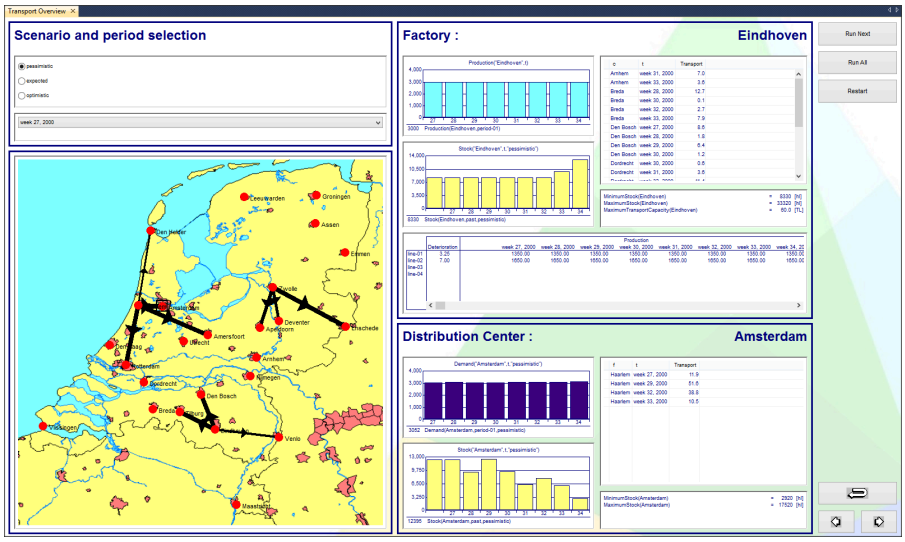


Figure 11.43: The completed *Transport Overview* page

Chapter 12

Absentee and Planning Overviews

This chapter

In this chapter you will construct two end-user pages including Gantt charts and composite tables for the display of model data. A Gantt chart is an advanced page object that is especially useful for displaying scheduling and planning data defined over time.

12.1 Gantt charts

Gantt chart description

A Gantt chart typically contains a number of interrelated *tasks/processes/jobs* viewed against a time scale. Such a chart consists of one or more rows in which horizontal bars are displayed. Each individual bar represents a single task, and the length of the bar gives a visual impression of when and for how long that specific task is to be performed. The rows typically refer to *resources* that are consumed by the individual tasks. It could be that your schedule involves several types of tasks (e.g. maintenance tasks and line usage tasks). In this case, the Gantt chart can be configured using colors and/or text inside bars to indicate what type of task is performed for each resource.

Controlling appearance

You can use several AIMMS identifiers to control the appearance of the Gantt chart. The extensive controls cannot be explained in a single paragraph. You can, however, exercise control over the time scales along the x-axis (see Figure 12.2), and over the position and color of each individual bar.

Three Gantt charts

In this chapter you will construct three Gantt charts. The first Gantt chart will be used to plan the vacation periods for each factory on a weekly basis. The second Gantt chart will be used to schedule official holidays on a daily basis. Using these two Gantt charts your end-user will be able to graphically schedule holidays and vacations by merely clicking on the bars inside these charts. The third Gantt chart is not designed for data input, but will be used to display the overall maintenance and line usage output of the model.

12.2 The Absentee Overview page

Viewing the entire page

In this section you will construct the entire page shown in Figure 12.1. The two Gantt charts and the composite tables will be treated in separate subsections.

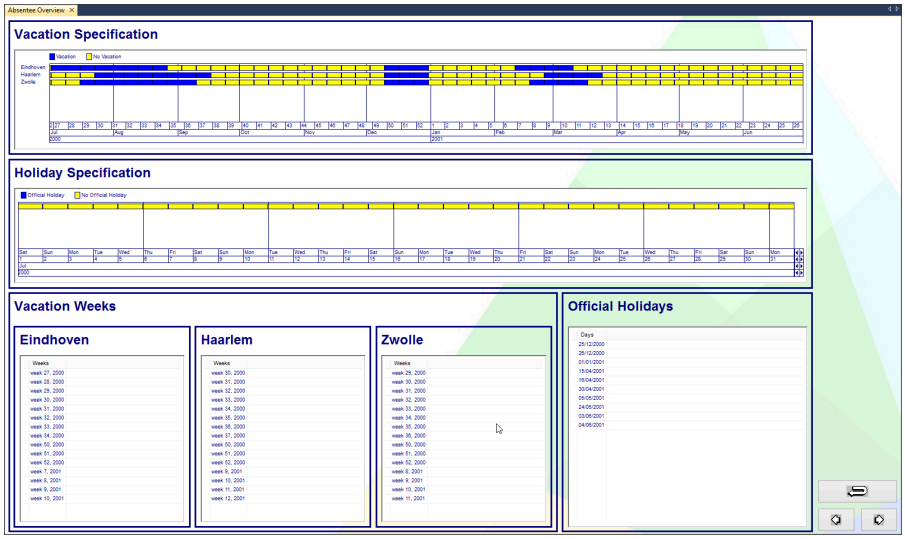


Figure 12.1: The completed *Absentee Overview* page

12.2.1 The vacation Gantt chart

Row and bar specification

The vacation Gantt chart will contain a single row for each factory. A factory can be viewed as a resource with workers. An amount of the resource is consumed when workers are on vacation. In this Gantt chart there will be two types of colored bars in each row. One bar is to denote that a particular week is scheduled as a 'Vacation', while the other bar denotes the opposite. Part of the Gantt chart you will develop is shown in Figure 12.2.

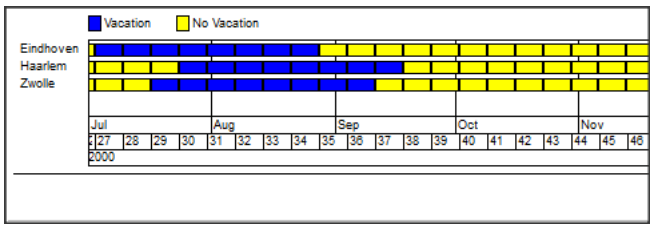


Figure 12.2: Part of the vacation planning Gantt chart

Required declarations

The Gantt chart will display all possible weeks along the x-axis. Every bar in this chart is specified by a *start*, indicating the specific week in which it starts, plus a *duration* to indicate the length of the bar. The vacation Gantt chart enables end-users to specify the vacation periods through mouse clicks. To build this facility you need to declare a few identifiers plus a simple procedure to toggle the bars between 'Vacation' and 'No Vacation'. Insert a new declaration section `Vacation Gantt Chart Declarations` in the `Absentee Overview` section of your model, and add the following declarations.

```
Set VacationGanntChartBarTypes {
    Index       : v;
    Definition  : data { 'Vacation', 'No Vacation' };
}

ElementParameter VacationGanttChartStartingWeek {
    IndexDomain : w;
    Range       : Weeks;
    Definition  : w;
}

Parameter VacationGanttChartDuration {
    IndexDomain : (f,w,v);
}
```

Toggling the bars

You can make AIMMS execute a particular procedure whenever an end-user selects a bar in the Gantt chart. In this example you want the procedure to toggle between 'Vacation' and 'No Vacation'. The following single statement achieves this task:

```
VacationGanttChartDuration(f,w,v) := 1 - VacationGanttChartDuration(f,w,v);
```

Whenever the corresponding procedure is executed, the value of the duration parameter switches between 0 and 1.

Declaring the toggling procedure

Create a new procedure called `ToggleVacationGanttChart(f,w)` as shown in Figure 12.3. Use the **Argument** wizard to declare `f` as an element parameter in the set `Factories` and with property 'Input'. Similarly, declare `w` as an element parameter in the calendar `Weeks` also with property 'Input'. Next, enter the statement from the previous paragraph in the **Body** attribute.

Using 'Duration' three ways

The duration parameter will be used in three different ways. First, as mentioned previously, it will be used to denote the length of a bar. The value 1 corresponds exactly to the length of the time interval along the x-axis, namely one week. In addition, this parameter will be used as a domain parameter of the Gantt chart, indicating which bars are to be drawn. Finally, the duration parameter will be used to establish the link between the Gantt chart and the set `VacationWeeks(t)` used in the mathematical program.

Gantt chart initialization

The procedure to initialize the Gantt chart is as short as the procedure to toggle the duration parameter. Only the following statement is needed in the **Body** attribute:

```
VacationGanttChartDuration(f,w,'No Vacation') :=
       1 - VacationGanttChartDuration(f,w,'Vacation')
```

With all values at their initial default of zero, this statement will initialize all weeks to 'No Vacation' weeks. Please add a procedure `InitializeVacationGanttChart` as shown in Figure 12.3, and insert the above statement into the **Body** attribute.

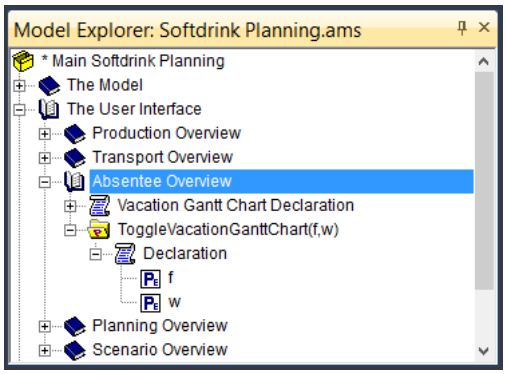


Figure 12.3: The contents of the `Absentee Overview` section

Adding to `MainInitialization`

At this point you should go back to the `MainInitialization` procedure, and add the statement `InitializeVacationGanttChart;` at the end of its **Body** attribute. You can quickly locate this procedure in your model tree by pressing the *Ctrl-F* key combination, or by pressing the **Find** button on the toolbar (see Figure 12.4).

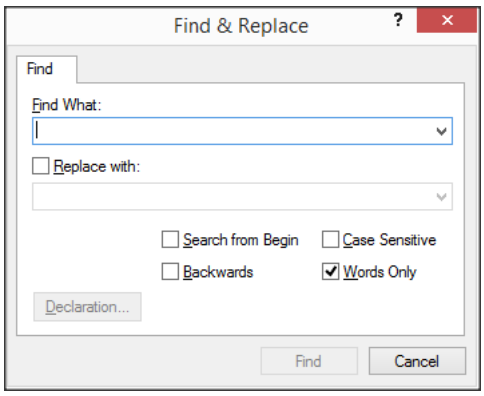


Figure 12.4: The **Find & Replace** dialog box

Executing the procedure

To prevent any initialization error when specifying the Gantt chart, you can now execute the `InitializeVacationGanttChart` procedure by selecting it in the model tree and issuing the **Run Procedure** command from the right-mouse pop-up menu.

Creating the Gantt chart object

You are now ready to create the vacation planning Gantt chart on a page by following the steps below:

- open the *Absentee Overview* page in **Edit** mode,
- press the **New Gantt Chart** button on the toolbar,
- drag a rectangle that matches the desired Gantt chart size on your page, and
- use the **Wizard** buttons to complete the **Gantt Chart** dialog box as shown in Figure 12.5.

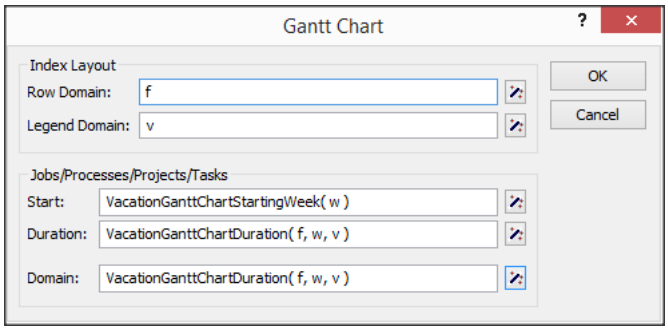


Figure 12.5: The **Gantt Chart** dialog box for vacation planning

Specifying the x-axis

The x-axis of the Gantt chart will initially display the descriptions of the elements in the calendar `Weeks`. AIMMS can change the labels along the x-axis by mapping the calendar element descriptions to the corresponding moments in time. In this tutorial, the element descriptions contain references to weeks, months and years. To change the time reference along the x-axis in the Gantt chart, you should execute the following steps:

- select the Gantt chart,
- open its **Properties** dialog box,
- select the **X-axis** tab,
- select 'Real-time Calendar' as the 'Type of X-axis',
- check 'Weeks', 'Months' and 'Years' as in Figure 12.6,
- select 'weeks' as the 'Unit of Measurement',
- enter "2000-06-26" (with the quotes) as the 'Reference Time',
- use the **Wizard** button to select the 'String Parameter' `BeginDateOfCalendar` as the 'Left Bound',
- use the **Wizard** button to select the 'String Parameter' `EndDateOfCalendar` as the 'Right Bound', and
- press the *Apply* button.

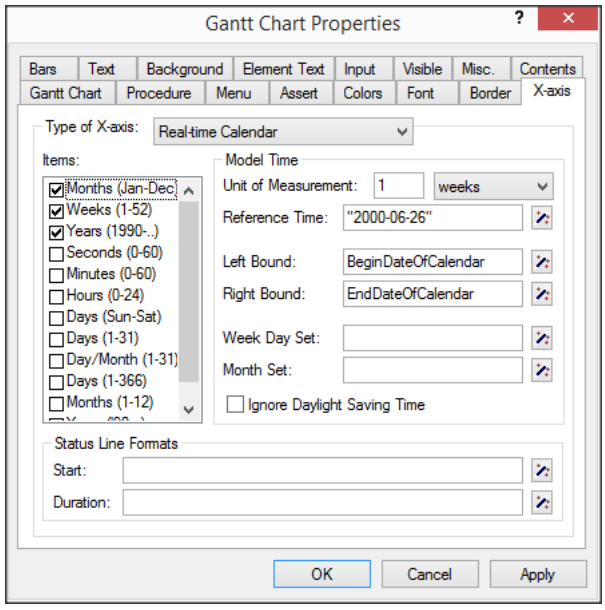


Figure 12.6: The **X-axis** tab of the **Gantt Chart Properties** dialog box

Implementing automatic toggling

To implement automatic toggling between the 'Vacation' and 'No Vacation' bar type, you should complete the **Procedure** tab as in Figure 12.7.

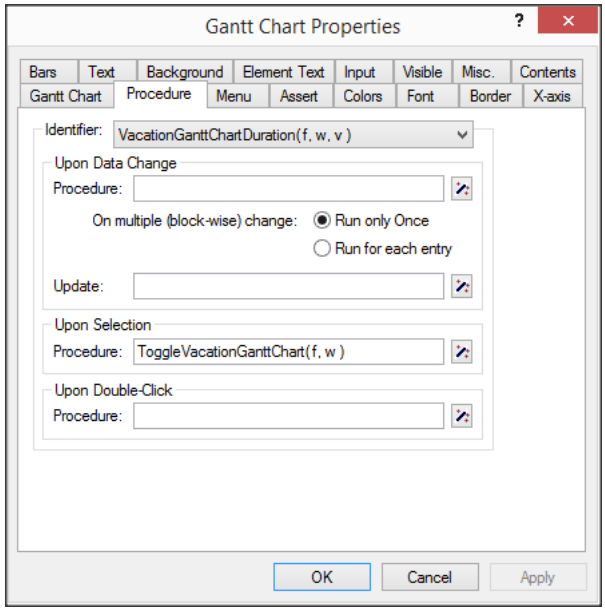


Figure 12.7: The **Procedure** tab of the **Gantt Chart Properties** dialog box

Changing the font size

Depending on the size of your Gantt chart, and the size of your screen, the default font used in the Gantt chart might be too large. You are advised to create a new 'Gantt Chart Font' with size 7 instead of the default 8 in the same manner as that shown in Section 10.3.

Testing the Gantt chart

The Gantt chart should now look like the one in Figure 12.8. To test the chart you should put the page in user-mode by pressing the **Page User Mode** button on the page toolbar. When clicking the mouse on any particular bar, its color should change and the status line at the bottom of the Gantt chart will be adjusted accordingly.

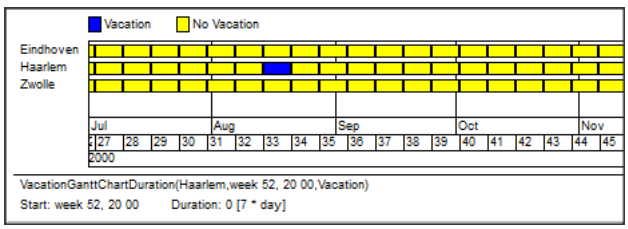


Figure 12.8: The completed vacation Gantt chart

Linking the Gantt chart

By clicking on a bar of the Gantt chart, the end-user modifies the value of the parameter VacationGanttChartDuration(f,w,v). This change in input data must be passed to the set VacationWeeks used in the mathematical program. You can accomplish this data link quite easily by providing the following statement as the **Definition** attribute of this set:

```
{ w | VacationGanttChartDuration(f,w,'Vacation') }
```

12.2.2 The holiday Gantt chart

Similar Gantt charts

The holiday Gantt chart is similar to the vacation Gantt chart. The main differences are that the holiday Gantt chart is specified in terms of days instead of weeks, and that it contains a single row rather than three.

Bar specification

The holiday Gantt chart will contain two types of bars. One bar type indicates that a particular day is an official holiday, while the other bar type denotes the opposite. These two bar types will also form the legend as shown in Figure 12.1.

Required declarations

You should now insert a new declaration section named Holiday Gantt Chart Declarations inside the section Absentee Overview. In the new declaration section the following three identifiers need to be entered:

```
Set HolidayGanttChartBarTypes {
    Index       : h;
    Definition  : data { 'Official Holiday', 'No Official Holiday' };
}

ElementParameter HolidayGanttChartStartingDay {
    IndexDomain : d;
    Range       : Days;
    Definition  : d;
}

Parameter HolidayGanttChartDuration {
    IndexDomain : (d,h);
}
```

Toggling procedure

Then, introduce a procedure ToggleHolidayGanttChart(d) in the same way as the procedure ToggleVacationGanttChart(f,w) in the previous subsection. Its argument d should be declared as an element parameter in the set Days with **Property** attribute 'Input', and its **Body** attribute should contain the following statement:

```
HolidayGanttChartDuration(d,h) := 1 - HolidayGanttChartDuration(d,h);
```

Gantt chart boundaries ...

Due to the large number of days in the overall planning period, it is impossible to view all individual days in a single Gantt chart. Scroll bars are needed. AIMMS allows you to specify string parameters as the left and right bounds of the Gantt chart. When the string parameters are *updatable* model identifiers the values of these parameters will adjust as you scroll through time. Note that the bound parameters of the vacation Gantt chart in the previous subsection were string parameters with a definition and are therefore not updatable. Their values cannot be changed and, as a result, AIMMS does not show any scroll bars.

... need to be declared

Please add the following two declarations to the Holiday Gantt Chart Declarations section:

```
StringParameter HolidayGanttChartLeftBound;

StringParameter HolidayGanttChartRightBound;
```

Gantt chart initialization

Both bound parameters plus the duration parameter need to be initialized in a new procedure InitializeHolidayGanttChart. You can place this procedure directly underneath the procedure ToggleHolidayGanttChart. The **Body** attribute should be specified as follows:

```
HolidayGanttChartLeftBound  := BeginDateOfCalendar;
HolidayGanttChartRightBound := "2000-08-01";

HolidayGanttChartDuration(d,'No Official Holiday') :=
    1 - HolidayGanttChartDuration(d,'Official Holiday');
```

Note that the duration parameter initialization is identical to the one in the vacation Gantt chart.

Adding to MainInitialization

At this point you should go back to the MainInitialization procedure, and add the statement InitializeHolidayGanttChart; at the end of its **Body** attribute. As shown previously, you can quickly locate this procedure in your model tree by pressing the *Ctrl-F* key combination or by pressing the **Find** button  on the toolbar

Executing the procedure

To prevent any initialization error while specifying the Gantt chart, you should now execute the InitializeHolidayGanttChart procedure by selecting it in the model tree and issuing the **Run Procedure** command from the right-mouse pop-up menu.

Model tree

Figure 12.9 shows part of the model tree that contains the declarations associated with the holiday Gantt chart.

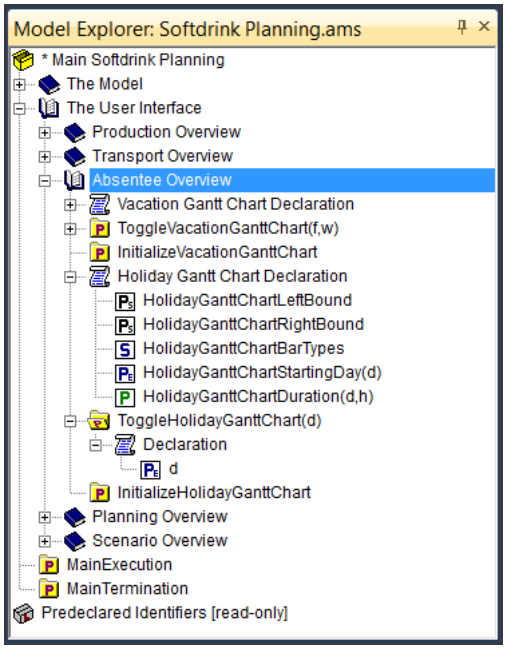


Figure 12.9: The contents of the `Absentee Overview` section

Creating the holiday chart

You are now ready to actually create the holiday specification Gantt chart underneath the vacation specification Gantt chart following the steps below:

- open the *`Absentee Overview`* page in **Edit** mode,
- press the **New Gantt Chart** button on the toolbar,
- drag a rectangle that matches the desired Gantt chart size on your page, and
- use the **Wizard** buttons to complete the **Gantt Chart** dialog box as shown in Figure 12.10.

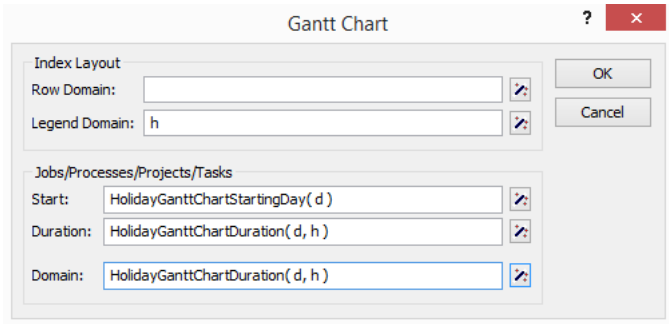


Figure 12.10: The **Gantt Chart** dialog box for holiday planning

Specifying the x-axis

The x-axis of the Gantt chart will initially display the descriptions of the elements in the calendar 'Days'. To change the reference of time to days, months and years along the x-axis in the Gantt chart, execute the following steps:

- select the Gantt chart,
- open its **Properties** dialog box,
- select the **X-axis** tab,
- select 'Real-time Calendar' as the 'Type of X-axis',
- check 'Days (Sun-Sat)', 'Days (1-31)', 'Months' and 'Years' as illustrated in Figure 12.11,
- select 'days' as the 'Unit of Measurement',
- use the **Wizard** button to select the 'String Parameter' `BeginDateOfCalendar` as the 'Reference Time',
- use the **Wizard** button to select the 'String Parameter' `HolidayGanttChartLeftBound` as the 'Left Bound',
- use the **Wizard** button to select the 'String Parameter' `HolidayGanttChartRightBound` as the 'Right Bound', and
- press the **Apply** button.

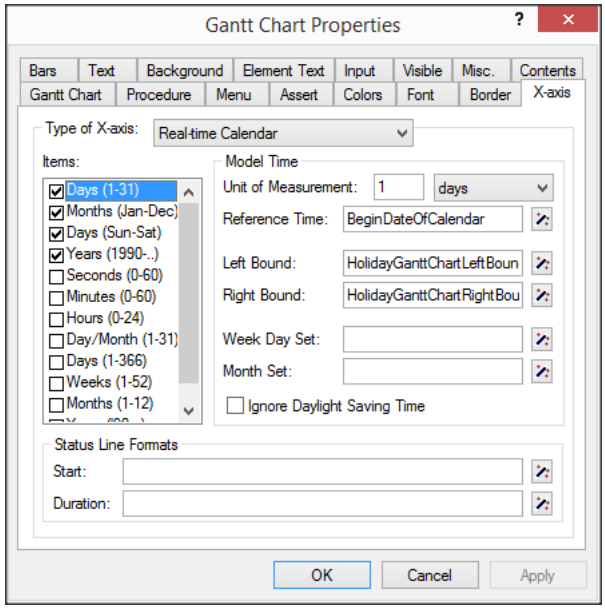


Figure 12.11: The **X-axis** tab of the **Gantt Chart Properties** dialog box

Viewing the holiday chart

Once you have followed the instructions in the previous two paragraphs, your screen should resemble the picture shown in Figure 12.12.

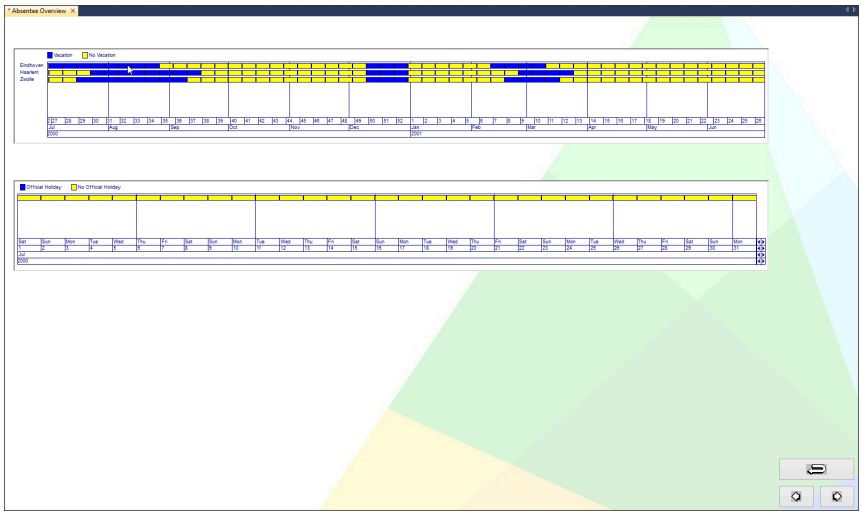


Figure 12.12: The holiday and vacation specification page

Implementing automatic toggling

To implement automatic toggling between the 'Official Holiday' and 'No Official Holiday' bar types, you should complete the **Procedure** tab as in Figure 12.13.

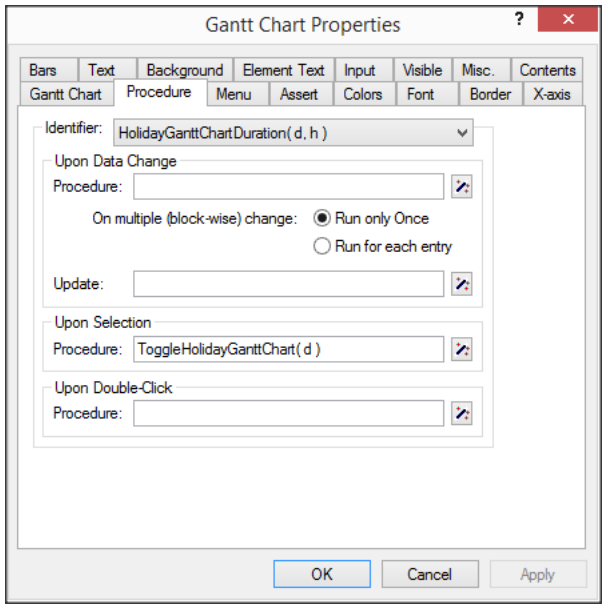


Figure 12.13: The **Procedure** tab of the **Gantt Chart Properties** dialog box

Linking the Gantt chart

By clicking on a bar of the Gantt chart, the end-user modifies the value of the parameter `HolidayGanttChartDuration(d,h)`. This change in input data must be passed to the set `OfficialHolidays`, declared in Chapter 6, and used inside the mathematical program. You can accomplish this data link quite easily by using the following statement as the **Definition** attribute of the set `OfficialHolidays`:

```
{ d | HolidayGanttChartDuration(d,'Official Holiday') }
```

12.2.3 Completing the page

Adding four more tables

You still need to add four more tables to your current page before it resembles the one shown in Figure 12.1. These tables provide a clear summary of the vacation and holiday information as specified in the two Gantt charts.

Creating a first composite table

A composite table in AIMMS can contain several identifiers provided that they share the same index domain. The first such table that you will create however, contains only a single identifier, namely the set to display all vacation weeks for the 'Eindhoven' factory. To create this table you should perform the following actions:

- make sure the page is in **Edit** mode,
- press the **New Composite Table** button ,
- draw a rectangle on the page,
- select the set `VacationWeeks` on the first tab of the **Identifier** wizard box, and
- select 'Eindhoven' as the 'Fixed Element' of the index `f` as shown in Figure 12.14

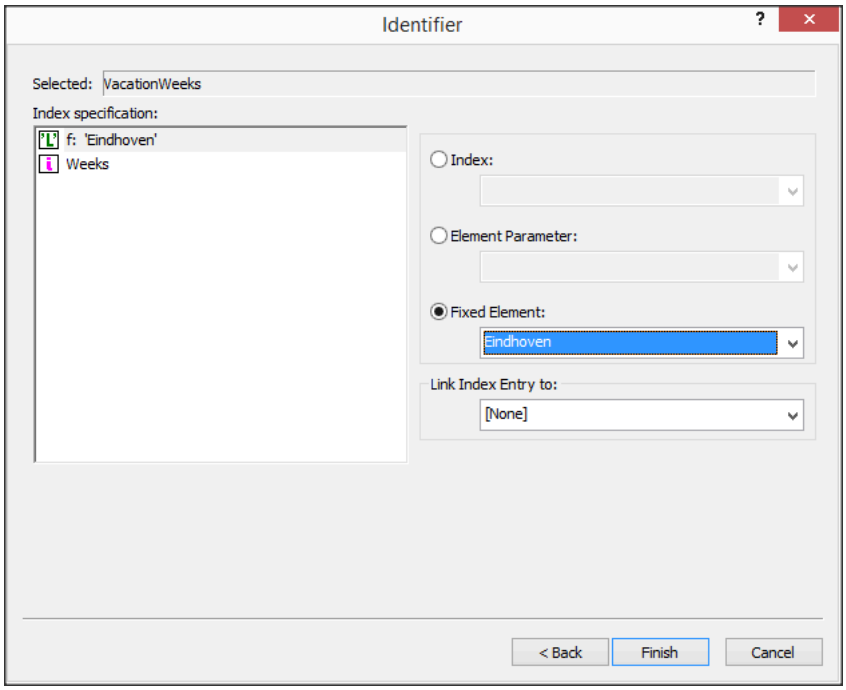


Figure 12.14: The contents of the identifier wizard box

Checking the table

Having created your first composite table, you can immediately verify its correct response to changes in the vacation Gantt chart. Simply click somewhere in the 'Eindhoven' row of the vacation Gantt chart, and the contents of the table should adjust immediately.

Copying and Pasting

To create two similar composite tables for the factory in 'Haarlem' and the factory in 'Zwolle', you can either follow the same steps, or create the tables using copy-and-paste facilities. The latter option requires the following actions:

- copy and paste the composite table for 'Eindhoven',
- open the **Properties** dialog box of the copied composite table,
- go to the **Contents** tab,
- select the domain identifier `VacationWeeks('Eindhoven',Weeks)`,

- press the **Modify** button,
- press the **Next** button in the **Identifier** wizard ,
- change the 'Fixed Element' from 'Eindhoven' to 'Haarlem' (or 'Zwolle'),
- press the **Finish** button, and
- press the **OK** button.

Creating the fourth table

You can create the fourth composite table in the same way as you created the first table. This new table should contain the set `Official Holidays`.

Enhancing the page

The page on your screen does not yet look like the one shown in Figure 12.15. If you like, you can enhance your page by, for instance, aligning the data objects, adding text objects and rectangles, and changing font sizes and colors.

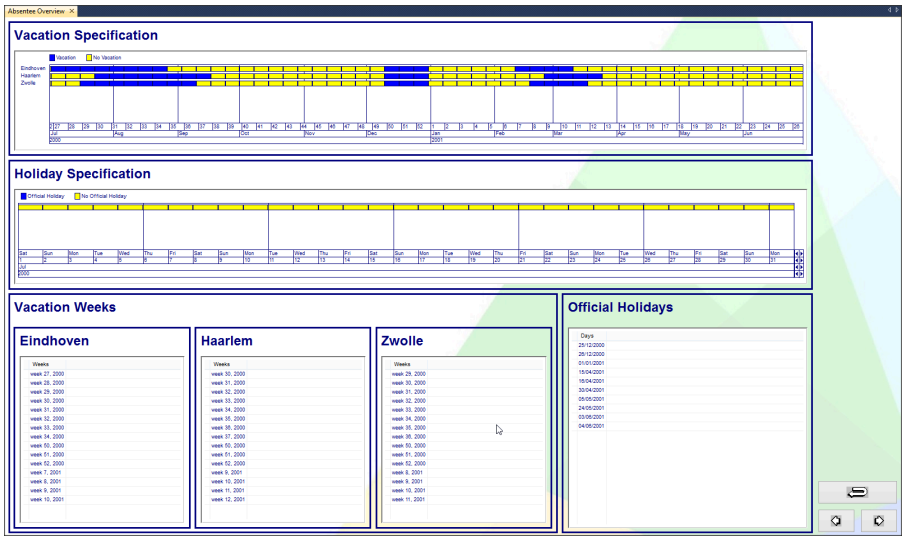


Figure 12.15: The completed *Absentee Overview* page

12.3 The Planning Overview page

Viewing the entire page

In this section the entire page as shown in Figure 12.16 will be constructed. The Gantt chart and the tables will be treated in separate subsections.

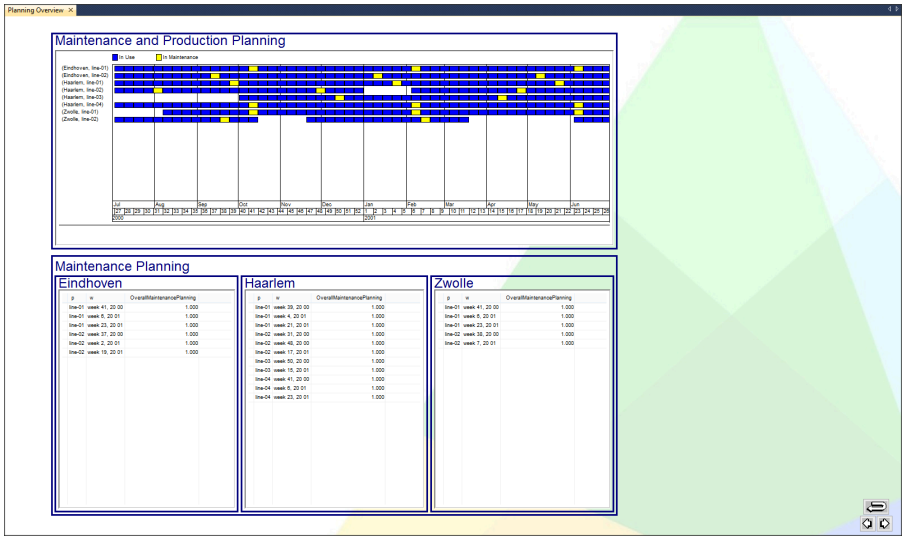


Figure 12.16: The completed *Planning Overview* page

12.3.1 The planning Gantt chart

Row and bar specification

The planning overview page should display a Gantt chart that summarizes the planning and maintenance schedule for each combination of factory and production line. Therefore, each such combination will be a row of the Gantt chart. In each row there will be two types of bars. One type of bar denotes that the corresponding production line is 'In Use', while the other type denotes that the line is 'In Maintenance'. These two bar types will form the legend in the Gantt chart.

Required declarations

The planning Gantt chart contains one new feature compared to the Gantt charts discussed earlier. In the description of each row there is a reference to two elements instead of one, namely a factory and a production line. As a result, a compound set rather than a simple set is needed to specify each row description. Please insert a new declaration section `Planning Gantt Chart Declarations` in the `Planning Overview` section, and enter the following declarations:

```
Set PlanningGanttChartRows {
    SubsetOf    : (Factories, ProductionLines);
    Index       : r;
    Definition  : {
        { (f,p) | p in FactoryProductionLines(f) }
    }
}

Set PlanningGanttChartBarTypes {
    Index       : b;
    Definition  : data { 'In Use', 'In Maintenance' };
}

ElementParameter PlanningGanttChartStartingWeek {
    IndexDomain : w;
    Range       : Weeks;
    Definition  : w;
}

Parameter PlanningGanttChartDuration {
    IndexDomain : (r,w,b);
}
```

Refreshing the planning Gantt chart

After each step in the rolling horizon procedure the zero-one parameters `OverallLineUsagePlanning(f,p,w)` and `OverallMaintenancePlanning(f,p,w)` are both updated to contain the planning information of the first week of the planning horizon as produced by the mathematical program. It is precisely this 'first week' information that is needed to update the corresponding 'duration' parameter used in redrawing the planning Gantt chart. Once the duration parameter has been updated, AIMMS will automatically refresh the Gantt chart on the *Planning Overview* page.

Update procedure

You should now insert a new procedure `UpdatePlanningGanttChart(iw)` in the `Planning Overview` section of the model (as shown in Figure 12.17). Its argument `iw` should be declared as an element parameter in the set `Weeks` with **Property** attribute 'Input'. Its **Body** attribute should contain the following statements:

```
PlanningGanttChartDuration(f,p,iw,'In Use') := 1  onlyif
   (OverallLineUsagePlanning(f,p,iw) and not OverallMaintenancePlanning(f,p,iw));

PlanningGanttChartDuration(f,p,iw,'In Maintenance') := 1  onlyif
   OverallMaintenancePlanning(f,p,iw);
```

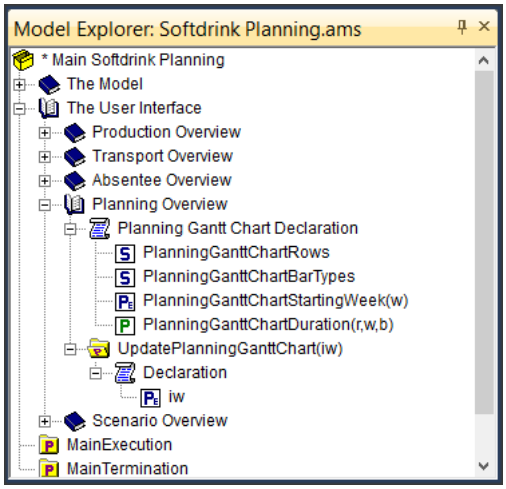


Figure 12.17: The `Planning Overview` section of the model tree

Inserting the update procedure

The above `UpdatePlanningGanttChart(iw)` procedure needs to be run after each step of the rolling horizon process. Due to its link with the parameters `OverallLineUsagePlanning(f,p,w)` and `OverallMaintenancePlanning(f,p,w)`, it is logical to insert the procedure call as the last statement inside the procedure `RegisterInOverallPlanning(iw,ip)` as shown in Figure 12.18.

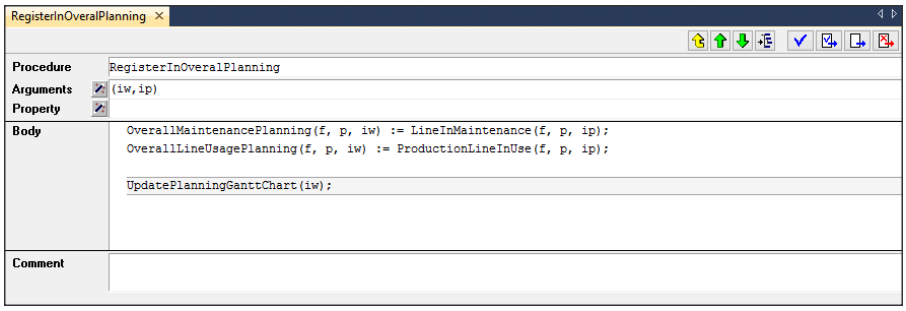


Figure 12.18: The **Body** attribute of the procedure `RegisterInOverallPlanning`

Creating the planning chart

You are now ready to create the maintenance planning Gantt chart on the *`Planning Overview`* page by following the steps outlined below.

- open the *`Planning Overview`* page in **Edit** mode,
- press the **New Gantt Chart** button on the toolbar,
- drag a rectangle that matches the desired Gantt chart size on your page, and
- use the **Wizard** buttons to complete the **Gantt Chart** dialog box as shown in Figure 12.19.

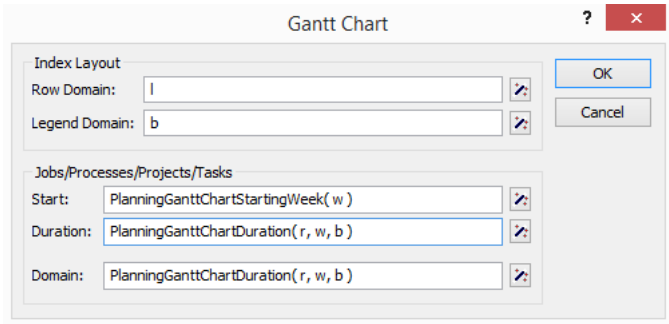


Figure 12.19: The **Gantt Chart** dialog box for the maintenance planning Gantt chart

Specifying the x-axis

The x-axis of the planning Gantt chart should be the same as in the vacation specification Gantt chart discussed earlier, namely with references to weeks, months and years. To change the current time reference along the x-axis of the Gantt chart, you should execute the following steps:

- select the Gantt chart,
- open its **Properties** dialog box,
- select the **X-axis** tab,
- select 'Real-time Calendar' as the 'Type of X-axis',
- check 'Weeks', 'Months' and 'Years' as in Figure 12.20,
- enter "2000-06-26" (with the quotes) as the 'Reference Time',
- select `BeginDateOfCalendar` as the 'Left Bound',
- select `EndDateOfCalendar` as the 'Right Bound', and
- press the **Apply** button.

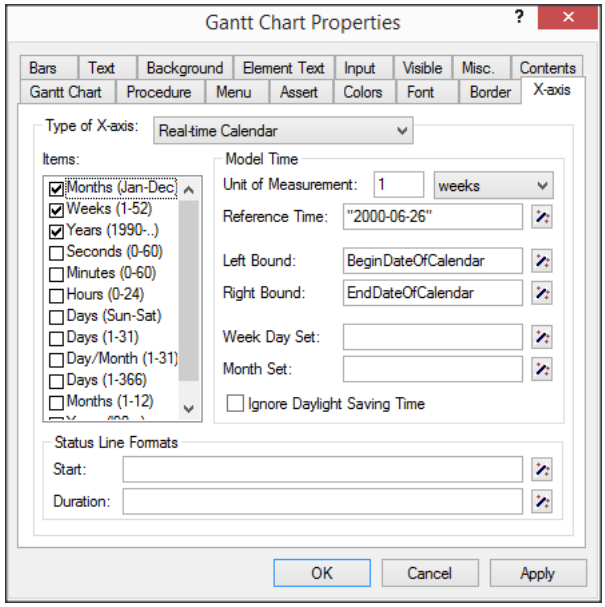


Figure 12.20: The **X-axis** tab of the **Gantt Chart Properties** dialog box

12.3.2 Completing the page

Adding three tables

Once you have finished the planning overview Gantt chart, all that is left to do is to add the three composite tables shown in Figure 12.21. Add the three tables displaying the identifiers

- `OverallMaintenancePlanning('Eindhoven',p,w)`,
- `OverallMaintenancePlanning('Haarlem',p,w)`, and
- `OverallMaintenancePlanning('Zwolle',p,w)`

in the same way that you added such tables on the *`Absentee Overview`* page.

Enhancing your page

The page on your screen does not yet look like the one shown in Figure 12.21. If you like, you can enhance your page by, for instance, aligning the data objects, adding text objects and rectangles, and changing font sizes and colors.

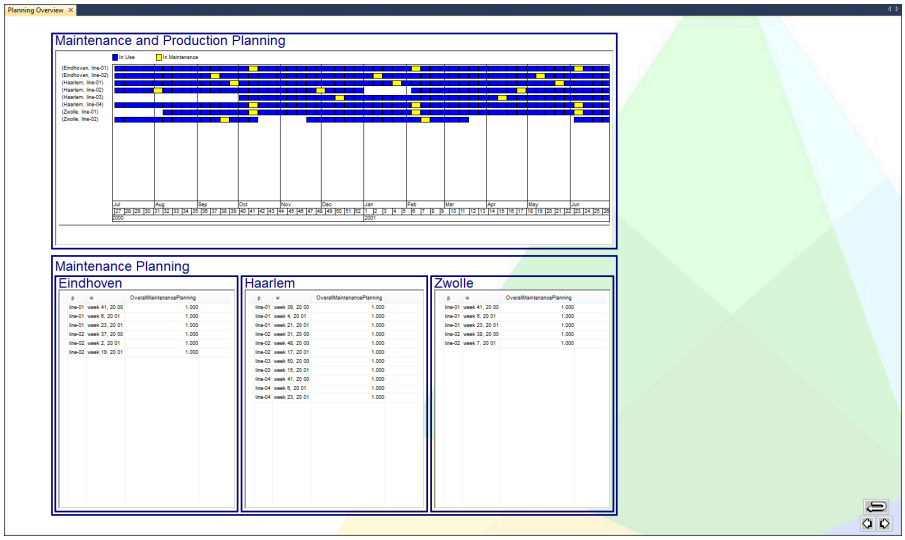


Figure 12.21: The completed *Planning Overview* page

Chapter 13

Building User-Menus

This chapter

In this chapter you will enhance the end-user interface by adding a menubar to your application.

13.1 Menu management

Menubars

A *menubar* is displayed as a horizontal bar at the top of a page, and contains pop-up menus to activate commands. Menus can be opened using point-and-click actions.

Toolbars

A *toolbar* is an optional horizontal bar positioned just below the menubar, and contains a row of bitmap buttons. These buttons provide easy access to the most frequently used commands.

Pop-up menus

A *pop-up menu* consists of a set of menu items and other pop-up menus. Pop-up menus are opened from menubars and right-mouse actions.

Menu items

Menu items represent the commands that are actually executed. They contain text describing the command plus details of an optional shortcut to activate the command from the keyboard.

Separators

Separators are used to structure menu items within a pop-up menu. Separators are visible as horizontal separation lines in pop-up menus or as spaces between buttons on toolbars (see Figure 13.1).

Default bars

By default, an AIMMS page in **User** mode will contain the menubar and toolbar as shown in Figure 13.1.

Figure 13.1: The default page menubar and toolbar

13.2 The Softdrink Planning menubar

Designing for end-users

In general, you design menubars and toolbars for your end-users to use in **User** mode. Developer-specific commands, such as providing access to the model tree, should not appear on end-user pages.

Using conventions

When you structure your menubars, you should try to adhere to acceptable conventions wherever possible. In addition, your end-users will find it easier if menubars are consistent across pages. A typical example of a convention is to include an **Exit** command as the last menu item in the first menu of the menubar.

Menubar structure

The menubar structure that you will use in this tutorial contains the following seven menus:

- the **File** menu for backups, printing and quitting,
- the **Edit** menu for performing common edit manipulations,
- the **Data** menu for storing and retrieving data,
- the **Run** menu to control the rolling horizon process,
- the **Overview** menu to provide easy access to the other pages,
- the **Window** menu to keep track of open windows, and
- the **Help** menu to provide application-specific help.

The AIMMS *Menu Builder*

User menus are created and specified using the AIMMS **Menu Builder**. This tool displays a tree that contains all menubars and toolbars in a hierarchical fashion. The look and feel of this menu tree is similar to the other tree-based AIMMS tools.

Opening the Menu Builder

To create the desired menubar structure you should first open the **Menu Builder** by pressing the **Menu Builder** button on the AIMMS toolbar or by pressing the *Ctrl+F9* key, and open the `Default Page Menubar` in the menu tree. The initial menu tree is shown in Figure 13.2.

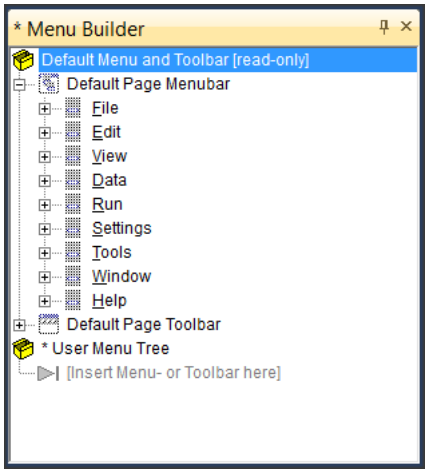


Figure 13.2: The **Menu Builder** with the initial menu tree

Default bars can help you

The `Default Page Menubar` and the `Default Page Toolbar` in the initial menu tree are read-only. This property is indicated by the disabled icons in the menu tree. Nevertheless, these bars can be used as a base construct from which you can start building your own menubars and toolbars. In this tutorial you will be asked to copy and paste several parts of the `Default Page Menubar` while creating your own `Softdrink Planning Menubar`.

Creating a menubar

To create your first menubar you should take the following actions:

- select the `User Menu Tree`,
- press the **New Menubar** button on the tool bar,
- specify 'Softdrink Planning Menubar' as its name, and
- press the *Enter* key to register this name.

13.2.1 The File menu

Menu contents

Figure 13.3 shows the proposed **File** menu containing one submenu and five menu items. The **Backup** submenu relates to the backup of data, while the **Print** menu item prints the contents of the active window. The other menu items are self-explanatory.

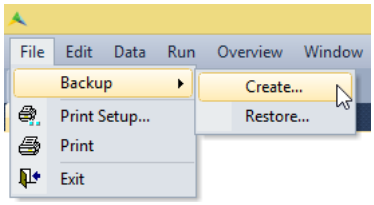


Figure 13.3: The proposed **File** menu

Creating the **File** *menu*

To create this **File** menu you need to perform the following actions:

- select the `Softdrink Planning Menubar` in the tree,
- double-click on the menubar icon to open this node,
- press the **New Menu** button,
- specify '&File' as the name of this new menu, and
- press the *Enter* key to register the name.

Ampersand character

The ampersand in the string '&File' will automatically create a shortcut triggered by the *Alt-F* key combination. The letter following the ampersand will be underlined in the actual menu (see Figure 13.3). The ampersand can be placed in front of any character in the string.

Creating the **Backup** *menu*

To create the **Backup** submenu of the **File** menu, you should follow these steps:

- select the **File** menu in the menu tree,
- double-click on the menu icon to open this node,
- press the **New Menu** button,
- specify 'Backup' as the name of this new menu, and
- press the *Enter* key to register the name.

The **Menu Builder** on your screen should resemble Figure 13.4.

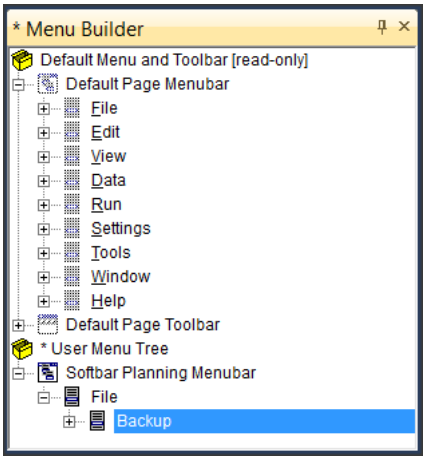


Figure 13.4: The **File** menu so far

Using duplicate menus

Duplicating existing menus and menu items offers two main advantages. First of all, duplication provides a quick and easy way to construct menus: you do not have to re-enter the corresponding menu actions. Secondly, duplicate menu items are easier to maintain, since an update of one of them is automatically propagated to all the others.

Duplicating menu items

All menu items in the **File** menu will be duplicates of already existing menu items. Please carry out the following groups of steps relating to various menu items:

- go to the **File**-**Backups**-**Data** menu of the `Default Page Menubar`,
- select the two menu items 'Create' and 'Restore' simultaneously,
- press the **Copy** button on the toolbar,
- select the **Backup** menu created previously,
- open it and click on 'Insert Menu item here', and
- select the **Paste as Duplicate** command from the **Edit** menu.

- press the minus sign in front of the **Backup** menu, and
- press the **Separator** button on the toolbar.

- go to the **File** menu of the `Default Page Menubar`,
- select the menu items **Print Setup** and **Print** simultaneously,
- press the **Copy** button on the toolbar,
- select the separator you just created, and
- select the **Paste as Duplicate** command from the **Edit** menu.

- press the **New Separator** button on the toolbar.

- go to the **File** menu of the `Default Page Menubar`,

- select the menu item **Exit**,
- press the **Copy** button on the toolbar,
- select the separator you just created, and
- select the **Paste as Duplicate** command from the **Edit** menu.

The complete **File** menu should be as shown in Figure 13.5.

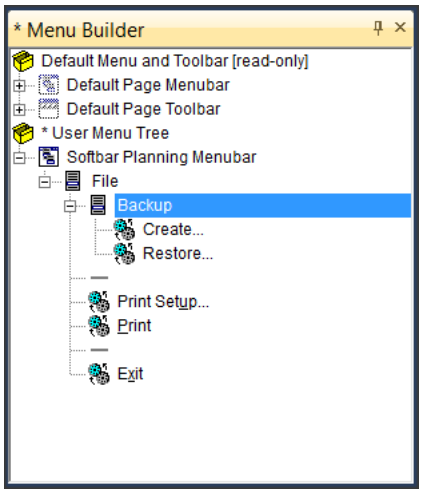


Figure 13.5: The complete **File** menu

13.2.2 The Edit and Data menus

Menu contents

The **Edit** and **Data** menus to be created should be identical to the corresponding menus already in the `Default Page Menubar`.

Creating the **Edit** *and* **Data** *menus*

To create the **Edit** and **Data** menus you should follow these steps:

- go to the `Default Page Menubar`,
- select the **Edit** and **Data** menus simultaneously,
- press the **Copy** button on the toolbar,
- select the **File** menu from the `Softdrink Planning Menubar`,
- make sure it is closed, and
- select the **Paste as duplicate** command from the **Edit** menu.

The `Softdrink Planning Menubar` with the new **Edit** and **Data** menus is shown in Figure 13.6.

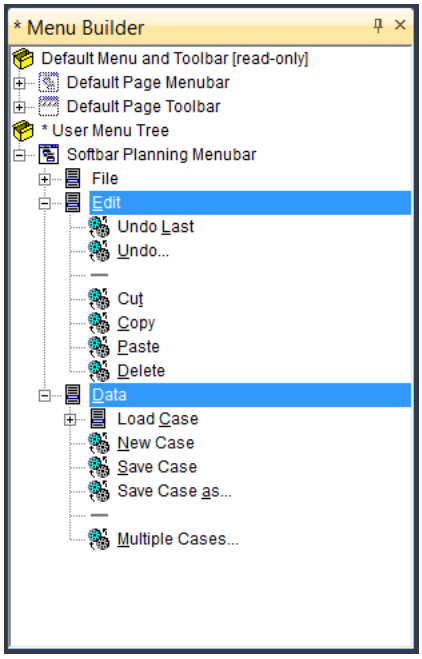


Figure 13.6: The new **File** and **Data** menu

13.2.3 The Run menu

Menu contents

The **Run** menu will contain commands to control the rolling horizon process. There are no standard actions, and you will have to create the menu items plus their actions explicitly. You should first create the three menu items plus separator, as shown in Figure 13.7 using the following steps:

- select the **Data** menu from the menu tree,
- close this menu if it is open,
- press the **New Menu** button ,
- specify '&Run' as the name of this new menu,
- press the *Enter* key to register the name,
- open it by double clicking on its icon,
- press the **New Item** button on the toolbar,
- enter 'Run Next' (unquoted) as its text,
- press again the **New Item** button on the toolbar,
- enter 'Run All' (unquoted) as its text,
- press the **New Separator** button on the toolbar,
- press once again the **New Item** button on the toolbar, and
- enter 'Restart' (unquoted) as its text.

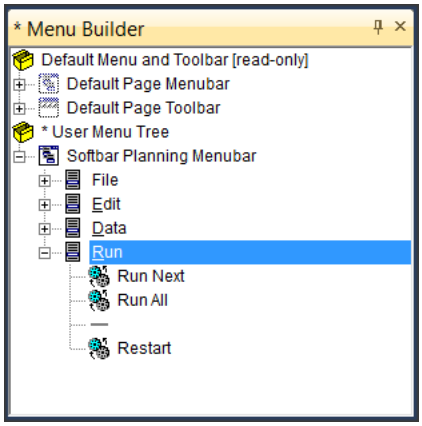


Figure 13.7: The **Run** menu

Specifying the first menu action

Having created the three menu items you now have to specify the commands that are executed when these menu items are selected. The following steps specify the command associated with the **Run Next** menu item:

- select the **Run Next** menu item,
- press the **Properties** button on the toolbar,
- press the **Actions** tab,
- select the 'Run' action,
- press the **Add** button,
- select the 'Procedure' option (not the 'Page Procedure' option),
- use the **Wizard** button to select the procedure `RollHorizonOnce`,
- press the *Finish* button, and
- press the *OK* button.

The completed **Action** tab of the **Menu Properties** dialog box should be as shown in Figure 13.8.

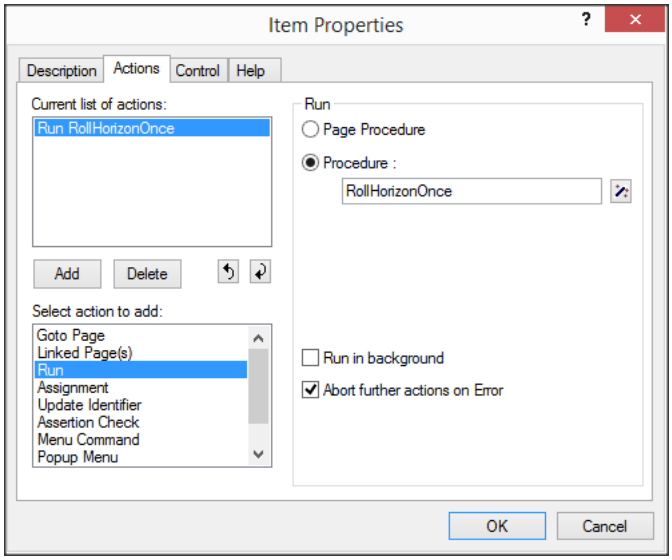


Figure 13.8: The **Action** tab of the **Menu Properties** dialog box

Specifying the remaining two menu actions

Repeat the above steps to link the procedure `RollHorizonToEnd` to the **Run All** menu item. Then repeat these steps once more to link the procedure `MovePlanningIntervalToStartOfCalendar` to the **Restart** menu item.

13.2.4 The Overview menu

Menu contents

The **Overview** menu will provide separate menu items to access each of the five overview pages. You do not need to specify these menu items separately, you can make use of the page structure in the Page Manager.

Navigation menus

The **New Navigator** button allows you to add navigation menus to your application. These navigation menus, with menu items and possibly submenus, all refer to pages. The menus are structured in the same hierarchical fashion as the corresponding pages in the **Page Manager**. As a result, navigation menus are automatically updated in AIMMS whenever the structure of pages in the page tree is modified.

*Creating the **Overview** menu*

To create the complete **Overview** menu as a navigation menu you should execute the following steps:

- select the **Run** menu from the menu tree,
- close this menu if it is open,
- press the **New Menu** button ,
- specify '&Overview' as the name of the menu,

- press the *Enter* key to register the name,
- open the new **Overview** menu,
- press the **New Navigator** button on the toolbar,
- specify 'Overview Pages' as the name of the menu, and
- press the *Enter* key to register the name.

The menu tree on your screen should look like the one shown in Figure 13.9.

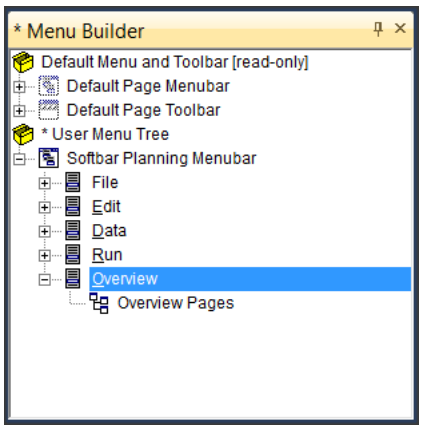


Figure 13.9: The menu tree so far

Specifying the navigation properties

To specify the pages that are to be displayed through the **Overview** menu you should perform the following actions:

- select the 'Overview Pages' navigation item from the menu tree,
- press the **Properties** button button on the toolbar,
- select the **Navigation** tab,
- select 'Other Page' as the option within 'Reference Page' (see also Figure 13.10),
- press the **Wizard** button at the right of the 'Other Page' edit field,
- select the *Contents* page, and
- press the **OK** button twice.

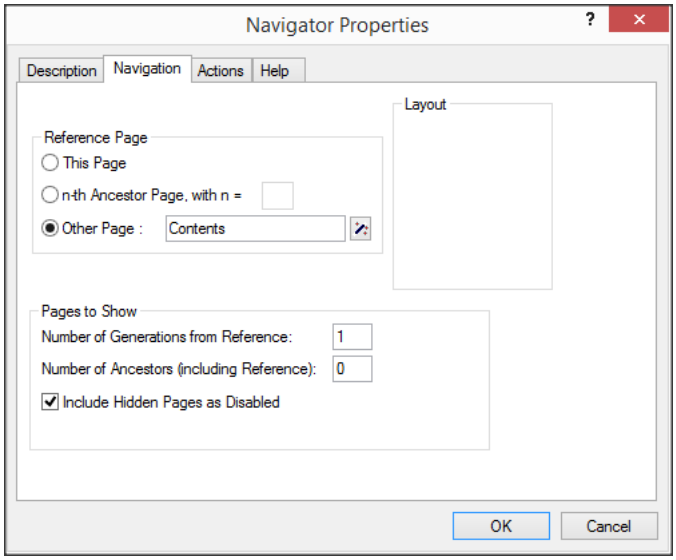


Figure 13.10: The completed **Navigation** tab of the **Menu Properties** dialog box

The resulting **Overview** menu will look like the one shown in Figure 13.11. *Viewing the* **Overview** *menu*

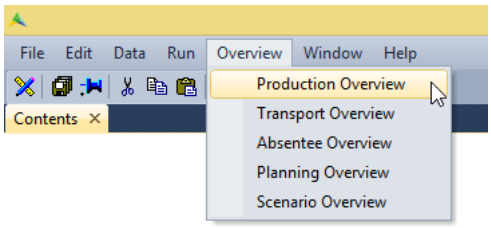


Figure 13.11: The **Overview** menu

13.2.5 The Window menu

The **Window** menu of the `Softdrink Planning Menubar` will be identical to the **Window** menu of the `Default Page Menubar`. *Menu contents*

To duplicate the **Window** menu from the `Default Page Menubar` you should perform the following actions: *Duplicating the* **Window** *menu*

- select the 'Window' menu from the `Default Page Menubar`,
- press the **Copy** button on the toolbar,
- select **Overview** menu from the `Softdrink Planning Menubar`,

- make sure it is closed, and
- select the **Paste as Duplicate** command from the **Edit** menu.

13.2.6 The Help menu

Menu contents

The contents of the **Help** menu is shown in Figure 13.12. The first menu item will open the AIMMS Help document. The second menu item will display the model summary in a PDF viewer. The third menu item will open an 'About' dialog box with some application-specific information.

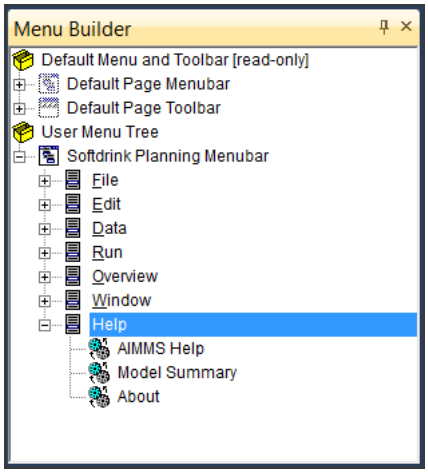


Figure 13.12: The **Help** menu in the `Softdrink Planning Menubar`

Creating the **Help** *menu*

By now, you should be able to create the **Help** menu and its three menu items on your own. Note that the three menu items should be created from scratch using the **New Item** button on the toolbar.

Specifying the first menu item

Rather than duplicating the first menu item, you are asked to specify the menu command directly by executing the following actions:

- select the 'AIMMS Help' menu item,
- press the **Properties** button on the toolbar,
- press the **Actions** tab,
- select the 'Menu Command' option,
- press the **Add** button,
- select the 'Help-Contents and Index' entry (see Figure 13.13), and
- press the **OK** button.

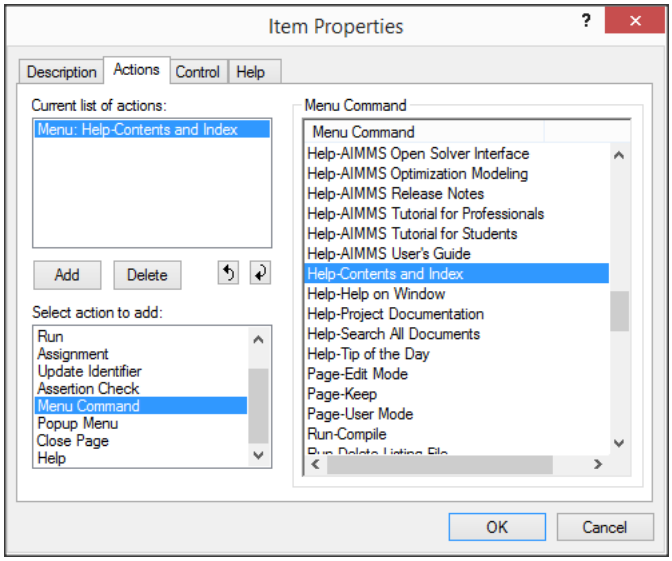


Figure 13.13: The **Action** tab of the **Menu Properties** dialog box

Creating a procedure ...

To specify the **Model Summary** menu command you need to declare an auxiliary AIMMS procedure. To keep your model tree well-organized you should first create a new model section called `Softdrink Planning Menubar` underneath the `Scenario Overview` section, and then create a procedure `ShowModelSummary` inside this section as shown in Figure 13.14. This procedure should have the following **Body** attribute:

```
ShowHelpTopic( "section.3.4", "Tutorial/AIMMS_tutorial_for_professionals.pdf" );
```

Note that you might need to change the path of the tutorial file that is passed as the second argument of the function `ShowHelpTopic`.

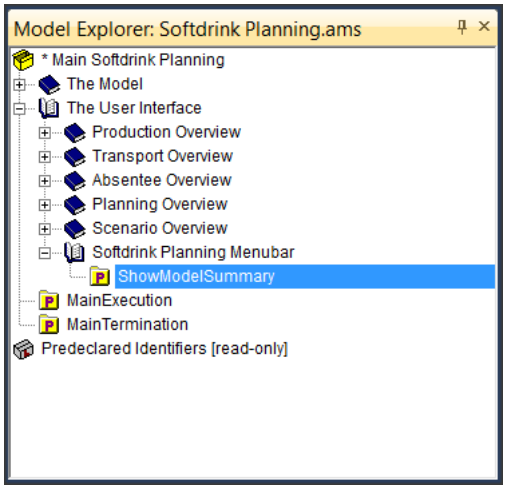


Figure 13.14: The `Softdrink Planning Menubar` section of the model tree

... and specifying the second menu item

You are now ready to link the procedure you have just created to the **Model Summary** menu command using the following actions:

- select the 'Model Summary' menu item,
- press the **Properties** button on the toolbar,
- press the **Actions** tab,
- select the 'Run' action,
- press the **Add** button,
- select the 'Procedure' option,
- use the **Wizard** button to select the procedure `ShowModelSummary`,
- press the **Finish** button, and
- press the **OK** button.

Creating a dialog page ...

The last item in the **Help** menu opens an 'About' dialog box providing some application-specific information such as a version number or copyright information. In AIMMS you can create a *dialog page* with the following actions:

- open the **Page Manager**,
- create a new page with the name 'About Softdrink Planning' (see Figure 13.15),
- open the page in **Edit** mode,
- open the **Page Properties** dialog box
- check the 'Behaves as Dialog' checkbox underneath 'Style',
- press the **OK** button, and
- resize it to give a reasonably sized dialog box.

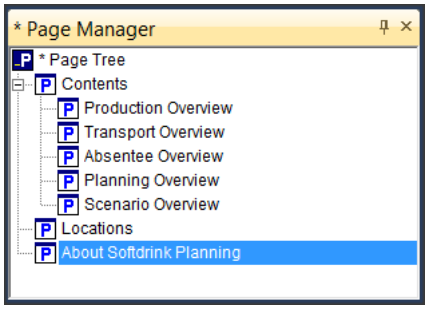


Figure 13.15: The page tree with the new *About Softdrink Planning* dialog page

Providing its contents . . .

You can insert whatever contents into the *About Softdrink Planning* dialog page you want. Figure 13.16 serves as an example, and contains a **Close** button, a logo, plus text displaying information about the application. This page is also available for import from the 'Pages' subdirectory. The page import process was described in the last section of the previous chapter.

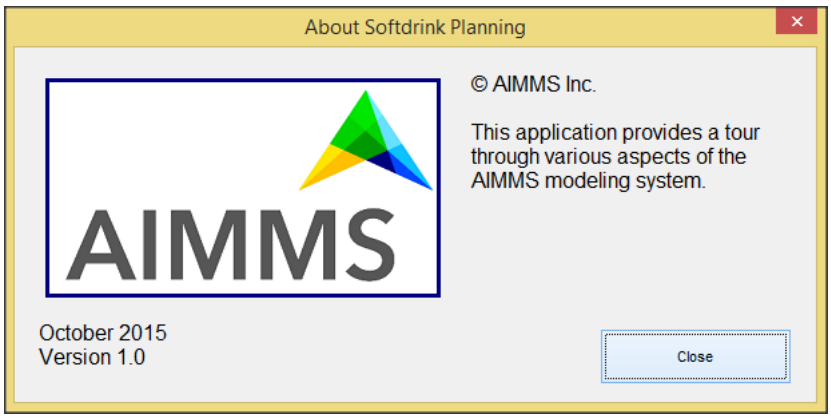


Figure 13.16: The **About Softdrink Planning** dialog box

. . . and specifying the third menu item

Please specify the third menu command by performing the following steps:

- select the **About** menu item,
- press the **Properties** button on the toolbar,
- press the **Actions** tab,
- select 'Linked Page(s)' as the action to add,
- press the **Add** button,
- press the **New Page Link** button ,
- select the *About Softdrink Planning* page (see Figure 13.17), and
- press the **OK** button twice in a row.

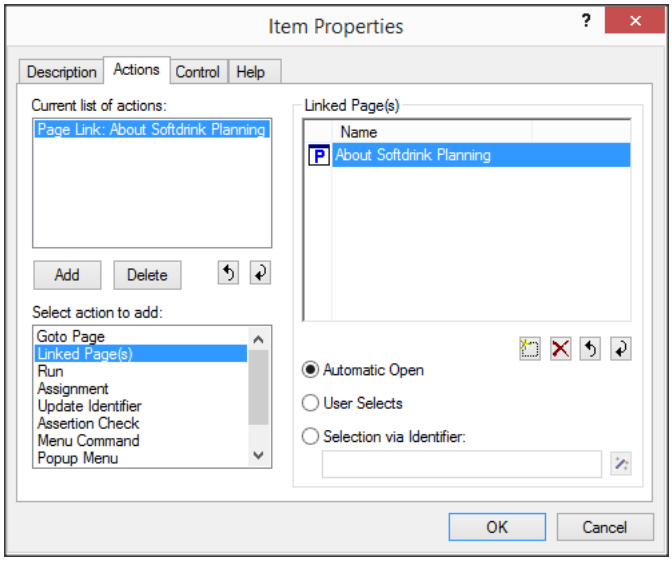


Figure 13.17: The **Action** tab of the menu **Item Properties** dialog box

13.2.7 Linking the menubar to pages

Instead of linking to pages ...

You have now completed the specification of the `Softdrink Planning Menubar`. Instead of linking this menubar to each individual page, it is much more convenient to link it to the *`Background Color`* template. This template is shared by all pages, and menubars on pages are, by default, inherited from templates.

... link to a single template

To link the menu bar to the *`Background Color`* template the following actions are required:

- open the *`Background Bitmap`* template in **Edit** mode,
- open its **Page Properties** dialog box,
- select the **Menu** tab,
- select 'Other' as the Menu Bar option (see Figure 13.18),
- press the **Wizard** button on the right of the 'Other' edit field,
- select `Softdrink Planning Menubar`, and
- press the **OK** button twice.

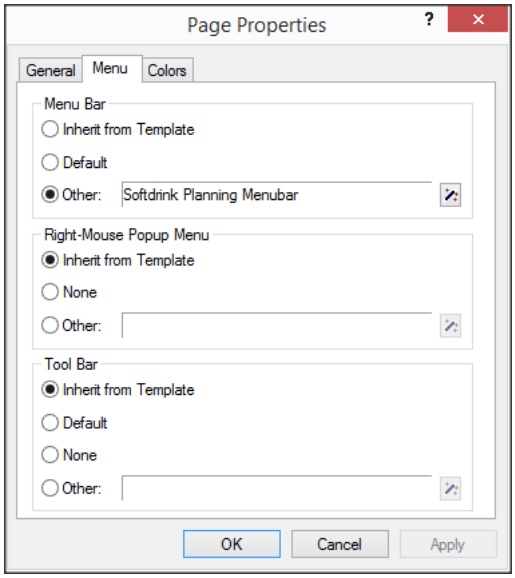


Figure 13.18: The **Menu** tab of the **Page Properties** dialog box

Viewing the result

You are now ready to use the newly created menubar. Change the page mode by pressing the **Page User Mode** button  on the toolbar. The `Softdrink Planning Menubar` created in this chapter should appear on all your pages, and is shown in Figure 13.19.

Figure 13.19: The complete 'Softdrink Planning Menubar'

Chapter 14

Data Management

This chapter

In this chapter, you will learn how to manage your model data using *cases*. Such management is typically based on using menu commands. You will also write a procedure to generate cases automatically during an AIMMS session. These cases are then viewed and compared in a multiple case overview.

14.1 Storing the solution in a case

What is a case?

A *case* is a set of data values at an instant in time and contains the values of a subset of all model identifiers. Such a subset is referred to as a *case type*. The default case type is the set of all identifiers. Cases enable you to save intermediate data values for inspection at a later moment. You can also use a case to continue your work during a later AIMMS session.

Creating a case

Following an iteration of the rolling horizon process, initiated by pressing the **Run Next** button, you can save both your input and the solution values in a new case by executing the following steps:

- select the **Save Case as**... command from the **Data** menu,
- specify 'Solution After First Roll.data' (without the quotes) in the 'File Name' edit field, and
- press the **Save** button (see Figure 14.1).

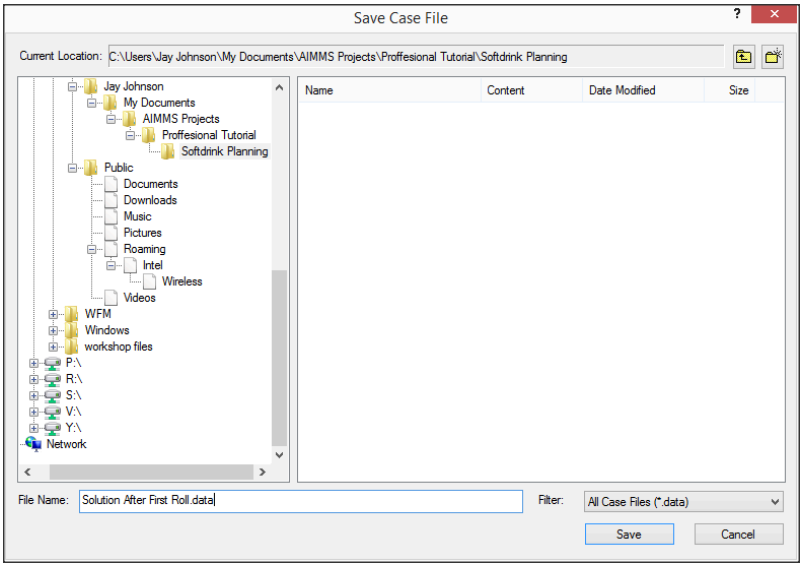


Figure 14.1: Creating your first case

Loading a case

The following commands close and re-open your AIMMS project. Then, by loading the case you have just saved, you will have incorporated all your current data values. Please follow these instructions:

- change to the default page menubar by setting the current page to **Edit** mode,
- select the **Close Project** command from the **File** menu,
- open the project again,
- select the **Load Case** submenu from the **Data** menu,
- select the **as Active**... command,
- select the 'Solution After First Roll.data' entry from the list box, and
- press the **Open** button (see Figure 14.2).

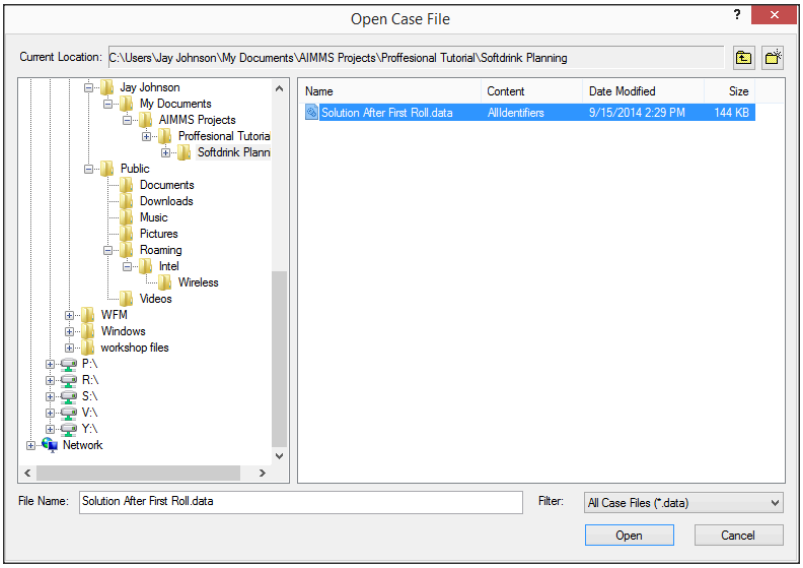


Figure 14.2: Loading your first case

The active case

In AIMMS, all the data that you are currently working with are referred to as the *active case*. The name of the currently active case is displayed in the status bar at the bottom of the AIMMS window as shown in Figure 14.3.

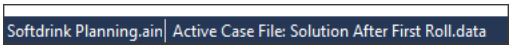


Figure 14.3: Part of the AIMMS status bar

14.2 Saving holidays and vacations in a case file

Creating an identifier . . .

First you need to declare and specify a subset of `AllIdentifiers` with the identifiers for the vacation and holidays. Please create a model section named `Data Management` directly underneath the section `Softdrink Planning Menubar`. In this section create a declaration and name it `Data Management Declaration`. There you will put the new set called `VacationAndHolidayIdentifiers`.

. . . and specify its contents

To specify which model identifiers are to be stored in the new case file you need to take the following actions:

- open the attribute of the set `VacationAndHolidayIdentifiers` you just created,
- in the **Subset of** open the **Wizard** and in the dialog box type 'AllIdentifiers',

- press the **OK** button to close the Wizard,
- in the **Definition** open the **Wizard** and select the HolidayGanttChartDuration and VacationGanttChartDuration identifiers from the '*Subset of: AllIdentifiers*' list,
- press the **Close** and the **OK** to close the Wizard
- finally press the Check, Commit and Close button.

Specifying a case ...

Next, you should open the *Absentee Overview* page in **User** mode, and specify the vacation weeks and official holidays as listed in Table 14.1 by clicking on the two Gantt charts.

Vacation Weeks			Official
Eindhoven	Haarlem	Zwolle	Holidays
week 27, 2000	week 30, 2000	week 29, 2000	Dec 25, 2000
week 28, 2000	week 31, 2000	week 30, 2000	Dec 26, 2000
week 29, 2000	week 32, 2000	week 31, 2000	Jan 1, 2001
week 30, 2000	week 33, 2000	week 32, 2000	Apr 15, 2001
week 31, 2000	week 34, 2000	week 33, 2000	Apr 16, 2001
week 32, 2000	week 35, 2000	week 34, 2000	Apr 30, 2001
week 33, 2000	week 36, 2000	week 35, 2000	May 5, 2001
week 34, 2000	week 37, 2000	week 36, 2000	May 24, 2001
week 50, 2000	week 50, 2000	week 50, 2000	Jun 3, 2001
week 51, 2000	week 51, 2000	week 51, 2000	Jun 4, 2001
week 52, 2000	week 52, 2000	week 52, 2000	
week 7, 2001	week 9, 2001	week 8, 2001	
week 8, 2001	week 10, 2001	week 9, 2001	
week 9, 2001	week 11, 2001	week 10, 2001	
week 10, 2001	week 12, 2001	week 11, 2001	

Table 14.1: Vacation weeks and official holidays

... and saving it

To save the holiday and vacation data you have just specified you will need to create a new procedure in the Data Management section and name it HolidayAndVacationDataSave and specify the following statement in its **Body** attribute:

```
CaseFileSave(
    url                    :   "Cases\\Vacation and Holidays.data",
    contents               :   VacationAndHolidayIdentifiers);
```

Selecting a startup case

To load the Vacation and Holidays.data case file during project startup, you will need to make it a startup case in the **AIMMS Options** dialog box. You should follow the same steps used when you specified a startup page at the end of Chapter 10. The corresponding **Options** dialog box is shown in Figure 14.4.

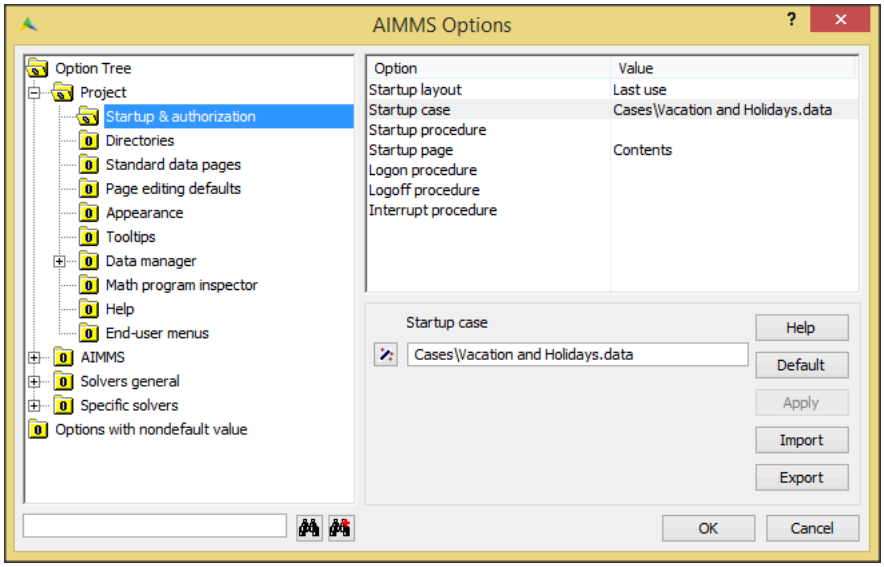


Figure 14.4: The **AIMMS Options** dialog box

14.3 Automatic case generation

This section

In this section, you will first build your own procedure that automatically generates cases. After this, you will develop an experiment in which you will study the effect of the length of the planning horizon on the total cost of running the company. Finally, you will create a multiple case object to view and compare the results of this investigation.

'What If' experiments

In a typical 'What If' experiment, you want to study the output of your model as a result of changes in data input. You can perform such an experiment through an interactive session. If the experiment is extensive and/or requires a great deal of CPU time, an alternative approach is to write a procedure to execute the entire experiment. It is then important to save the results in cases that are generated as the experiment evolves. The following paragraphs will show you how to construct an extensive experiment using an automatic case saving procedure.

Declaring required identifiers

The total cost of running the company will be the output of an experiment in which the length of the planning horizon is changed from 4 to 10 weeks. Please create a `Data Management Declarations` declaration section underneath the `Data Management` section in the model tree (see Figure 14.5) and declare the following identifiers in this declaration section:

```
ElementParameter CurrentPeriod {
```

```
    Range       : Periods;
}

Parameter TotalCostInCurrentPeriod {
    Unit        : $;
    Definition  : {
        sum[ s, ScenarioProbability(s) * (
        sum[ (f,p), FixedCostDueToLeaveChange *
                    ProductionLineLevelChange(f, p, CurrentPeriod) ] +
        sum[ f, UnitProductionCost(f) * Production(f, CurrentPeriod) ] +
        sum[ l, UnitStockCost(l) * Stock(l, CurrentPeriod, s) ]+
        sum[ (f,c), UnitTransportCost(f, c) * Transport(f, c, CurrentPeriod , s) ] ) ]
    }
}

Parameter AccumulatedTotalCost {
    Unit        : $;
}

Set AccumulateTotalCostIdentifiers{
    Subset of    :  AllIdentifiers
    Definition   :  'AccumulatedTotalCost'
}
```

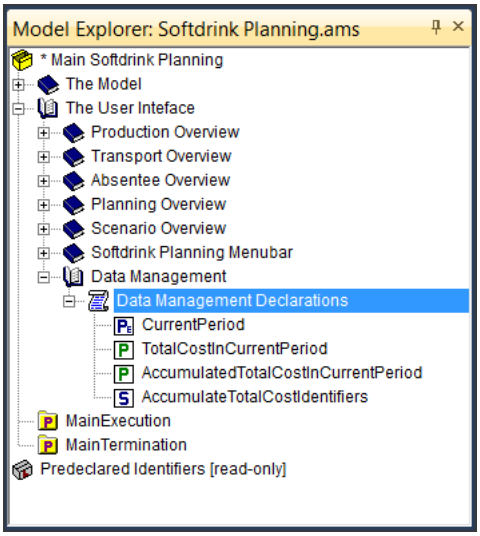


Figure 14.5: The `Data Management Declarations` section

Creating a new case type

To create a case that contains only a single identifier, namely `AccumulatedTotalCost`, you have to perform the following actions:

- create a set in the `Data Management Declarations` and name it `AccumulatedTotalCostIdentifiers`
- set `AllIdentifiers` in the **Subset of** attribute
- select the `AccumulatedTotalCost` identifier on the **Body** attributes wizard

- press the Check, Commit and Close button.

Building a SaveCase procedure ...

Next you need to create a procedure called SaveCase(CaseName) as shown in Figure 14.6. Use the **Argument** wizard to declare CaseName as a string parameter with property 'Input'. The **Body** attribute of the new procedure should be entered as follows:

```
CaseFileSave(
     url : FormatString("Cases\\\%s.data", CaseName),
     contents :  AccumulateTotalCostSet);
```

As noted previously, you can find explanations of predefined AIMMS functions in *The Function Reference*.

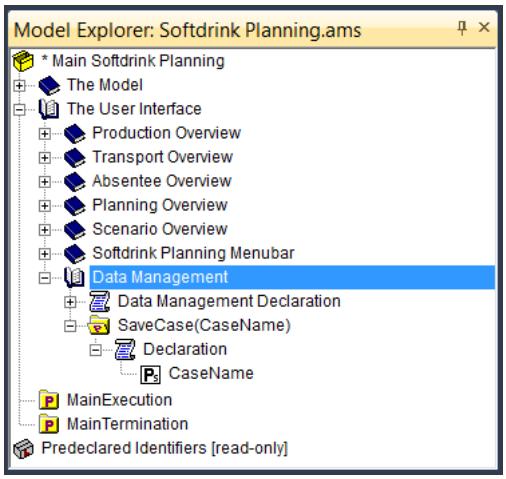


Figure 14.6: The SaveCase procedure in the model tree

... and specifying the experiment

Finally, you are now ready to specify the procedure RunExperiment in the Data Management section as shown in Figure 14.7. The contents of this procedure are extensive, but should be mostly self-explanatory. Note the use of the previously specified SaveCase procedure inside the following **Body** attribute:

```
NumberOfPeriodsInPlanningInterval := 4;

repeat "outer-loop"

    MovePlanningIntervalToStartOfCalendar;
    AccumulatedTotalCost := 0;
    CurrentPeriod := FirstPeriodInPlanningInterval;

    while ( LastWeekInPlanningInterval < LastWeekInCalendar ) do "inner-loop"

        RollHorizonOnce;
        AccumulatedTotalCost += TotalCostInCurrentPeriod;
```

```
        PageRefreshAll;
        break "inner-loop" when ( LeastCostPlan.ProgramStatus <> 'Optimal' );

    endwhile;

    if ( LeastCostPlan.ProgramStatus <> 'Optimal' ) then
        AccumulatedTotalCost := 0;
    else
        for (t | t > FirstPeriodInPlanningInterval) do
            CurrentPeriod := t;
            AccumulatedTotalCost += TotalCostInCurrentPeriod;
        endfor;
    endif;

    SaveCase(formatstring("Length-\%n", NumberOfPeriodsInPlanningInterval ));

    break "outer-loop" when ( NumberOfPeriodsInPlanningInterval = 10 );

    NumberOfPeriodsInPlanningInterval += 1;

endrepeat;
```

The completed `Data Management` section of the model tree should be as shown in Figure 14.7.

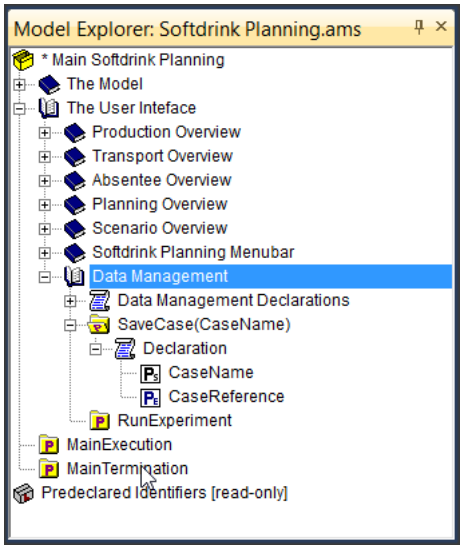


Figure 14.7: The final `Data Management` section

Preparing to run the experiment

Execution of the above experiment may take a while, depending on the speed of your computer. However, before executing the experiment, you should first comment out the `halt with` part of the `solve` statement in the procedure `SolveLeastCostPlan`. This line is useful to give an appropriate error message when solving the model for one particular period, but we don't want the experiment

to stop running upon finding a non-optimal solution for a certain period. The `break "inner-loop"` statement takes care of such situations in the `RunExperiment` procedure. To comment out this block, please do the following:

- locate the `SolveLeastCostPlan` procedure, by using the **Find** function of AIMMS,
- select the 3 lines of the `halt` clause of the `solve` statement, as illustrated in Figure 14.8,
- open the right-mouse pop-up menu and select **Comment Block**, and
- add a semicolon after the `SolveLeastCostPlan` statement.

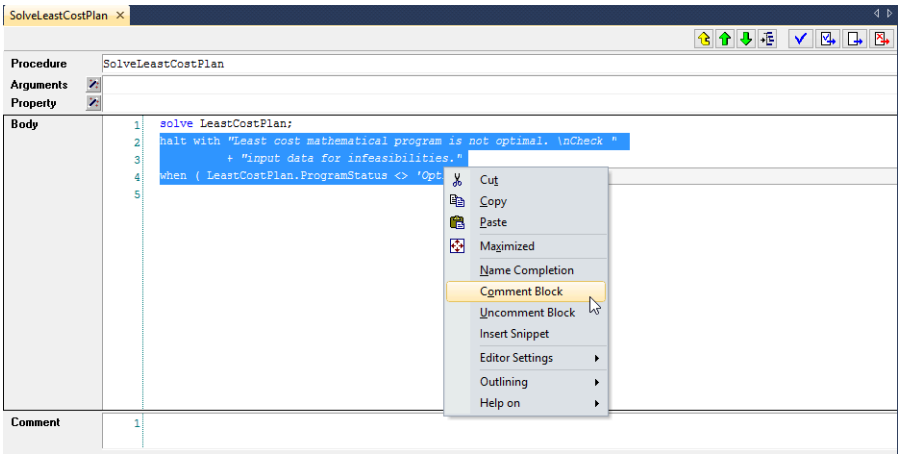


Figure 14.8: Commenting out the `halt` clause

Running the experiment

To initiate the actual experiment, you should perform the following actions:

- select the `RunExperiment` procedure node in the model tree, and
- select the **Run Procedure** command from the right-mouse pop-up menu.

The run could produce a number of warnings about the model being infeasible or unbounded. This is caused by some subproblems in the inner loop of the experiment having become insolvable. You can ignore these warnings for this tutorial.
After the experiment is complete, several cases should have been created in the 'Cases' directory in your project directory.

Creating a table ...

You are now in a position to create a table that displays the value of the parameter `AccumulatedTotalCost` for every case that has been generated during the experiment by executing the following steps:

- create a new page at the bottom of the page tree,
- enter 'Multiple Case Overview' as its name,

- press the *Enter* key to register the name,
- open the new page in edit mode,
- press the **New Table** button on the toolbar,
- draw a rectangle on the page, and
- select the parameter `AccumulatedTotalCost`.

... into a multiple case table

To transform this table into a *multiple case object*, you should do the following:

- open the **Table Properties** dialog box of the table object,
- select the **Table** tab if necessary,
- check the 'Multiple Case Object' checkbox (see Figure 14.9), and
- press the **OK** button.

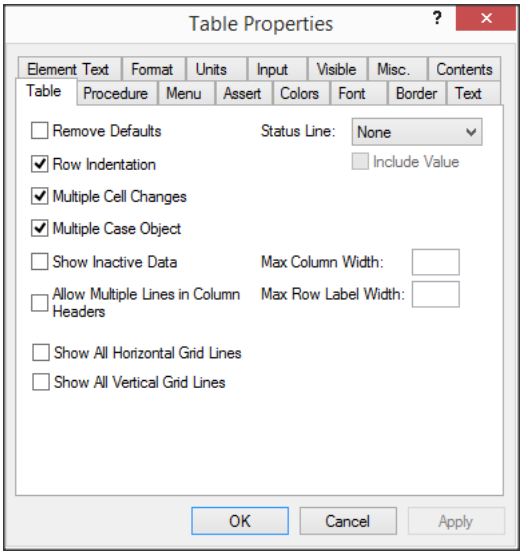


Figure 14.9: Creating a multiple case table

Specifying the case selection

The table should have been extended with an empty column. To specify the multiple case selection, you should perform the following steps:

- press the **Page User Mode** button on the toolbar,
- select the **Multiple Cases**... command from the **Data** menu,
- open the 'Cases' directory in your project directory and select 'Length 4' through 'Length 10' from the right list-box in the **Select Multiple Case Files** dialog box,
- press the **Add to Selection** button to transfer the selected cases to the right list-box (see Figure 14.10), and
- press the *OK* button.

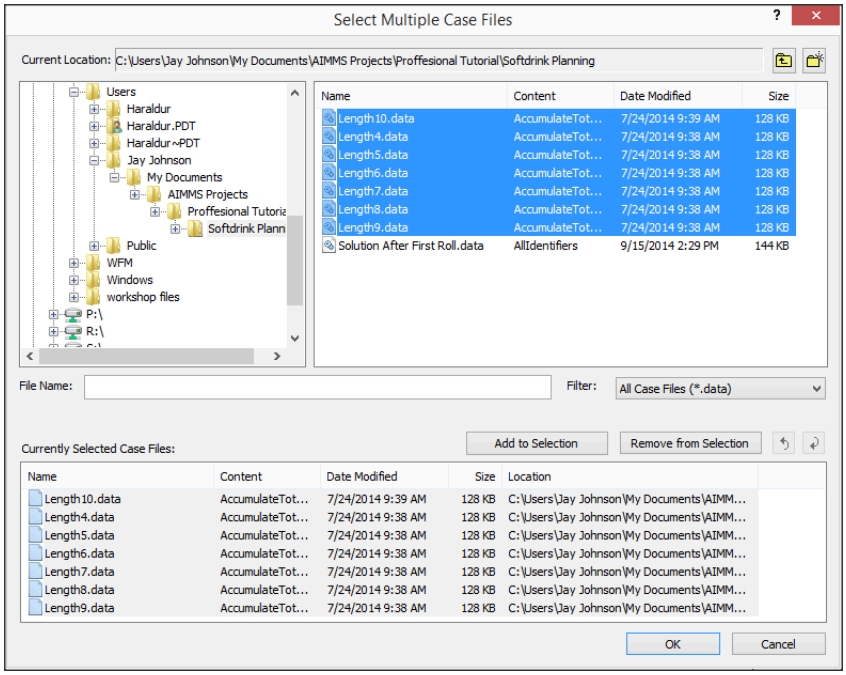


Figure 14.10: The **Select Multiple Cases** dialog box

Viewing the result

Having specified the multiple case selection, AIMMS will automatically load the required data from the cases and complete the table as in Figure 14.11.

	Length10	Length4	Length5	Length6	Length7	Length8	Length9
AccumulatedTotalCostInCurrentPeriod	1988511	2083987	0	0	0	2033192	1994707

Figure 14.11: A table displaying data for multiple cases

It is interesting to note that some entries in the table are left blank reflecting the fact that one of the subproblems in the "inner loop" of the experiment became insolvable. It is also interesting to note that the overall total cost does not decrease monotonically as the number of periods in the planning horizon increases. The experiment would seem to indicate that the number of periods should be greater than 10.

A-4 Available AIMMS Documents List

- AIMMS Getting Started
- AIMMS User's Guide
- AIMMS Language Reference
- AIMMS Function Reference
- AIMMS Optimization Modeling
- AIMMS Excel Add-in
- AIMMS Open Solver Interface
- AIMMS Tutorial For Beginners
- AIMMS Tutorial For Professionals