
AIMMS Tutorial for Professionals - Quantities and Time

This file contains only one chapter of the book. For a free download of the complete book in pdf format, please visit www.aimms.com

Copyright © 1993–2011 by Paragon Decision Technology B.V. All rights reserved.

Paragon Decision Technology B.V.	Paragon Decision Technology Inc.	Paragon Decision Technology Pte.
Schipholweg 1	500 108th Avenue NE	Ltd.
2034 LS Haarlem	Ste. # 1085	80 Raffles Place
The Netherlands	Bellevue, WA 98004	UOB Plaza 1, Level 36-01
Tel.: +31 23 5511512	USA	Singapore 048624
Fax: +31 23 5511517	Tel.: +1 425 458 4024	Tel.: +65 9640 4182
	Fax: +1 425 458 4025	

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

AIMMS is a registered trademark of Paragon Decision Technology B.V. IBM ILOG CPLEX and sc CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Ziena Optimization, Inc. XPRESS-MP is a registered trademark of FICO Fair Isaac Corporation. MOSEK is a registered trademark of Mosek ApS. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation. $\text{T}_{\text{E}}\text{X}$, $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$, and $\text{A}_{\text{M}}\text{S}_{\text{L}}\text{A}_{\text{T}}\text{E}_{\text{X}}$ are trademarks of the American Mathematical Society. LUCIDA is a registered trademark of Bigelow & Holmes Inc. ACROBAT is a registered trademark of Adobe Systems Inc. Other brands and their products are trademarks of their respective holders.

Information in this document is subject to change without notice and does not represent a commitment on the part of Paragon Decision Technology B.V. The software described in this document is furnished under a license agreement and may only be used and copied in accordance with the terms of the agreement. The documentation may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Paragon Decision Technology B.V.

Paragon Decision Technology B.V. makes no representation or warranty with respect to the adequacy of this documentation or the programs which it describes for any particular purpose or with respect to its adequacy to produce any particular result. In no event shall Paragon Decision Technology B.V., its employees, its contractors or the authors of this documentation be liable for special, direct, indirect or consequential damages, losses, costs, charges, claims, demands, or claims for lost profits, fees or expenses of any nature or kind.

In addition to the foregoing, users should recognize that all complex software systems and their documentation contain errors and omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors shall not be responsible under any circumstances for providing information or corrections to errors and omissions discovered at any time in this book or the software it describes, whether or not they are aware of the errors or omissions. The authors, Paragon Decision Technology B.V. and its employees, and its contractors do not recommend the use of the software described in this book for applications in which errors or omissions could threaten life, injury or significant loss.

This documentation was typeset by Paragon Decision Technology B.V. using $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ and the LUCIDA font family.

Chapter 6

Quantities and Time



6.1 Model Structure



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.

Initial tree

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 ?? and ??, and one model section for the user interface to be considered in Part ??. 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:

Creating two new sections ...

- ▶ 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.

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.

... and several subsections

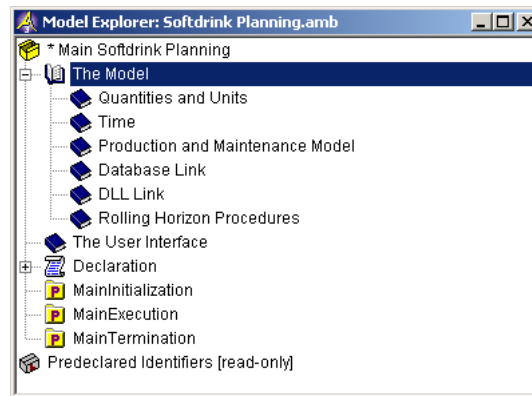




Figure 6.1: The structure of the section The Model

6.2 Entering quantity declarations

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:

Creating a declaration section

- ▶ 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.

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.




Units first

In Chapter ??, 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.

Volume quantity and its base unit

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

Declaring the volume quantity

- ▶ 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.

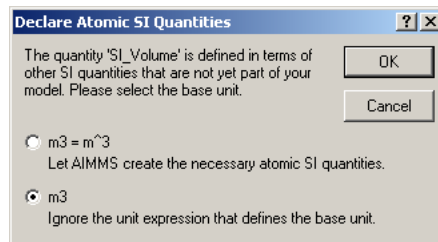


Figure 6.2: The **Ignore Unit Expression** dialog box

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.

Specifying its attributes

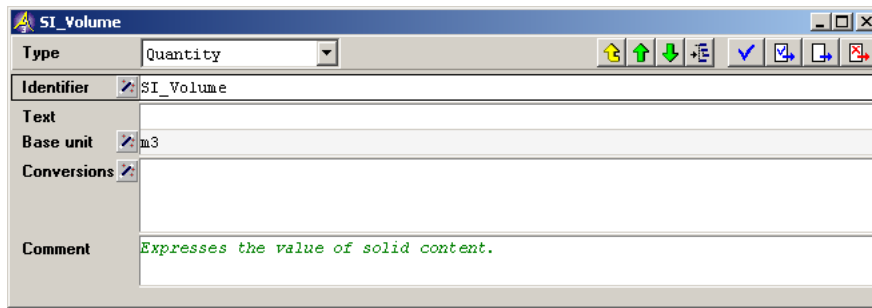




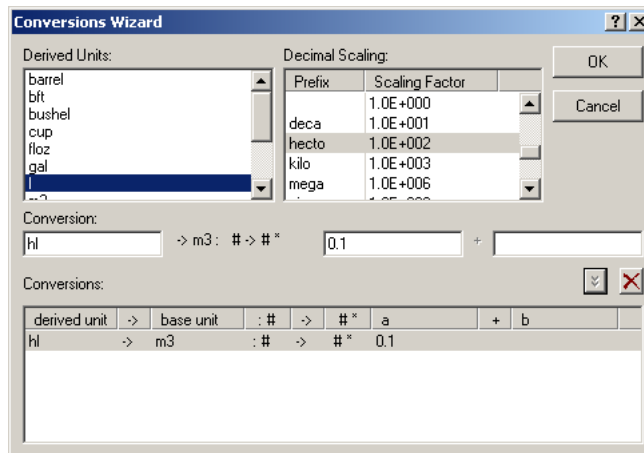
Figure 6.3: The initial attribute form of the quantity SI_Volume

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

Specifying the unit conversion of [hl]


- ▶ 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.

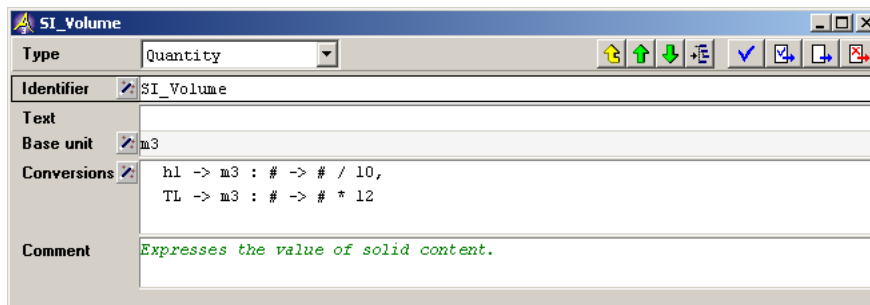
Figure 6.4: The selections in the **Conversions** Wizard

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:

Specifying the unit conversion of [TL]

- ▶ 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  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.




SI_Volume	
Type	Quantity
Identifier	SI_Volume
Text	
Base unit	m3
Conversions	hl -> m3 : # -> # / 10, TL -> m3 : # -> # * 12
Comment	Expresses the value of solid content.

Figure 6.5: The completed attribute form of the quantity SI_Volume

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:

Specifying the currency quantity

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

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'.

Specifying the time quantity

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.

Week-to-day conversion

SI_Time_Duration	
Type	Quantity
Identifier	SI_Time_Duration
Text	
Base unit	day
Conversions	week -> day : # -> # * 7
Comment	Expresses the value for the duration of periods.

Figure 6.6: The completed attribute form for the quantity SI.Time.Duration

The model tree so far is shown in Figure 6.7.

Your tree thus far

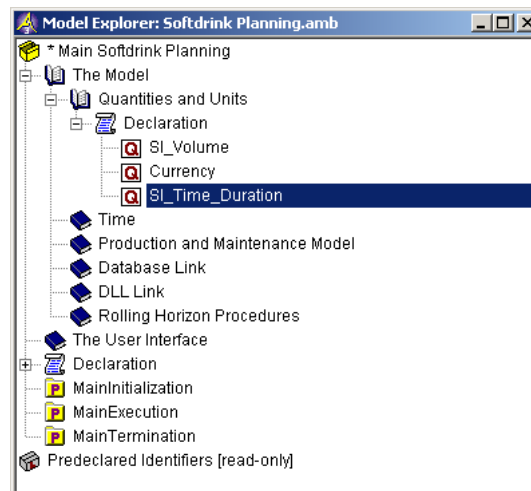



Figure 6.7: The intermediate model tree showing all quantity identifiers

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.

Saving your changes

6.3 Entering time declarations

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*.

Special data types

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.

Advanced concepts

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.

Calendars

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.

Horizons

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.

Timetables

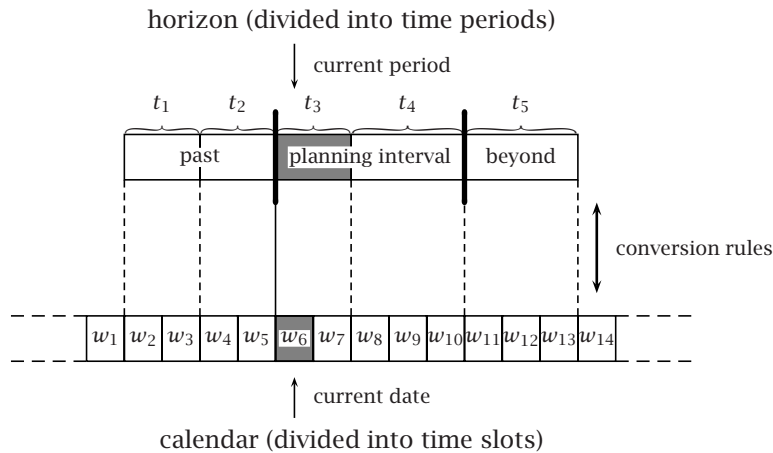


Figure 6.8: Linking a calendar and horizon

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

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:


Two declaration sections

- ▶ 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

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


Creating the first parameter NumberOfPeriods

- ▶ 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.

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

Parameter attributes

- ▶ 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.

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`.



Range checking

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.

Creating the second parameter

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

Declaring a horizon

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

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

Entering the first 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.

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

*Specifying the
Interval length
attribute*

- ▶ 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*.

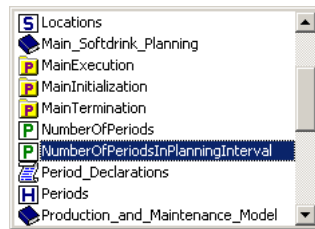



Figure 6.10: The **Name Completion** pop-up menu

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`.

*Specifying the
Definition
attribute*

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

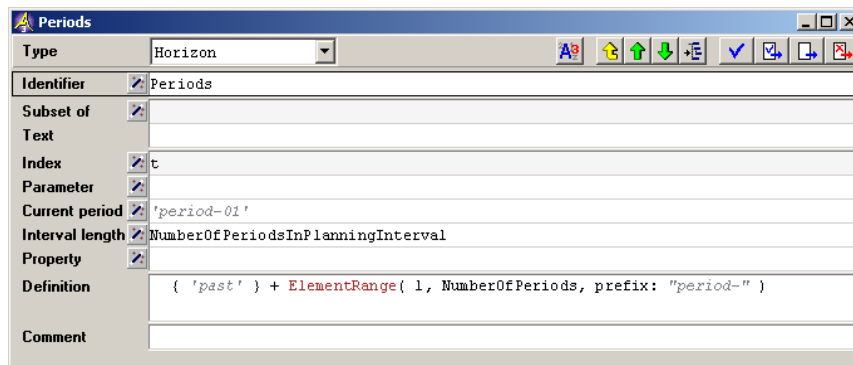


Figure 6.11: The completed attributes of the horizon 'Periods'

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.

Name completion and dragging

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.

The function ElementRange

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).

... its arguments

```

ElementRange (
ElementRange(
  [Input] From AS Parameter,
  [Input] To AS Parameter,
  [Optional] Incr = 1 AS Parameter,
  [Optional] Prefix = "" AS String Parameter,
  [Optional] Postfix = "" AS String Parameter,
  [Optional] Fill = 1 AS Parameter)

```

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

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:

... and its documentation

- ▶ 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.

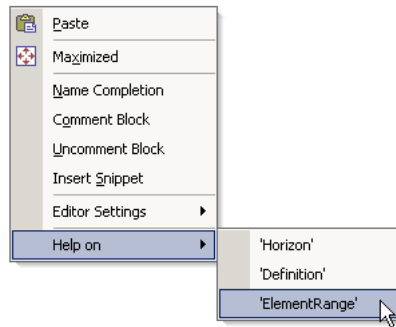



Figure 6.13: A right-mouse pop-up menu

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 week-ends 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.

*Remaining
period
declarations*



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:


*Number of days
in a period*

- ▶ 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.

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:

*Number of
weeks in period*

- ▶ 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.

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

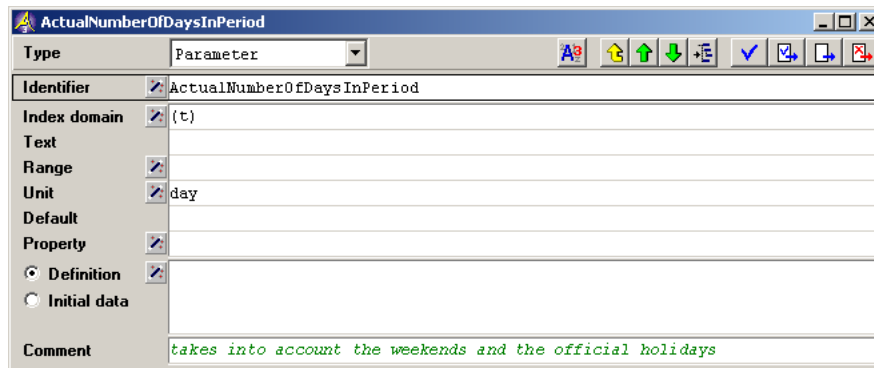
Name change propagation

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

Declaring the actual period length

- ▶ 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 'SLTime.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.





ActualNumberOfDaysInPeriod	
Type	Parameter
Identifier	ActualNumberOfDaysInPeriod
Index domain	(t)
Text	
Range	
Unit	day
Default	
Property	
<input checked="" type="radio"/> Definition <input type="radio"/> Initial data	
Comment	takes into account the weekends and the official holidays


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

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:

Declaring a parameter for the first period
...

- ▶ 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.

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.

... and completing its attributes

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

The model tree

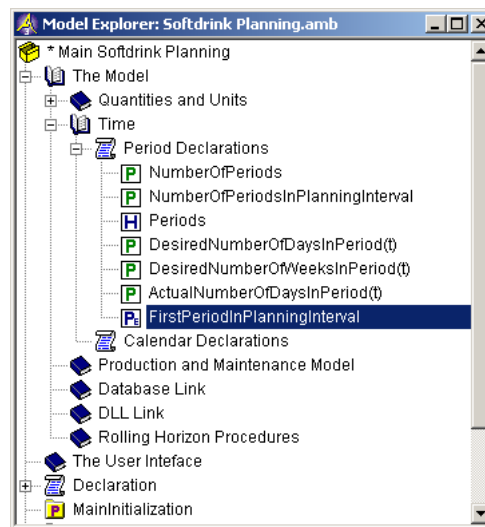




Figure 6.15: All period declarations in the model tree

6.3.2 Calendar-related declarations


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:

Declaring begin and end dates

- ▶ 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.



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.

... completing their attributes

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.

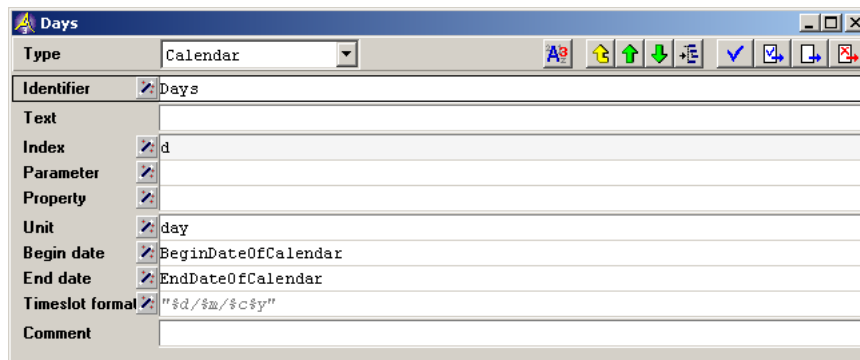
To declare the calendar Days, execute the following steps:

- ▶ 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.

Declaring a calendar

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.

Specifying the calendar attributes



Attribute	Value
Type	Calendar
Identifier	Days
Text	
Index	d
Parameter	
Property	
Unit	day
Begin date	BeginDateOfCalendar
End date	EndDateOfCalendar
Timeslot format	"%d/%m/%C%y"
Comment	

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

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.

Timeslot format

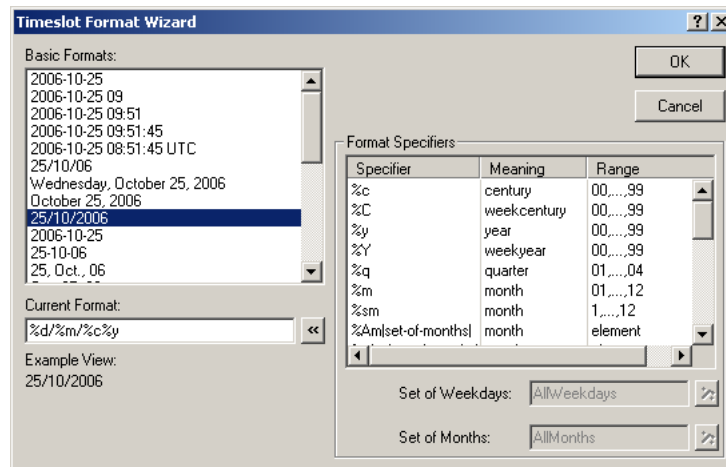


Figure 6.17: The **Timeslot Format** wizard

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.

Declaring subsets ...

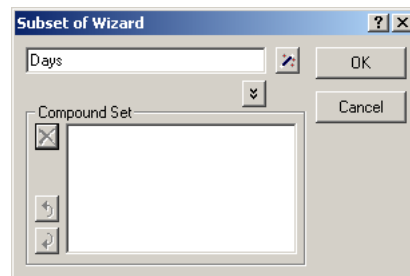


Figure 6.18: The **Subset of** wizard

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:

*... of the
calendar Days*

```

SET:
  identifier      : WeekendDays
  subset of      : Days
  definition      : { d | TimeslotCharacteristic(d,'weekday') > 5 }

SET:
  identifier      : OfficialHolidays
  subset of      : Days

SET:
  identifier      : InactiveDays
  subset of      : Days
  definition      : WeekendDays + OfficialHolidays

SET:
  identifier      : Mondays
  subset of      : Days
  definition      : { d | TimeslotCharacteristic(d,'weekday') = 1 }

SET:
  identifier      : DaysInPeriod
  index domain   : t
  subset of      : Days

```

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.

*Timeslot
characteristics*

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:

*Viewing the
weekend days*

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

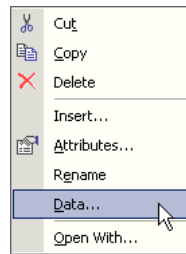


Figure 6.19: A right-mouse pop-up menu

AIMMS will now display the corresponding data page as shown in Figure 6.20. *Data page*
 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.

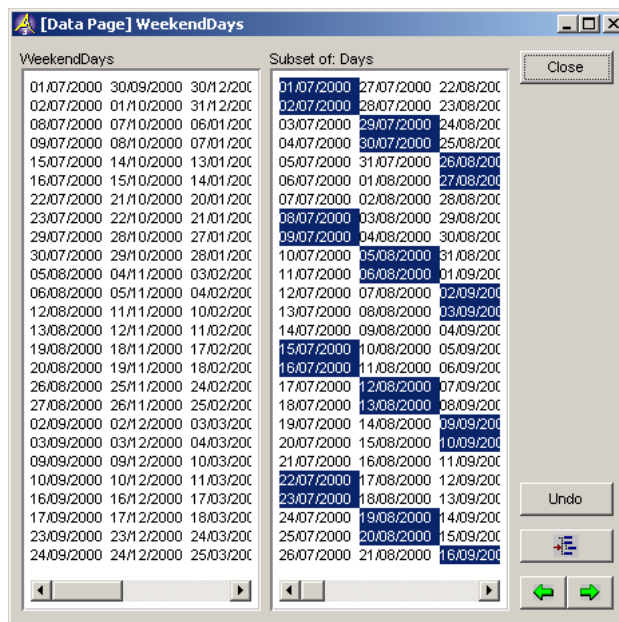


Figure 6.20: Data page for the set WeekendDays

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.

Creating a weekly calendar

Weeks	
Type	Calendar
Identifier	Weeks
Text	
Index	w
Parameter	
Property	
Unit	7 * day
Begin date	BeginDateOfCalendar
End date	EndDateOfCalendar
Timeslot format	"week %sW, %C%Y"
Comment	

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.

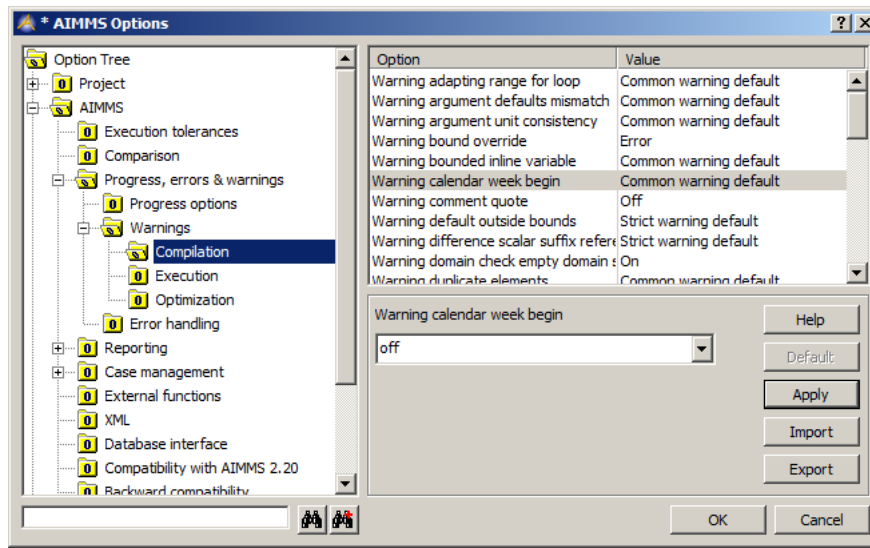


Figure 6.22: The AIMMS Options dialog box

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.

Declaring week-related references

```
SET:
  identifier      : InactiveWeeks
  subset of      : Weeks

ELEMENT PARAMETER:
  identifier      : WeekInPeriod
  index domain    : t
  range          : 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:

Relating days to weeks

```
ELEMENT PARAMETER:
  identifier      : DayToWeek
  index domain    : 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.

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

One more identifier

```
ELEMENT PARAMETER:
  identifier   : LastWeekInCalendar
  range       : Weeks
  definition   : last(Weeks)
```

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

Model tree

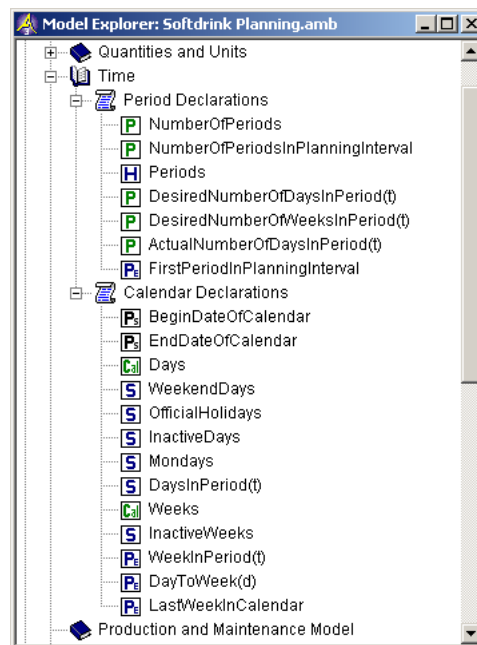


Figure 6.23: All calendar related declarations in the model tree