
AIMMS User's Guide - Collaborative Project Development

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 © 1993–2018 by AIMMS B.V. All rights reserved.

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

AIMMS is a registered trademark of AIMMS B.V. IBM ILOG CPLEX and CPLEX is a registered trademark of IBM Corporation. GUROBI is a registered trademark of Gurobi Optimization, Inc. KNITRO is a registered trademark of Artelys. WINDOWS and EXCEL are registered trademarks of Microsoft Corporation. \TeX , \LaTeX , and $\AMS-\LaTeX$ 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 AIMMS 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 AIMMS B.V.

AIMMS 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 AIMMS 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, AIMMS 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, AIMMS 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 AIMMS B.V. using \LaTeX and the LUCIDA font family.

Chapter 3

Collaborative Project Development

This chapter discusses the options you have in AIMMS to organize a project in such a way that it becomes possible to effectively work on the project with multiple developers. While it is very natural to start working on a project with a single developer, at some time during the development of an AIMMS application, the operational requirements of the problem you are modeling may become so demanding that it requires multiple developers to accomplish the task.

This chapter

During the initial prototyping phase of a model, an AIMMS project is usually still quite small, allowing a single developer to take care of the complete development of the prototype. The productivity tools of AIMMS allow you to quickly implement different formulations and analyze their results, while you are avoiding the overhead of having to synchronize the efforts of multiple people working on the prototype.

*From proto-
typing phase...*

During the development of an operational AIMMS application this situation may change drastically. When an AIMMS application is intended to be used on a daily basis, it usually involves one or more of the following tasks:

*... to operat-
ional phase*

- retrieving the input data from one or more data sources,
- validation and transformation of input data,
- extending the core optimization application with various, computationally possibly demanding, operational requirements,
- preparing and writing output data to one or more data sources,
- building a professionally looking end-user GUI, and/or
- integrating the application into the existing business environment.

Depending on the size of the application, implementing all of these tasks may become too demanding for a single developer.

One possible approach to allow multiple developers to work on a single AIMMS application is to divide the project into several logical sub-projects, either based on the tasks listed in the previous paragraph, or more closely related to the logic of your application. AIMMS supports sub-projects in the form of model libraries. Using libraries especially makes sense, if the functionality

*Dividing a
project into
sub-projects*

developed in a library can be re-used by multiple AIMMS applications. If a library is small enough, individual developers may take on the development of the library.

In the software development world teams commonly use a Version Control System, such as `git`, `subversion`, or `TFS`, to share and merge their coding work to a common repository. As all development sources of an AIMMS application are stored as readable text files (a.o. `.aimms`, `.ams` and `.xml`), AIMMS projects can be easily managed using the version control system of your choice. Using a version control system will make it straightforward to work together on a single code base in parallel by merging the contributions of the various team members, and to use branches to differentiate between development and production code, or to work on multiple developments independently. Using version control for your AIMMS projects will usually result in higher productivity and more control.

Managing project source using a VCS

Although AIMMS effectively supports the use of a version control system for the development of your AIMMS applications, the AIMMS IDE does not offer integration with any specific version control system. All version control systems come with commandline and/or graphical tools for regular version control tasks such as committing, showing logs, diffing two versions, merging, creating branches and tags, and so on. You should use these tools, whenever you want to commit your changes to an AIMMS project under version control.

No VCS integration

3.1 Library projects and the library manager

AIMMS *library projects* allow you to divide a large AIMMS project into a number of smaller sub-projects. Library projects are especially useful if you intend to share parts of an application between multiple projects. Each library project in AIMMS provides

AIMMS library projects

- a tree of model declarations,
- a page tree,
- a template tree, and
- a menu tree.

In addition, a library project may provide its own collection of user project files, user colors and fonts.

Besides enabling multiple developers to work in a single project, library projects can also be used to define a common collection of templates that define the look-and-feel of multiple projects. In this manner, you change the look-and-feel of multiple applications just by changing the templates in the shared library project.

Shared templates

By adding a library project to the main AIMMS project, the objects defined by the library, such as identifiers, pages, templates, etc., become available for use in the main project. In addition, if a library project is writable, the contents of the library can also be edited through an AIMMS project in which it is included.

Adding libraries to a project

You can add libraries to an AIMMS project in the **AIMMS Library Manager** dialog box illustrated in Figure 3.1. You can open the library manager through the **File-Library Manager...** menu.

The library manager

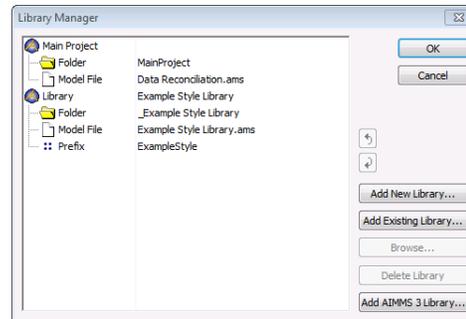


Figure 3.1: The **AIMMS Library Manager**

Using the library manager AIMMS allows you to

- create new libraries,
- add existing libraries to your project,
- edit library properties, and
- remove libraries from your project.

Each library project in AIMMS will be stored in a separate directory, containing the following files and folders:

Library storage

- the `Project.xml` file holding a reference to the project's main model source file (with an `.ams` extension), as well as all additional model source files included in the main model source file, together containing all identifier declarations, procedures and functions that are relevant to the project,
- `PageManager.xml`, `TemplateManager.xml` and `MenuBuilder.xml` files describing the page, template and menu tree defined in the project, with all individual pages and templates being stored in the Pages and Templates subfolders,
- Settings and Tools subfolders containing the saved settings for user colors, fonts, and the various tools in the AIMMS IDE, and
- the User Files folder for storing all user files that you store within the project.

These files will be automatically created by AIMMS when you create a new library project. To add an existing library to an AIMMS project, you just need to select its library project file.

To avoid name clashes between objects in the library and the main project or other libraries, all the object names in a library are stored in a separate namespace. Outside of the library, a global prefix associated with the library has to be used to access the library objects. When you create a new library project, AIMMS will come up with a default library prefix based on the library name you entered. For an existing library project, you can view and edit its associated library prefix in the library manager.

Library prefix

After you have added one or more library projects to your main AIMMS project, AIMMS will extend

Using library projects

- the model tree in the **Model Explorer**,
- the page tree in the **Page Manager**,
- the template tree in the **Template Manager**, and
- the menu tree in the **Menu Builder**

with additional root nodes for every library project added to your project. In general, within any of these tools, you are free to move information from the main project tree to any of the library trees and vice versa. In addition, the AIMMS dialog boxes for user project files, user colors and fonts allow you to select and manage objects from the main project or any of the libraries. The precise details for working with library projects in each of these tools are discussed in full detail in the respective chapters discussing each of the tools.

3.2 Guidelines for working with library projects

Unless you started using library projects from scratch, you need to convert the contents of your AIMMS project as soon as you decide to divide the project into multiple library projects. The first step in this process is to decide which logical tasks in your application you can discern that meet the following criteria:

Identifying independent tasks

- the task represents a logical unit within your application that is rather self-contained and can possibly be shared with other AIMMS projects,
- the task can be comfortably and independently worked on by a separate developer or developer team, and
- the task provides a limited interface to the main application and/or the other tasks you have identified.

Good examples of generic tasks that meet these criteria are the tasks listed on page 33. Once your team of developers agrees on the specific tasks that are relevant for your application, you can set up a library project for each of them.

The idea behind library projects is to be able to minimize the interaction between the library, the main project and other library projects. At the language level AIMMS supports this by letting you define an *interface* to the library, i.e. the set of public identifiers and procedures through which the outside world is allowed to connect to the library. Library identifiers not in the interface are strictly private and cannot be referenced outside of the library. The precise semantics of the interface of a library module is discussed in Section 35.5 of the Language Reference.

*Library
interface...*

This notion of public and private identifiers of a library module does not only apply to the model source itself, but also propagates to the end-user interface. Pages defined in a library can access the library's private identifiers, while pages defined outside of the library only have access to identifiers in the interface of the library.

*... used in
model and GUI*

The concept of an interface allows you to work independently on a library. As long as you do not change the declaration of the identifiers and procedures in its interface, you have complete freedom to change their implementation without disturbing any other project that uses identifiers from the library interface. Similarly, as long as a page or a tree of pages defined in a library project is internally consistent, any other project can add a reference to such pages in its own page tree. Pages outside of the library can only refer to identifiers in the library interface, and hence are not influenced by changes you make to the library's internal implementation.

*Minimal
dependency*

If your application already contains model source and pages associated with the tasks you have identified in the previous step, the next step is to move the relevant parts of your AIMMS project to the appropriate libraries. You can accomplish this by simply dragging the relevant nodes or subtrees from any of the trees in the main project to associate tree in a library project. What should remain in the global project are the those parts of the application that define the overall behavior of your application and that glue together the functionality provided by the separate library projects.

*Conversion to
library projects*