Guide for evaluators

9.9

QUICK REFERENCE

FICO® Xpress Optimization

FICO

©2006–2026 Fair Isaac Corporation. All rights reserved. This documentation is the property of Fair Isaac Corporation ("FICO"). Receipt or possession of this documentation does not convey rights to disclose, reproduce, make derivative works, use, or allow others to use it except solely for internal evaluation purposes to determine whether to purchase a license to the software described in this documentation, or as otherwise set forth in a written software license agreement between you and FICO (or a FICO affiliate). Use of this documentation and the software described in it must conform strictly to the foregoing permitted uses, and no other use is permitted.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, please report them to us in writing. Neither FICO nor its affiliates warrant that this documentation is error-free, nor are there any other warranties with respect to the documentation except as may be provided in the license agreement. FICO and its affiliates specifically disclaim any warranties, express or implied, including, but not limited to, non-infringement, merchantability and fitness for a particular purpose. Portions of this documentation and the software described in it may contain copyright of various authors and may be licensed under certain third-party licenses identified in the software, documentation, or both.

In no event shall FICO or its affiliates be liable to any person for direct, indirect, special, incidental, or consequential damages, including lost profits, arising out of the use of this documentation or the software described in it, even if FICO or its affiliates have been advised of the possibility of such damage. FICO and its affiliates have no obligation to provide maintenance, support, updates, enhancements, or modifications except as required to licensed users under a license agreement.

FICO is a registered trademark of Fair Isaac Corporation in the United States and may be a registered trademark of Fair Isaac Corporation in other countries. Other product and company names herein may be trademarks of their respective owners.

Patent(s): www.fico.com/en/patents

FICO® Xpress Optimization 9.9

Last Revised: 15 May, 2026

How to Contact the Xpress Team

Sales and Maintenance

If you need information on other Xpress Optimization products, or you need to discuss maintenance contracts or other sales-related items, contact FICO by:

- Phone: +1 (408) 535-1500 or +44 207 940 8718
- Web: www.fico.com/optimization and use the available contact forms

Product Support

Customer Self Service Portal (online support): www.fico.com/en/product-support

Email: Support@fico.com (Please include *'Xpress'* in the subject line)

For the latest news and Xpress software and documentation updates, please visit the Xpress website at http://www.fico.com/xpress or subscribe to our mailing list.

FICO® Xpress Optimization

Guide for evaluators

Release 9.9

15 May, 2026

Introduction

Welcome to FICO® Xpress Optimization! This guide provides a framework for evaluating Xpress. Using this guide you will be able to:

1. Verify that the Xpress installation was successful.
2. Decide which Xpress products to evaluate.
3. Evaluate Xpress.

1 Verify that the Xpress installation was successful

You can test that the Xpress installation was successful by launching Xpress Workbench (on Windows and macOS operating systems only) or by running console commands (all operating systems).

1. *Launch Xpress Workbench:* Xpress Workbench is the development environment for Xpress Mosel and Xpress Insight. To run Xpress Workbench, double click on the Xpress Workbench icon on your desktop:  or select *FICO* ≫ *Xpress* ≫ *Xpress Workbench* from the Windows start menu. You may also start up Xpress Workbench by typing `xpworkbench` at the command prompt or by double clicking onto an Xpress Mosel model file (file with extension `.mos`).
2. *Run console commands:* At the command prompt[1], type the following sequence of commands:

   ```
   optimizer
   ↵ (Enter)
   quit
   ```

   You will see output that looks like the following:

```
C:\> optimizer
FICO Xpress Solver 64bit v9.1.0 Apr 14 2023
(c) Copyright Fair Isaac Corporation 1983-2023. All rights reserved
 Optimizer v41.01.07   [C:\xpressmp\bin\xprs.dll]
Enter problem name >
[xpress C:\] quit
```

[1] To obtain a command prompt under Windows select *Start* ≫ *Programs* ≫ *Accessories* ≫ *Command Prompt*

2 Decide which Xpress products to evaluate

In order to make this decision you must consider the type of problem that you wish to solve, the tools that you wish to use for model development, the tools that you wish to use for model deployment, and the platform you intend to use.

2.1 Type of problem

Xpress has solver engines for different types of problems:

Type of problem	Solving engine
Linear Programming (LP)	Xpress Optimizer
Mixed Integer Programming (MIP)	
Quadratic Programming (QP)	
Mixed Integer Quadratic Programming (MIQP)	
Quadratically Constrained Quadratic Programming (QCQP/MIQCQP)	
Second Order Cone Programming (SOCP/MISOCP)	
Non-Linear Programming (NLP)	Xpress NonLinear, Xpress Global
Mixed Integer Non-Linear Programming (MINLP)	
Constraint Programming (CP)	Xpress Kalis

Further information

See Xpress documentation (in the `docs` subdirectory of your Xpress installation or at https://www.fico.com/fico-xpress-optimization/docs/latest):

- Classification of mathematical programming problems: “Getting Started with Xpress”, 1.1. ‘Mathematical Programming’.
- Modeling with Mosel, in particular LP/MIP: Xpress Mosel User Guide.
- Formulation of mathematical programming problems: see the book: “Applications of Optimization with Xpress-MP”, Part I: ‘Developing Linear and Integer Programming models’.
- Non-linear Programming: Xpress NonLinear Manual, Xpress Global User Manual.
- Constraint Programming: Xpress Kalis Mosel User Guide.

2.2 Tools for model development and specification

Xpress provides the following tools for model development and specification:

- **Xpress Mosel/Xpress Workbench:** Mosel is an advanced modeling and programming language. Xpress Workbench is the visual environment to develop Mosel models and Insight apps.
- **Xpress Solver object-oriented APIs:** Object-oriented library callable from supported programming languages.
- **Xpress Optimizer libraries:** Optimizer functions and procedures provided for low-level integration with applications developed in supported programming languages.

Also, a model may be specified by a matrix in MPS or LP format.

2.3 Tools for model deployment

Model deployment strongly depends on the tool used to specify the model. Deployment is achieved through the use of Xpress libraries. Alternative deployment options for Mosel models are distributed or Cloud computing applications controled via XPRD (see the *Advanced Evaluators Guide* for further detail) and the embedding into distributed, multi-user applications using Xpress Insight.

Model specified in	Deployed using
Mosel	Xpress Insight
	Mosel libraries (C++/C, Java, VBA, .NET)
	XPRD (C++/C, Java)
Object-oriented Solver libraries	Object-oriented Solver libraries (C++, Java, .NET)
Optimizer libraries	Optimizer libraries (C++/C, Java, .NET, Python, R)
Kalis libraries	Kalis libraries (C++, Java, Python)

Note that for Constraint Programming (Xpress Kalis), the model must be specified either via Xpress Mosel or using the Kalis libraries.

Further information

See Xpress documentation (in the `docs` subdirectory of your Xpress installation or at http://www.fico.com/fico-xpress-optimization/docs/latest):

- Introductory examples (Mosel/Optimizer): Getting Started with Xpress.
- Mosel libraries examples: "Xpress Mosel User Guide", Part III: 'Working with the Mosel Libraries'.
- Mosel libraries documentation: Mosel Library Reference Manual.
- Object-oriented APIs: Solver C++ User Guide, Solver Java User Guide and Solver .NET User Guide.
- Optimizer: Optimizer Reference Manual.
- Kalis: Xpress Kalis Mosel Reference Manual and Xpress Kalis Libraries User Guide.

2.4 Platform

Xpress Workbench is available on Microsoft Windows platforms, on macOS, and on the FICO Analytic Cloud. Xpress Kalis is available on Microsoft Windows, macOS and Linux. All other products are available on every supported platform.

3 Evaluate Xpress

The following five typical evaluator scenarios are defined depending on the choice of products to evaluate (see question 2):

- **Scenario 1:** Develop the model in Mosel for any kind of problem and deploy in any programming language or as an Xpress Insight application.

Tool for model development:	Mosel
Type of problem:	Any
Tool for model deployment:	(a) Xpress Insight (b) Mosel Libraries (C++/C, Java, VBA, .NET)

- **Scenario 2:** Implement and deploy the model in a programming language using the object-oriented API for Xpress solvers.

Tool for model development:	Object-oriented solver API
Type of problem:	LP, MIP, QP, MIQP, NLP, MNLP
Tool for model deployment:	Object-oriented solver API (C++, Java, .NET)

- **Scenario 3:** Use your custom application to develop the model and then call the Xpress Optimizer libraries. Available for problems that can be solved with Xpress Optimizer.

Tool for model development:	Custom application that calls optimizer libraries.
Type of problem:	LP, MIP, QP, MIQP
Tool for model deployment:	Optimizer libraries (C++/C, Java, .NET, Python, R)

- **Scenario 4:** Run a matrix that is readily available in LP or MPS format. Such a model may be executed through Xpress Workbench, console commands, or applications that call Xpress Optimizer library functions.
- **Scenario 5:** Use Xpress Insight to view the app demos and examples provided with the Xpress installation.

Each scenario defines a sequence of specific evaluation steps.

3.1 Evaluation steps for Scenario 1

Tool for model development:	Mosel
Type of problem:	Any
Tool for model deployment:	(a) Xpress Insight (b) Mosel Libraries (C++/C, Java, VBA, .NET)

This scenario uses examples from the “Getting Started with Xpress” document that can be found in directory `xpressmp\examples\getting_started\Mosel` (where `xpressmp` is the installation directory of Xpress).

3.1.1 Launch Xpress Workbench

Xpress Workbench is the visual development environment for Windows. To run Xpress Workbench, double click on the Xpress Workbench icon on your desktop, or select *FICO* ≫ *Xpress* ≫ *Xpress Workbench* from the Windows sart menu. Otherwise, you may also start up Workbench by typing `xpworkbench` in a DOS window or by double clicking on a model file (file with extension `.mos`).

3.1.2 Open Mosel model in Xpress Workbench

Locate the directory containing the evaluation examples in your Xpress installation (`xpressmp\examples\getting_started\Mosel`) and open the file `foliolp.mos` by double clicking on its name in the file browser. Alternatively, if you have already started Xpress Workbench select *Open a file* in the Xpress Workbench entry screen and browse to select the file `foliolp.mos`. The model "Portfolio optimization with LP" will open in the central pane (the Workbench editor). This model seeks an optimal investment portfolio using ten securities (shares), subject to some risk and regional constraints. The comments in the code (text following the exclamation mark '!') describe the meaning of the different statements. Note that `RET` is an array of real values representing the expected return of the shares. The decision variables in the model are given by the array `frac` of type `mpvar`. The procedure `maximize` calls Xpress Optimizer to maximize the objective function. The code also contains statements to print the optimal solution and solution values.

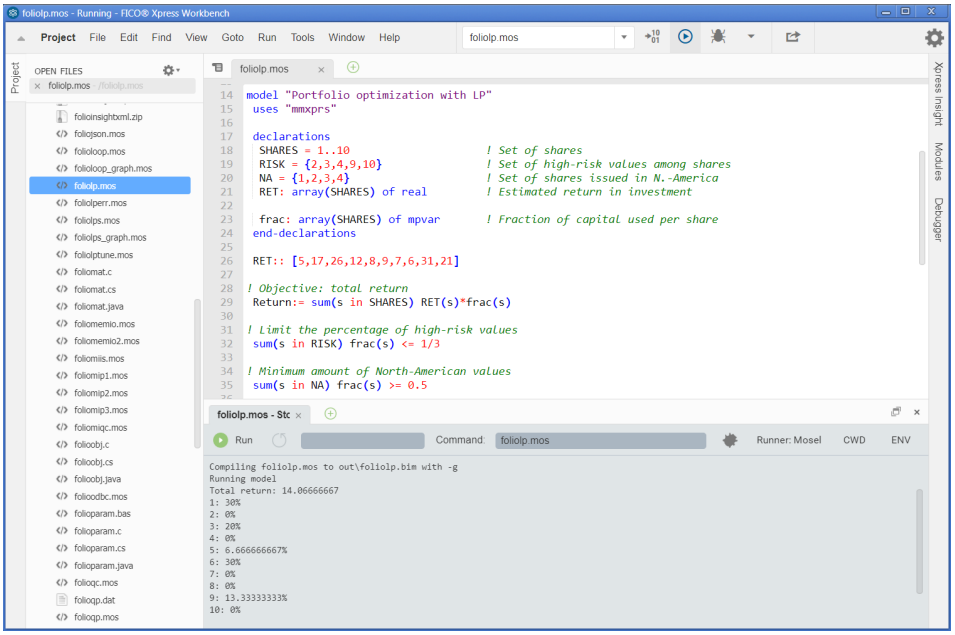


Figure 1: Workbench window after a model run

Further information

- For more information on the model formulation for this example see: "Getting Started with Xpress", Chapter 2: 'Building models'.
- For more information on the Mosel representation of the model for this example see: "Getting Started with Xpress", 3.2 'LP model'.
- For more information on Mosel and other Mosel examples see "Xpress Mosel User Guide", Chapter 1: 'Getting started with Mosel', 1.1 'Entering a model'.

- New users of Mosel may also wish to take a look at the Appendix 'Good modeling practice with Mosel' of the "Xpress Mosel User Guide".

3.1.3 Compile and run the Mosel model

To execute a Mosel model select menu *Run* or click on the green 'Run' icon next to the dropdown file selection box at the top (make sure the desired filename displays in the box). The output log and the status of the model execution are shown at the bottom of the Xpress Workbench screen (see Figure 1), and it should read *Mosel exited with code 0* and *Process exited with code: 0* at the end of the display. If Mosel detects any errors during compilation or model execution these will equally be reported in this logging window.

While developing a model you may wish to just compile a model without running it with a data instance, *e.g.* to check for syntax errors. Select menu entry *Run* ≫ *Build* to compile the selected model. Upon successful compilation you will see the message *Created foliolp.bim* in the logging window at the bottom and the compiled file appears in the workspace file listing, otherwise the log reports any errors that have occurred.

Now select the *Debug* icon to run the model once more and open the *Debugger* pane at the right. The model execution will be suspended just before its termination, allowing you to inspect the values of the model objects that have been populated by the model execution (see Figure 3). In debug mode you can also set breakpoints or execute the Mosel model step-by-step to analyse its behaviour.

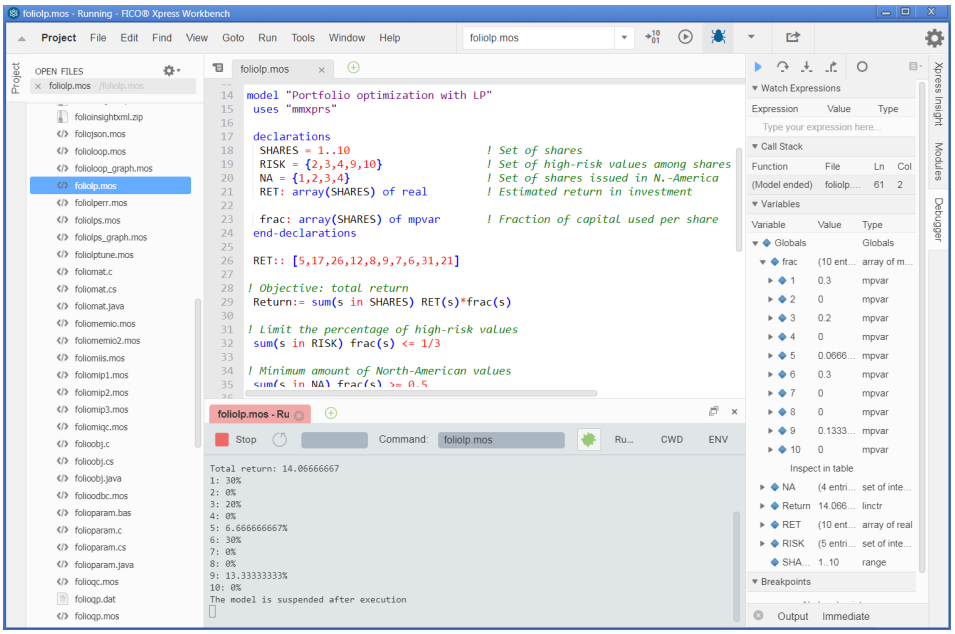


Figure 2: Workbench window with debugger display

Further information

- For more on compilation and compilation errors see: "Getting Started with Xpress", 3.3. 'Correcting errors and debugging a model'.
- For more on Xpress Workbench see the Xpress Workbench User Guide.

String indices: The model data and solution are more easily understandable when using string indices.

Open and run model `foliolps.mos`. Explore the model output and the entities in the debugger display.

Further information

- For more information on running the Mosel model for this example see: "Getting Started with Xpress", 3.4. 'Solving and viewing the solution'.
- For more information on string indices for this example see: "Getting Started with Xpress", 3.4.1. 'String indices', and also "Xpress Mosel User Guide", 2.1.3. 'The burglar problem revisited'.

3.1.4 Work with data in Mosel

In Mosel models you may work with a large variety of data sources, ranging from simple text files and other file formats such as databases to data exchanged in memory between a Mosel model and a host application or between several concurrent Mosel models. We show here the most frequent cases, namely text files, spreadsheets, and database access via ODBC.

Text files

Open and run model `foliodata.mos`. Note that this model has a `parameters` block and an `initializations from` block. The parameters include the data input file, output file, and other model constants. Parameters can be reset at run time, and they are particularly important when a model is deployed within a business application. The `initializations from` block reads data from the file `folio.dat`. This file has a Mosel-specific format that can be seen by opening it in Xpress Workbench by selecting menu *File* ≫ *Open...* or double click on the filename in the directory contents listing to the left of the main editor window. The index sets and data arrays in the model are created dynamically based on the data in the input file. Finally this model directs the solution output to an external free-format file `result.dat` by calling the procedures `fopen` and `fclose`.

Further information

For more information on working with data and using parameters in Mosel see: "Getting Started with Xpress", Chapter 4: 'Working with data'.

Spreadsheets and databases

Open and run model `folioexcel.mos`. The `initializations from` block in this program reads data from an MS Excel spreadsheet. The model is accompanied by data in the file `folio.xls`. Note that this example changes the sets `RISK` and `NA` into arrays of Booleans to receive the data from the file. The `initializations to` block outputs the problem results back to the spreadsheet. If the Excel file is open when writing to it the output data does not get saved, letting you choose whether to keep the results or not. Repeated model executions will overwrite previous output in the target range.

A second very similar model, `folioodbc.mos`, reads data from an ODBC data source (*e.g.*, the MS Access database `folio.mdb`) and outputs the problem results back to the database. The prefix to the filename in the `initializations` blocks now is `mmodbc.odbc`, corresponding to the type of data we work with; all else is the same as in the Excel model. The ODBC database access facility can also be employed with MS Excel spreadsheets. However, some restrictions apply and we recommend to use the Excel-specific data access as shown in the model `folioexcel.mos`.
Warning: In order to run this example, the ODBC driver for the corresponding data source must be present.

As an alternative to the Excel-specific access to spreadsheets shown in the example file

`folioexcel.mos`, Mosel also provides generic interfaces to XLS, XLSX, and CSV format files that are usable including on non-Windows platforms. The model file `foliosheet.mos` uses the generic spreadsheet interface to access data in the file `folio.xls` (with this interface, the output file needs to be closed when writing to it from Mosel). And the example `foliocsv.mos` works with CSV format data held in the file `folio.csv`.

Further information

- Using the ODBC module: "Xpress Mosel User Guide", Chapter 2: 'Some illustrative examples' and the whitepaper Using ODBC and other database interfaces with Mosel.
- Documentation of the ODBC and spreadsheet interfaces: "Mosel Language Reference Manual", Chapters: '*mmodbc*' and '*mmsheet*'.
- Overview of other possibilities of data exchange with external sources: "Xpress Mosel User Guide", 16.1 'Generalized file handling'.
- Examples of advanced communication methods with external data sources are given in the whitepaper Generalized file handling in Mosel.

3.1.5 Mosel MIP and quadratic models

Open and run the following Mosel models:

`foliomip1.mos`: This model introduces the array of binary variables **buy** to impose a constraint limiting the number of different shares taken into the portfolio.

`foliomip2.mos`: This model redefines the array of variables **frac** to be semi-continuous, so that at least a certain minimum amount of the budget is spent on each share that is bought.

`folioqp.mos`: This model uses a quadratic formulation to minimize the portfolio variance subject to achieving a target expected return. The Mosel program solves the problem twice, where the second run imposes a limit on the number of shares taken into the portfolio.

For each model, explore the solution and information displayed in the Run Bar tabs.

Further information

- "Getting Started with Xpress", Chapter 6: 'Mixed Integer Programming' and Chapter 7: 'Quadratic Programming'.
- Complete list of available MIP variable types: "Xpress Mosel User Guide", Chapter 4: 'Integer Programming'.

3.1.6 Other problem types: Constraint Programming, Nonlinear Programming

All models we have seen so far use Xpress Optimizer for problem solving (chosen with the statement `uses "mmxprs"` at the begin of the model). If we wish to use a different solver, we need to indicate the name of the corresponding solver module.

Open the model `assign.mos`: this model implements and solves an assignment problem with Xpress Kalis, that is, using Constraint Programming (CP) techniques. For given sets of workers and machines the problem is to assign exactly one worker to every machine, maximizing the total productivity. The productivity of a worker depends on the machine he is assigned to.

You may observe several differences to the models we have seen previously:

– The solver choice statement now is `uses "kalis"`.
– The CP decision variables are of the type `cpvar`; their domain (=admissible values) can be set with the procedure `setdomain`.
– CP models may have linear constraints (as in the 'Total productivity' constraint), however our model also uses other types of constraint relations, so-called 'global constraints'. The `element` constraint formulates a discrete function in one variable, and the `all_different` relation states that all variables in the constraint need to take a different value.
– The CP problem is solved with tree search methods. Instead of using the default search strategies, it is usually preferable to choose a more problem-specific strategy (using procedure `cp_set_branching`).
– The function `cp_maximize` is used to invoke the optimization.
All else (general structure, declarations, access to data, output printing) remains unchanged from what we have seen so far.

When running this model with Workbench the model output appears in the *Output* pane at the bottom of the workspace as with Mathematical optimization problems and in debug mode the model entity display in the *Debugger* tab is populated.

Other solver types available for Mosel include Xpress NonLinear for solving Nonlinear Programming problems (module 'mmxnlp'), and the module 'mmnl' gives access to the QCQP solver (for quadratically constrained problems) within Xpress Optimizer. Each solver module comes with problem-type specific functionality (such as variable and constraint types)—please see the corresponding manuals.

Further information

- **Xpress Kalis** provides access to the functionality of the **Constraint Programming** solver Kalis from within the Mosel environment or through the Kalis library APIs. For more information about Xpress Kalis see the documents Xpress Kalis Mosel User Guide, Xpress Kalis Mosel Reference Manual, and Xpress Kalis Libraries User Guide.
- **Xpress NonLinear** and **Xpress Global** provide access to a set of solvers for **Non-Linear** and **Mixed Integer Non-Linear Programming**. Xpress Global provides a branch-and-cut based algorithm for solving nonlinear and mixed-integer nonlinear problems to global optimality. Xpress NonLinear automatically selects a solver among the installed solvers of the Xpress suite (Simplex, Barrier, SLP, or Knitro) depending on the detected problem type. For more information about Xpress see the chapter mmxnlp of the "Xpress Mosel Language Reference Manual", the Xpress NonLinear Manual and the Xpress Global Reference Manual.

3.1.7 Deploying Mosel models to Xpress Insight

Xpress Insight has its own installer that needs to be executed in addition to the standard installation of the Xpress suite. Please see the *Xpress Insight quick installation guide* for details on the installation process.

Among the examples in the `examples/getting_started/Mosel` subdirectory of the Xpress installation you will find the two Insight app archives `folioinsight.zip` and `folioinsightxml.zip`. Both archives include a slightly edited version of the Mosel model file `foliodata.mos` with the required input datafile, the second archive adds an XML configuration file and VDL view definitions for the Xpress Insight GUI.

The *Evaluation Scenario 5* in Section 3.5 shows how to work with Xpress Insight.

Publishing Insight apps from Xpress Workbench

Now let us see how to re-generate the readily provided app archives with Workbench. Locate the archive `folioinsightxml.zip` among the examples in the `examples/getting_started/Mosel`

subdirectory of the Xpress installation and unzip it into a new directory, say `folioapp`. Start up Workbench with the option *Open a project* (or if Workbench is already open use the menu *Project* ≫ *Open Project*) and select the directory `folioapp` as the project location. Explore the contents of the app archive by opening the project source files (.mos, .xml, .vdl).

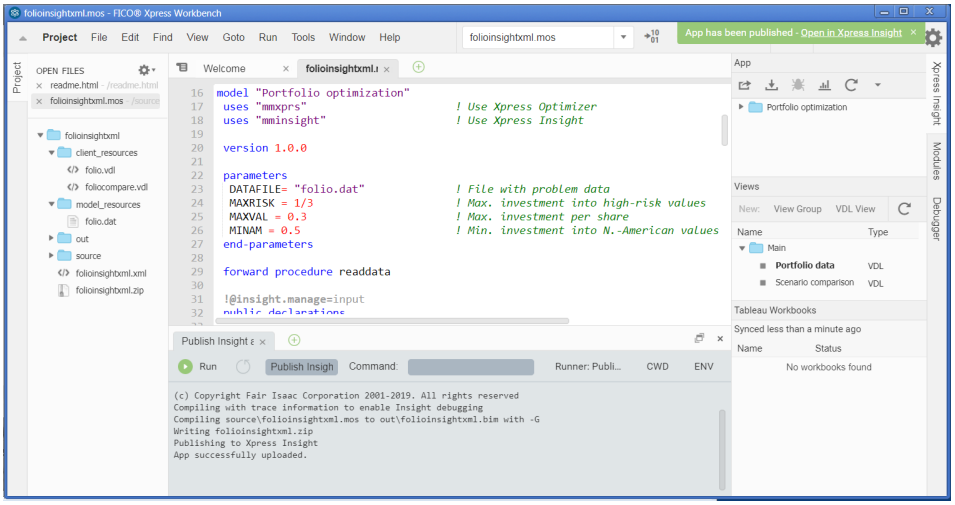


Figure 3: Publishing an Insight app from Xpress Workbench

To create a new app, select the 'publish to Insight' button of Workbench: you will be prompted to enter your Insight credentials (for a default desktop installation these are `admin / admin123`) and specify a new name for your app. After successfully publishing the app a green box with a link to the new app in the Insight Web Client appears at the top right corner of the Workbench workspace. Following this link will take you to the app loaded into the Insight Web Client as shown in Figure 6. Click onto the grey area to select and load a scenario, then choose the 'Run' button (for further detail, please see the instructions for the Insight Web Client in Section 3.5).

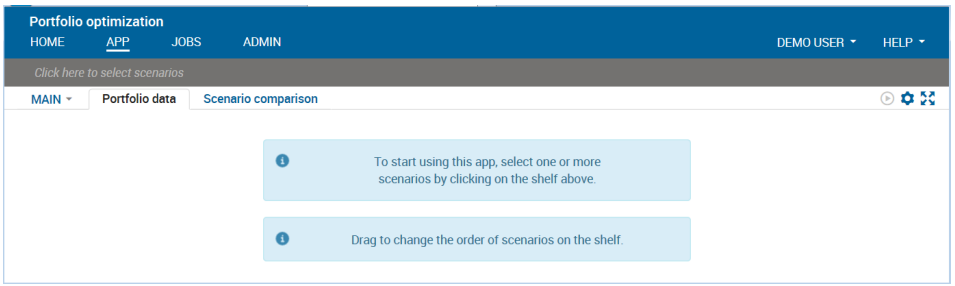


Figure 4: Xpress Insight Web Client after app loading

Further information

- Introductory example: "Getting Started with Xpress", Section 9.4 'Deployment to Xpress Insight'.
- Further examples: Xpress Insight Web Client User Guide.
- Documentation: Xpress Insight Developer Guide.

3.1.8 Deploying Mosel models to a host language

Open a Mosel model and select *Deploy* ≫ *Deploy* or click the deploy button. This will open the Deploy dialog box. Select the programming language you wish to run the Mosel model from, and click the *Next>* button. This will open a Source Code Dialog containing the code for deployment.

Further information

- Introductory example (Java): "Getting Started with Xpress", Chapter 9: 'Embedding a Mosel model in an application'.
- Further examples (C/Java/C#/VBA): "Xpress Mosel User Guide", Part III: 'Working with the Mosel Libraries'.
- Documentation of Mosel C libraries: Xpress Mosel Library Reference Manual.
- Documentation of Mosel Java libraries: Xpress Mosel Library Reference Manual JavaDoc.
- Documentation of the Mosel .NET interface: Xpress Mosel .NET Interface.

3.1.9 Mosel console commands

As an alternative to running Mosel models within Xpress Workbench you may execute them with the Mosel standalone version. This mode is often preferrable for testing and experimentation, for instance, if you wish to invoke a sequence of model runs from a batch file.

At the command prompt, type the following command:

```
mosel exec foliolp
```

You will see output that looks like the following:

```
Total return: 14.0667
treasury: 30%
hardware: 0%
theater: 20%
telecom: 0%
brewery: 6.66667%
highways: 30%
cars: 0%
bank: 0%
software: 13.3333%
electronics: 0%
```

You can also specify new values for model parameters to be applied when executing the model:

```
mosel exec foliodata MAXVAL=0.5 OUTFILE="result2.dat"
```

Further information

See "Xpress Mosel User Guide", 1.1 'Entering a model'. See also "Xpress Mosel Reference Manual", 1.1 'What is Mosel' – 'Running Mosel'.

3.1.10 Other Mosel topics

In addition to the topics addressed in this guide, the document "Getting Started with Xpress" describes how to draw **SVG user graphs** in Xpress Workbench (Chapter 5: 'Drawing user graphs') and how to

program heuristics with Mosel (Chapter 8: 'Heuristics'). Further details on these topics can be found in the "Xpress Mosel User Guide", Part II 'Advanced language features' that describes Mosel's programming facilities and includes other examples of heuristics, and under Part IV 'Extensions and tools' – 16.3 'Graphics with mmsvg'.

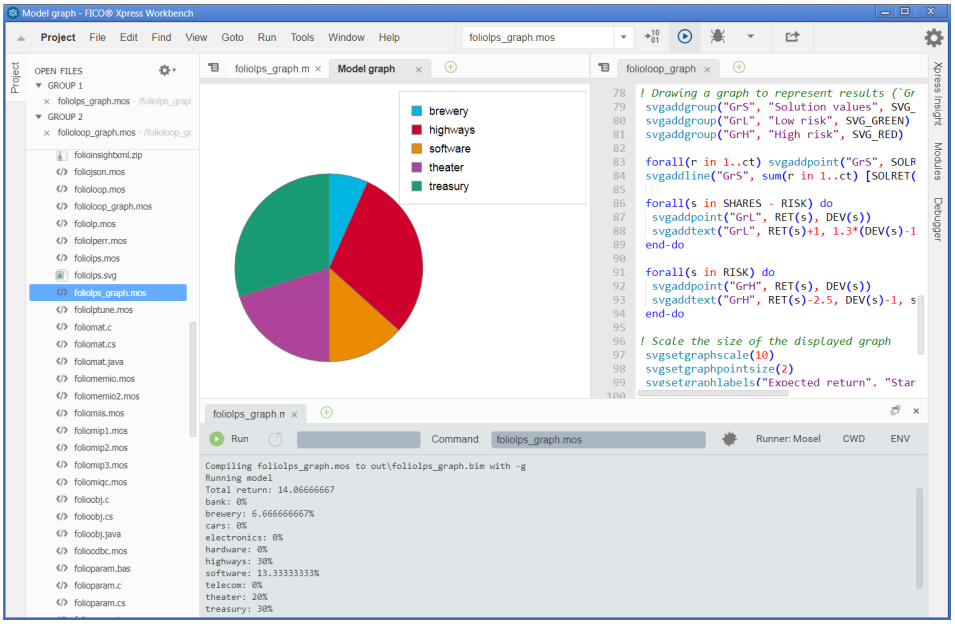


Figure 5: Workbench with SVG user graph

Additional examples of modeling and programming with Mosel can be found in the directory `\xpressmp\examples\`. Also, the book "Applications of Optimization with Xpress-MP" (Dash Optimization, 2002) shows how to formulate and solve a large number of application problems with Xpress, accessible online: http://examples.xpress.fico.com/example.pl#mosel_book

3.2 Evaluation steps for Scenario 2

Implement and deploy a model in a programming language using the object-oriented API for Xpress solvers.

Tool for model development:	Object-oriented solver API
Type of problem:	LP, MIP, QP, MIQP, NLP, MNLP
Tool for model deployment:	Object-oriented solver API (C++, Java, .NET)

Implementations in C++, Java and C# of the optimization problems from the guide "Getting Started with Xpress" along with some other examples of different types of optimization problems can be found in the Xpress installation under
`xpressmp\examples\solver\optimizer\cpp\objects`,
`xpressmp\examples\solver\optimizer\java\objects` and
`xpressmp\examples\solver\optimizer\csharp\objects`.

Further information

More examples and the complete documentation of the object-oriented APIs can be found in the Solver C++ User Guide, the Solver Java User Guide, the Solver .NET User Guide, the Xpress Optimizer C++

Reference, the Xpress Optimizer Javadoc and the Xpress Optimizer .NET Library Reference. Please refer to the Creating Xpress Solver Applications guide on how to setup and run programs with the Xpress libraries.

3.3 Evaluation steps for Scenario 3

Use your custom application to develop the model and then call the Xpress Optimizer libraries. Available for problems that can be solved with Xpress Optimizer

Tool for model development:	Custom application that calls optimizer libraries.
Type of problem:	LP, MIP, QP, MIQP
Tool for model deployment:	Optimizer libraries (C++/C, Java, .NET, Python, R)

See "Getting Started with Xpress", Part III: 'Getting started with the Optimizer', which presents examples in C language with detailed explanation. The corresponding example files can be found in your Xpress installation (directory `xpressmp\examples\getting_started\Optimizer`). The Optimizer Java and .NET interfaces provide similar functionality.

If you are working with Python please refer to the examples that are documented in the chapter 'Examples of use' of the "Xpress Optimizer Python Interface User Manual".

Further information

- Xpress Optimizer Reference Manual
- Xpress Optimizer Python Interface User Manual
- Xpress Optimizer R Interface Reference Manual
- For Non-linear programming (NLP,MNLP) see Xpress NonLinear Reference Manual and Xpress Global Reference Manual.
- Please refer to the Creating Xpress Solver Applications guide on how to setup and run programs with the Xpress libraries.

3.4 Evaluation steps for Scenario 4

Run a matrix that is readily available in LP or MPS format. Such a model may be executed through console commands, or applications that call Xpress Optimizer library functions.

3.4.1 Console commands

At the command prompt, type the following sequence of commands to execute the MPS matrix in file Folio.mps:

```
optimizer
readprob Folio
chgobjsense max
lpoptimize
printsol
quit
```

You will see output that looks like the following:

```
>optimizer
FICO Xpress Solver 64bit v9.1.0 Apr 14 2023
(c) Copyright Fair Isaac Corporation 1983-2023. All rights reserved
 Optimizer v41.01.07   [C:\xpressmp\bin\xprs.dll]
[xpress C:\] readprob Folio

Reading Problem FolioLP

Problem Statistics
           3 (      0 spare) rows
          10 (      0 spare) structural columns
          19 (      0 spare) non-zero elements
MIP Entity Statistics
           0 entities        0 sets        0 set members
[xpress C:\] chgobjsense max
[xpress C:\] lpoptimize
Maximizing LP FolioLP
Original problem has:
         3 rows           10 cols           19 elements
Presolved problem has:
         3 rows           10 cols           19 elements
Presolve finished in 0 seconds
Heap usage: 391KB (peak 548KB, 130KB system)

Coefficient range                    original                 solved
  Coefficients   [min,max] : [ 1.00e+00,  1.00e+00] / [ 1.00e+00,  1.00e+00]
  RHS and bounds [min,max] : [ 3.00e-01,  1.00e+00] / [ 3.00e-01,  1.00e+00]
  Objective      [min,max] : [ 5.00e+00,  3.10e+01] / [ 5.00e+00,  3.10e+01]
Autoscaling applied standard scaling

   Its        Obj Value      S   Ninf  Nneg        Sum Inf  Time
     0        42.600000      D      2     0       3.166667     0
     5        14.066659      D      0     0        .000000     0
Uncrunching matrix
Optimal solution found
Dual solved problem
  5 simplex iterations in 0.01 seconds at time 0

Final objective                          : 1.406665900000000e+01
  Max primal violation      (abs / rel) :       0.0 /       0.0
  Max dual violation        (abs / rel) :       0.0 /       0.0
  Max complementarity viol. (abs / rel) :       0.0 /       0.0
All values within tolerances
[xpress C:\] printsol

Problem Statistics
Matrix FolioLP
Objective *OBJ*

RHS *RHS*
Problem has      3 rows and     10 structural columns

Solution Statistics
Maximization performed
Optimal solution found after      5 iterations
Objective function value is    14.066659
type c/r to continue, anything else to finish >

Rows Section
   Number   Row     At      Value      Slack Value   Dual Value        RHS
 E     1  Cap       EQ      1.000000       .000000      8.000000      1.000000
 G     2  NA        LL       .500000       .000000     -5.000000       .500000
 L     3  Risk      UL       .333333       .000000     23.000000       .333333
type c/r to continue, anything else to finish >

Columns Section
   Number  Column   At      Value      Input Cost   Reduced Cost
```

```
 C      5  frac(1)   UL       .300000      5.000000      2.000000
 C      6  frac(2)   LL       .000000     17.000000     -9.000000
 C      7  frac(3)   BS       .200000     26.000000       .000000
 C      8  frac(4)   LL       .000000     12.000000    -14.000000
 C      9  frac(5)   BS       .066667      8.000000       .000000
 C     10  frac(6)   UL       .300000      9.000000      1.000000
 C     11  frac(7)   LL       .000000      7.000000     -1.000000
 C     12  frac(8)   LL       .000000      6.000000     -2.000000
 C     13  frac(9)   BS       .133333     31.000000       .000000
 C     14  frac(10)  LL       .000000     21.000000    -10.000000
[xpress C:\] quit
```

Further information

Xpress Optimizer Reference Manual", Chapter 6: 'Console and Library functions'.

3.4.2 Xpress Optimizer library functions

See "Getting Started with Xpress", Chapter 14: 'Matrix input', which presents an example in C language with detailed explanation.

3.5 Evaluation steps for Scenario 5

Xpress Insight has its own installer that needs to be executed in addition to the standard installation of the Xpress suite. Please see the *Xpress Insight quick installation guide* for details on the installation process.

3.5.1 Inspecting Insight apps with the Insight Web Client

With a default desktop installation, start up the *Xpress Insight Web Client* by directing your web browser to `http://localhost:8860/` and use the default credentials of `admin / admin123`. Mosel models with their data files and optional configuration settings are input into Xpress Insight in the form of app archives. Among the examples in the `examples/getting_started/Mosel` subdirectory of the Xpress installation you will find the two archives `folioinsight.zip` and `folioinsightxml.zip`. Both archives include a slightly edited version of the Mosel model file `foliodata.mos` with the required input datafile, the second archive adds an XML configuration file and VDI view definitions for the Xpress Insight GUI. Figure 6 shows a screenshot of the Xpress Insight GUI with the `folioinsightxml.zip` app.

Follow the instructions in Section 9.4.3 'Working with the Xpress Insight Web Client' of the 'Getting Started' guide to load the `folioinsightxml.zip` app and run a scenario. Edit the parameters and input data in the view *Portfolio data* and re-run via the 'Run optimization' button. You may also want to clone the scenario, edit some of its data (for example, try setting values 0.2 or 0.4 for the parameter 'Maximum investment per share') and compare the results in the view *Portfolio data*.

Further information

- Introduction to working with Xpress Insight: Xpress Insight Web Client User Guide.
- Documentation: Xpress Insight Developer Guide.
- Examples: load and explore the Insight app examples under the folder `examples/insight` of the Xpress distribution, in particular `quick_start.zip` (app template) and the demo of VDL features in `vdl/vdl_language.zip`.

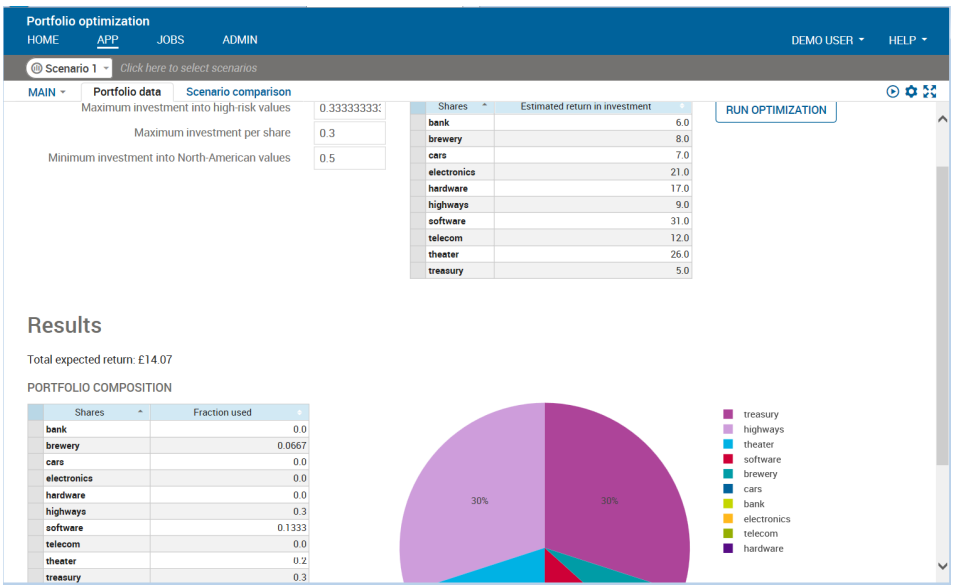


Figure 6: Xpress Insight Web Client