FICO® Xpress Workbench™

3.16

USER GUIDE

FICO® Platform

FICO®

FICO® Xpress Workbench 3.16

Deliverable Version: A

Last Revised: July 25, 2025

Contents

CHAPTER 1:
Introducing FICO® Xpress Workbench 7

What is the FICO® Analytic Cloud (FAC) 8
FICO® Platform 8
Installing and Launching the Desktop Edition of Xpress Workbench 8
Supported versions 9
Logging in and out of the FAC Edition of Xpress Workbench 9

CHAPTER 2:
Getting Started 11

First Steps 12
Creating a Project 12
Creating Projects with the FAC Edition 12
Create a Project with the Desktop Edition 13
The Xpress Workbench IDE 14
Editing Source Files 15
Publishing to Xpress Insight 16
Debugging Xpress Insight Scenarios 17
Setting Breakpoints 17
Debugging the Scenario 17
Finishing the Debug Session 18

CHAPTER 3:
Working with Projects 19

Creating Projects 19
Creating Projects with the FAC Edition 19
Create a Project with the Desktop Edition 20
Opening and Closing Projects with the Desktop Edition 21
Renaming Projects - FAC Edition Only 21
Deleting Projects - FAC Edition Only 21
Opening Projects 21
Backing up and Restoring Projects - FAC Edition Only 22
Displaying the Dashboard - FAC Edition Only 22

CHAPTER 4:
Editing Source Files 23
Opening Files 23
Creating Files 23
Creating Files from the File Tree 24
Creating Files from the File Menu 24
Creating Files from a Template 24
Creating Files from the View Designer 24
Renaming Files 25
Uploading Files - FAC Edition Only 25
Downloading Files - FAC Edition Only 26
Downloading the Project 26
Viewing File Revision History - FAC Edition Only 26
Coding Assistance 27
Jump to Definition 28
Finding Text 28
Replacing Text 29
Finding Text Across Many Files 30
Replacing Text Across Many Files 30
Highlighting Syntax 30
Automatically Completing Words 31
Automatically Completing Statements 31
Inline Documentation 32
Folding Code 32
Folding Code with the Mouse 33
Folding Code with the Keyboard 33
Formatting Code 33
Collaborating with Other Users - FAC Edition Only 33
Sending Messages to Users 34

CHAPTER 5:
Running and Debugging Models 35
Running Models 35
Stopping a Model 36
Changing the Working Directory 36
Debugging Models 37
Starting an Xpress Insight Debug Session 37
Setting and Removing Breakpoints 38
Suspending and Resuming the Model 39
Stepping through the Model 39
Examining Variables 40

Examining Arrays . . . 40
Saving Array Data as CSV Files . . . 41
Evaluating Expressions . . . 41
Compiling Models . . . 42
Configuring Compiler Options . . . 42
Choosing a Python Environment Using Conda . . . 44
Running Submodels and Parallel Solving . . . 45
Preparing Template Files for Mosel Deployment Wrappers . . . 46
Tuning Mosel Models . . . 46
Starting a Tuning Session . . . 46
Tuner Settings . . . 48
Manually Enabling the Tuner . . . 48
Profiling Models . . . 49
Starting a Profiling Session . . . 49
Collecting Coverage Data . . . 50
Tracing Model Execution . . . 51

CHAPTER 6:
Debugging Python Models . . . 52
Starting a Python Debug Session . . . 53
Setting and Removing Breakpoints . . . 53
Suspending and Resuming the Model . . . 54
Stepping through the Model . . . 54
Examining Variables . . . 55
Examining Tabular Data . . . 55
Evaluating Expressions . . . 56
Using the Python Debug Prompt . . . 56

CHAPTER 7:
Working with Xpress Insight . . . 59
Configuring an Xpress Insight Server - Desktop Edition Only . . . 59
Managing Insight Views . . . 60
Managing Views . . . 62
Importing Views . . . 62
Deleting Views . . . 63
Managing View Groups . . . 63
Deleting View Groups . . . 64
Changing the View order . . . 64
Creating a New View . . . 64
View Designer Overview . . . 65
Using the View Designer and VDL to Create Custom Views . . . 69
Publishing to Xpress Insight . . . 71

Updating Xpress Insight Apps 72
Opening Xpress Insight 73
Exporting Xpress Insight Apps 73
Debugging Scenarios 74
Manually Executing Scenarios 74
Cancelling Debug Sessions 75
Unlinking Projects from Xpress Insight 75
Command-Line Tool for Common Operations 76
Command-Line Tool Options and Arguments 76

CHAPTER 8:
Authoring Tableau Views for Insight Applications 80
Creating Workbooks 80
Editing Workbooks 81
Downloading Workbooks into Xpress Workbench 82
Viewing the Workbook Status 82

APPENDIX A:
Contacting FICO 83
FICO Customer Support 83
FICO Community 83
Documentation 83
FICO Learning 84
About FICO 84

INDEX 85

CHAPTER 1

Introducing FICO® Xpress Workbench

FICO® Xpress Workbench is a tool for creating, editing, and debugging FICO® Xpress Insight apps and standalone Xpress Mosel models.

Xpress Workbench enables you to:

- Create and edit source files
- Publish Xpress Insight apps to an Xpress Insight server
- Debug an Xpress Insight scenario while it is executing within an Xpress Insight server
- Build an Xpress Insight app archive
- Debug a Mosel model in a standalone instance of Mosel
- Compile a Mosel model into a BIM file for use in FICO® Xpress Executor or another application powered by Mosel

Xpress Workbench is available on FICO® Platform served on the FICO® Analytic Cloud (FAC) and also as a desktop version. Help topics that are specific to one version or the other will include *FAC Edition* or *Desktop Edition* in the topic title. Additional documentation resources that you are likely to need during the course of development are available and listed below.

- *Xpress Insight Mosel Developer Guide*
  The guide to Mosel app development with Xpress Insight
- *Xpress Insight Python Developer Guide*
  The guide to Python app development with Xpress Insight
- *Xpress Insight Mosel Reference*
  The reference for the Mosel interface to Xpress Insight
- *Xpress Insight Python Reference*
  The reference for the *xpressinsight* Python package for Xpress Insight
- *Mosel Language Reference Manual*
  The main reference for Xpress-Mosel, including details of the Mosel language and all modules supplied with the standard distribution
- *Mosel IO Whitepaper*
  A guide to file handling in Mosel

- *Optimizer Reference Manual*
  The main reference for FICO® Xpress
- *xpress_docs.zip*
  The complete documentation for the Xpress Suite

What is the FICO® Analytic Cloud (FAC)

The FICO® Analytic Cloud (FAC) delivers a comprehensive toolkit of cloud-enabled products, across analytics, decision management and optimization through the FICO® Platform, an integrated platform solution in which you can build, deploy, execute, and scale analytic-powered applications.

The Platform serves as the primary cloud infrastructure for building, hosting, and managing FICO components and apps. For more visit www.ficoanalyticcloud.com.

FICO® Platform

The FICO® Platform is a high-performance execution platform that provides a complete environment for developing, distributing, and using decision management solutions.

It can operationalize and connect analytically-powered applications and decisions to provide centralized control to author, develop, and deploy applications faster than any other solution.

Installing and Launching the Desktop Edition of Xpress Workbench

The desktop version of Xpress Workbench is installed alongside Xpress.

If you intend to use Xpress Insight in conjunction with Xpress and Xpress Workbench, ensure that you install Xpress Insight as the final install after Xpress and its automatic Xpress Workbench install.

To launch Xpress Workbench:

- In Windows, open the **Start** menu and click the **Xpress Workbench** icon in the **FICO** folder.
- On a Mac, open the **FICO Xpress** folder in **Applications** and click the **Xpress Workbench** icon.

You will be presented with an initial welcome page offering you the following options:

- **Recent projects** enables you to choose a recent project to open.
- **Create project**
- **Open existing file or folder** enables you to open an existing Workbench project, or open a source file from a folder on your computer.

- **Examples** contains a range of Mosel example files, such as:
  - Introductory examples
  - Insight examples
  - Nonlinear examples
  - Calling R from Mosel
  - Calling Python from Mosel
  - Browse all examples
- Xpress Documentation-link to the Mosel documentation on the FICO Community
- Ask the Xpress Community-link to the Optimization forum in the FICO Community

Supported versions

Xpress Workbench is available for Windows and macOS as part of the FICO Xpress installer. (For more information, see Supported Platforms.)

You can also install Xpress Workbench in a Docker container. For more information, see the Docker Deployment Guide.

Logging in and out of the FAC Edition of Xpress Workbench

Before you can use Xpress Workbench you must log in.

This topic provides information on obtaining the URL and logging in.

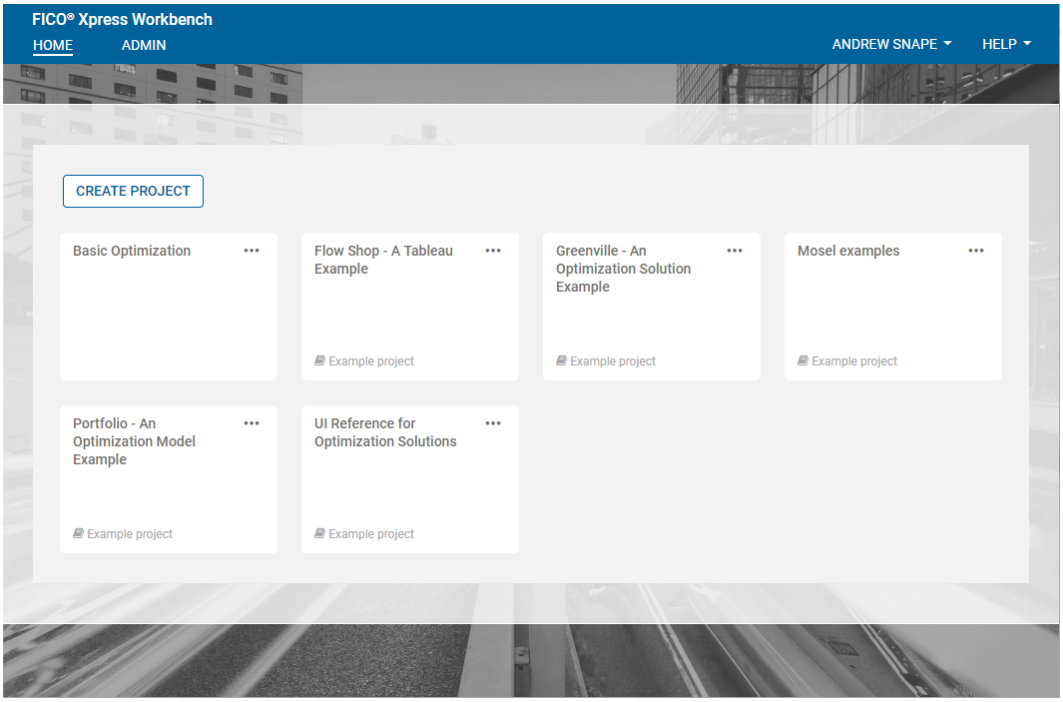
Logging In

To log in, you must know the following:

- Your login URL - contact your administrator to find out the URL.
- Your FICO Analytic Cloud username and password.

To log into Xpress Workbench, perform the following steps:

1. Enter the Xpress Workbench URL into the address bar of your browser.
2. Enter your username and password into the **Log in** page.
3. Click **Log In**. The Xpress Workbench Dashboard page is displayed.



From here you can open a project or create a new one.

Logging Out

Log out of Xpress Workbench before closing the browser or navigating to a different URL.

1. Click the arrow next to your username in the page header.
2. Click **Exit** in the drop-down list.

CHAPTER 2

Getting Started

This section contains brief references to resources that will help you convert existing Xpress Mosel models to work with corresponding FICO® Xpress Insight apps, or import Xpress Insight apps into FICO® Xpress Workbench.

Creating an Xpress Workbench Project

A project contains all the source files that make up an optimization app. To create a new Xpress Workbench project in either the Desktop Edition or the FAC Edition, see the later topic named **Creating a Project**.

Open an Existing Xpress Workbench Project

To open a project in Xpress Workbench: see the later topic named Opening Projects in Chapter 3-**Working with Projects**.

Converting Xpress Mosel Models to Xpress Insight apps with the Desktop Edition

To open an existing Xpress Workbench project within the desktop version, see the later topic named **Opening and Closing Projects with the Desktop Edition**.

If you wish to convert existing Xpress Mosel models and build them into FICO® Xpress Insight apps, refer to Chapter 6 and subsequent sections of the Xpress Insight Mosel Developer Guide.

Importing Xpress Insight apps to Xpress Workbench with the FAC Edition

Developers can import an Xpress Insight app into Xpress Workbench to continue development. The process is designed to be as streamlined as possible.

Perform the steps outlined below:

1. Log in to Xpress Workbench and create a new project.
2. Load your new project by clicking on its tile on the **Home** page.
3. A new project contains some default files and folders which will not be needed. Using the right mouse-button menu, delete:
   - `client_resources` (folder)
   - `model_resources` (folder)

- `source` (folder)
- `application.xml`

4. Using your operating system's file explorer, navigate to the root folder of your Xpress Insight app and upload all relevant files and folders into your new, now empty project.

First Steps

This section contains a tutorial that shows you how to use Xpress Workbench to create an Xpress Insight app and publish it to an Xpress Insight server. The following topics are covered:

- Creating projects
- Editing source files
- Publishing to Xpress Insight
- Debugging Xpress Insight scenarios

Creating a Project

A project contains all the source files that make up an optimization app.

In this section you will create a new project and open it.

Creating Projects with the FAC Edition

Complete the following steps to create a project:

1. From the **Dashboard** page, click **Create Project**.
2. Enter a name for the project.
3. (Optional) If you have a ZIP file containing project files that you want to upload, click **Choose File** and select your ZIP file.
4. Click **Save**.
   - If you uploaded a ZIP file, it is extracted into the new project.
   - If you did not upload a ZIP file, your new project contains the following files, which make up a minimal Xpress Insight application:
     - `application.xml` - an Xpress Insight companion file that defines the app name and some custom views
     - `source/application.mos` - a Mosel source file containing the minimal code needed to run within Xpress Insight
     - `model_resources/input.dat` - a data file which the Mosel model will read to initialize its input data
     - `client_resources/introduction.vdl` - a VDL custom view that displays information about the project

- `client_resources/input.vdl` - a VDL custom view that displays some input data from the scenario
- `client_resources/results.vdl` - a VDL custom view that displays some result data from the scenario
- `client_resources/css/application.css` - a CSS file where you can add custom stylesheet rules for all the VDL views

Create a Project with the Desktop Edition

A project contains all the source files that make up an Xpress Insight app or Mosel application.

Create a new Mosel project, Xpress Insight Mosel project, or Xpress Insight Python project.

1. To create a new project:
   - To create a simple Mosel project, choose the **Create Project > CREATE MOSEL PROJECT** menu option.
   - To create a new Xpress Insight project, you must select to work in either Python or Mosel.
     - To develop using Mosel, choose the **Create Project > CREATE INSIGHT (MOSEL) PROJECT** menu option.
     - To develop using Python, choose the **Create Project > CREATE INSIGHT (PYTHON) PROJECT** menu option.
2. Navigate your filesystem to a folder where you wish to store your project and click **OK**.

   The project opens—depending on the kind of project you created, it will contain a minimal Xpress Insight app for Python or Mosel, or a minimal Mosel application. You can use a Mosel project to begin the development of a standalone Mosel model. The remainder of this guide focuses, however, on activities relating to Xpress Insight Mosel projects and apps.

   Note: The Xpress Insight Python Developer Guide offers information on using Python with Xpress Workbench.

Opening and Closing Projects with the Desktop Edition

After a project has been created, you can open and close it in subsequent Xpress Workbench sessions.

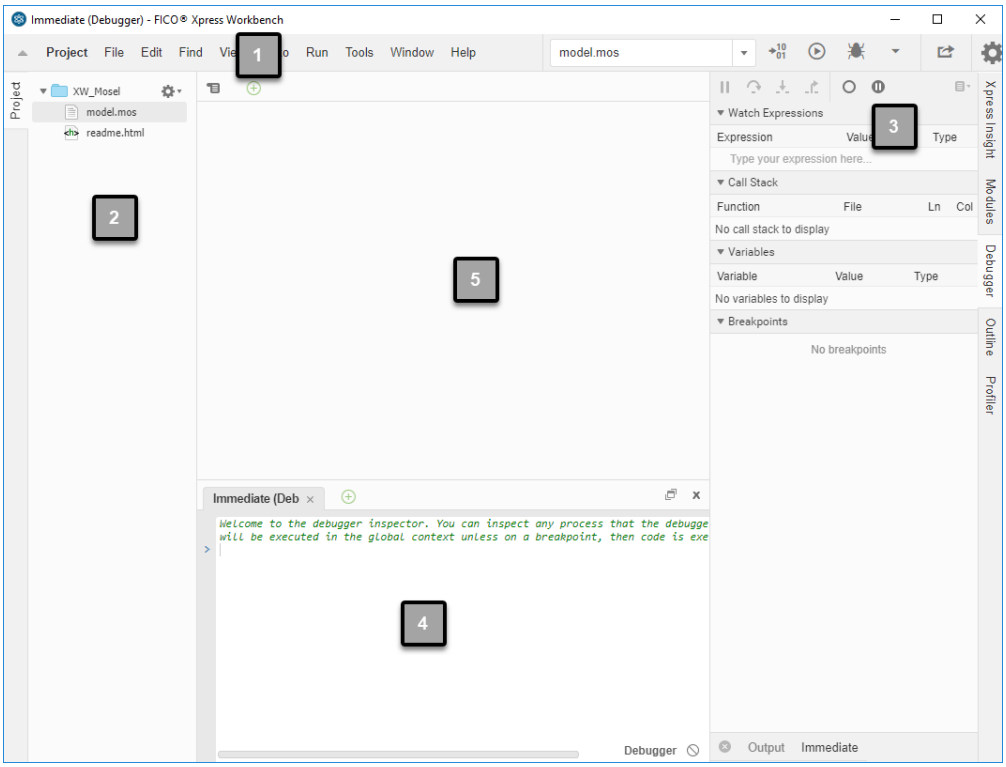
To open an existing project select **Project** from the top-level menu, choose **Open Project**, follow the prompts and use the file system navigator to locate the folder in which your existing project is located.

To close an open project, select **Project** from the top-level menu and choose **Close Project**.

The Xpress Workbench IDE

The Xpress Workbench Integrated Development Environment contains five major panels.

The Xpress Workbench IDE helps you edit every type of source file that makes up your optimization app, along with many other file types. It also offers an intuitive representative method for editing views (vdl files) in a drag and drop interface.



Item	Description
1	Top Menu ■ On the left is displayed the major command menus that you will use to create and edit your Mosel file. ■ On the right are the buttons which compile, run, and debug the current file. ■ There are also buttons that enable editing of the Compiler Options and Project Settings (Preferences).
2	Left sidebar—Tabbed panels allow you to: ■ see a hierarchical view of the current **Project**, to view and open any file in the current project. ■ View a list of Xpress Workbench **Commands**. ■ When a .vdl file is selected, the **Palette** displays the elements that can be used to create Views for Xpress Insight. **Note:** The user interface only displays tabs appropriate to the type of file that is open and active in Workbench.

Item	Description
3	Right sidebar—Pop out menus enable you to: ■ Connect to **Xpress Insight**, to build and publish Insight apps. ■ View a complete list of **Modules** available to your Mosel script. ■ Open the **Outline** tab to view the structure of the current Mosel file, listing the declarations. Clicking on a displayed element will highlight the declaration in the code editor (If the declaration is in a different file due to a Mosel *include* statement, Workbench will open the source file if necessary). ■ Use the **Profiler** to display any profiling data contained in the accompanying .mos.prof file, generated by the Mosel profiler — an error message will display if the data cannot be read. For more, see Profiling Mosel Models.
4	Bottom tabbed pane ■ The results from each executed Mosel file are displayed in a new tab in this area.
5	File editing pane

Editing Source Files

In this section you will edit some of these files to add functionality to the app.

Editing Mosel Files

When you use Xpress Workbench to create a new Mosel project, a correctly structured Mosel model file named `model.mos` is automatically generated in the project folder. This model file provides the basis for your model creation.

Note: Xpress Workbench supports code folding, auto-completion, and syntax highlighting in the Mosel language. For more see Coding Assistance on page 27.

Editing Insight App Source Files

Every new Xpress Insight project contains the files for a minimal Xpress Insight app. Developers can use Xpress Workbench to create visual applications hosted in Xpress Insight, known as *Views*, that enable non-technical business users to interact with optimization models, such as those written in Mosel, and perform in-depth what-if analysis.

Business users work with underlying models using visual components and business terms to specify values for the scenarios they are exploring. Scenarios can be iteratively developed and executed, and their results analyzed.

Xpress Workbench provides the View Designer to enable rapid development of these custom views by allowing you to associate user interface components with model data for viewing and editing.

For more see Using the View Designer and VDL to Create Custom Views on page 69

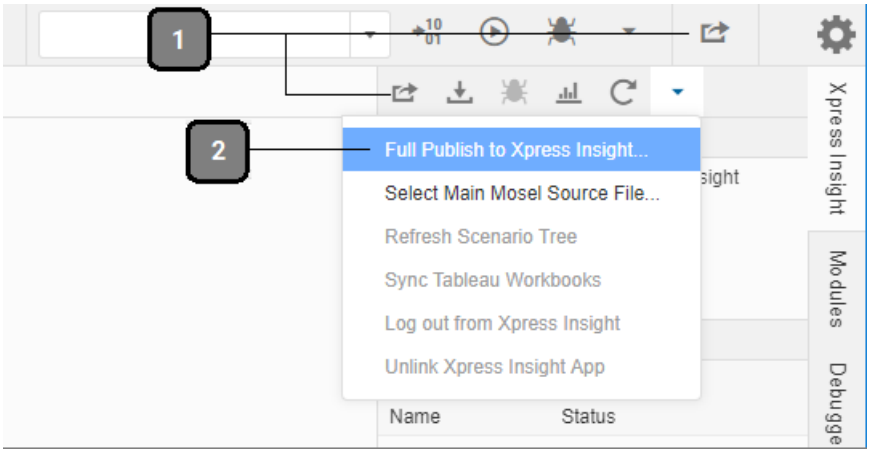
Publishing to Xpress Insight

In this section you will publish the project to an Xpress Insight server and test it.

If you do not have access to an Xpress Insight server, contact your administrator.

Note: If you are working with the Desktop Edition and this is the first time you have used Xpress Insight, you will first need to configure a server. For more information, see Configuring an Xpress Insight Server on page 59.

All Xpress Workbench features that relate to Xpress Insight are accessed from the **Xpress Insight** sidebar. You can find this sidebar on the right side of the Xpress Workbench **Project** page. There are two options when publishing to Xpress Insight, Full or Quick.



1. Quick Publish - Use this option to include only the files that have been changed since the last publish
2. Full Publish - Always use this option when publishing a project to Xpress Insight for the first time. Additionally, use this option:
   - when adding a new view.
   - for the final publish prior to release.

Complete the following steps to publish a project to Xpress Insight for the first time:

1. Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.
2. Click the **Publish** button, which is the first button from the left within the sidebar.
3. Use the **Xpress Insight Server** drop-down to select the server where you want to publish the app.
4. In the list of Xpress Insight apps, select **Create a new app**.

   Note: If you are using the FAC Edition on the FICO® Analytic Cloud, the list of Xpress Insight apps may take several seconds to load.
   By default, the list of servers includes servers from all solutions. To filter the list to include only servers in the current solution, select **Show only from current solution**.
5. Click **Publish**.

6 Log on to Xpress Insight in a new tab, open the new app, and review the VDL view that you created.

Debugging Xpress Insight Scenarios

In this section you will debug a Mosel model within Xpress Insight by executing a scenario.

Before continuing, make sure you have followed the steps in the previous section to publish a project to Xpress Insight.

Note: For more on Debugging Mosel models, see Running and Debugging Models on page 35

Setting Breakpoints

Before you start debugging, set a breakpoint in the Mosel model. When the model reaches a breakpoint, the scenario being executed pauses. While the scenario is paused, you can use the debugger to inspect the state of the model.

Breakpoints are set by clicking in the breakpoint area, which is the narrow grey area where the line numbers for the source file are displayed. When you hover the mouse to the left of the line numbers, the cursor changes to a hand icon to show that clicking here sets a breakpoint.

1 In the file tree, locate the **source** folder and click the arrow icon to its left.

2 When the folder expands to reveal **application.mos**, double-click its filename.

3 Locate the following line (near line 25):
```
initializations from "input.dat"
```

4 Click in the breakpoint area for this line. A red disk appears to show that a breakpoint has been set.

Debugging the Scenario

Complete the following steps to start debugging a scenario:

1 Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.

2 Expand the scenario tree and select the scenario you want to debug.

3 Click the **Debug** button, which is the third button from the left within the sidebar.

4 In the menu that appears, click **Debug Scenario - LOAD**.

- The app is published to Xpress Insight before the debug session starts.
- The line with the breakpoint is highlighted to show that model execution is suspended at that line.
- The **Debugger** sidebar is open, where you can see the call stack and the values of the variables defined by the model.

Now that the debug session is established, you can step through the model and see how the state changes as the execution progresses.

5 In the **Debugger** sidebar, find the **Variables** section and click the **arrow** icon to the left of **Globals**.

   You see a list of the global variables defined in the model. The value of *MY_INTEGER* is 0.

6 Step over the current line by clicking the second button from the left in the **Debugger** sidebar, or by pressing F10.

   The next executable line is highlighted to show that model execution has moved forward.

7 Expand the **Globals** entry within the **Variables** section again.

   The value of *MY_INTEGER* is now 10.

Finishing the Debug Session

Complete the following steps to finish the debug session:

1 Resume the model by clicking the first button from the left in the **Debugger** sidebar, or by pressing F8.

   The last line of the model is highlighted to show that the model is suspended after having finished executing. You can now review the final state of the model.

2 Click the **Resume** button one more time, and click **OK** when prompted.

CHAPTER 3

Working with Projects

A project is a collection of source files in Xpress Workbench and typically includes:

- Mosel source files
- Input data files for the Mosel model
- The Xpress Insight companion file
- HTML and JavaScript custom views
- VDL custom views
- Tableau workbooks

When you first log into Xpress Workbench, you see the **Dashboard** page. Here you can do the following tasks:

- See the projects on the server
- Create new projects
- Rename projects
- Delete projects
- Open projects

Creating Projects

The steps required to create a new project and open it in both the Desktop and FAC Edition are covered in the **Getting Started** chapter.

See the earlier topic Creating a Project on page 12.

Creating Projects with the FAC Edition

Complete the following steps to create a project:

1. From the **Dashboard** page, click **Create Project**.
2. Enter a name for the project.
3. (Optional) If you have a ZIP file containing project files that you want to upload, click **Choose File** and select your ZIP file.

4 Click **Save**.

- If you uploaded a ZIP file, it is extracted into the new project.
- If you did not upload a ZIP file, your new project contains the following files, which make up a minimal Xpress Insight application:
  - `application.xml` - an Xpress Insight companion file that defines the app name and some custom views
  - `source/application.mos` - a Mosel source file containing the minimal code needed to run within Xpress Insight
  - `model_resources/input.dat` - a data file which the Mosel model will read to initialize its input data
  - `client_resources/introduction.vdl` - a VDL custom view that displays information about the project
  - `client_resources/input.vdl` - a VDL custom view that displays some input data from the scenario
  - `client_resources/results.vdl` - a VDL custom view that displays some result data from the scenario
  - `client_resources/css/application.css` - a CSS file where you can add custom stylesheet rules for all the VDL views

Create a Project with the Desktop Edition

A project contains all the source files that make up an Xpress Insight app or Mosel application.

Create a new Mosel project, Xpress Insight Mosel project, or Xpress Insight Python project.

1 To create a new project:

- To create a simple Mosel project, choose the **Create Project > CREATE MOSEL PROJECT** menu option.
- To create a new Xpress Insight project, you must select to work in either Python or Mosel.
  - To develop using Mosel, choose the **Create Project > CREATE INSIGHT (MOSEL) PROJECT** menu option.
  - To develop using Python, choose the **Create Project > CREATE INSIGHT (PYTHON) PROJECT** menu option.

2 Navigate your filesystem to a folder where you wish to store your project and click **OK**.

The project opens—depending on the kind of project you created, it will contain a minimal Xpress Insight app for Python or Mosel, or a minimal Mosel application. You can use a Mosel project to begin the development of a standalone Mosel model. The remainder of this guide focuses, however, on activities relating to Xpress Insight Mosel projects and apps.

Note: The Xpress Insight Python Developer Guide offers information on using Python with Xpress Workbench.

Opening and Closing Projects with the Desktop Edition

After a project has been created, you can open and close it in subsequent Xpress Workbench sessions.

To open an existing project select **Project** from the top-level menu, choose **Open Project**, follow the prompts and use the file system navigator to locate the folder in which your existing project is located.

To close an open project, select **Project** from the top-level menu and choose **Close Project**.

Renaming Projects - FAC Edition Only

Complete the following steps to rename a project:

1. Hover the mouse over the project you want to rename.
2. Click the **pencil** icon that appears.
3. Enter a new name for the project.
4. Click **Save**.

Deleting Projects - FAC Edition Only

Complete the following steps to delete a project:

1. Hover the mouse over the project you want to delete.
2. Click the **cross** icon that appears.
3. Click **OK**.

Opening Projects

To open a project:

- If you are using the FAC Edition, when you are logged in to the FAC, click the project's name to open the **Project** page.
- If you are using the Desktop Edition, select **Project** from the top-level menu, choose **Open Project** , follow the prompts and use the file system navigator to locate the folder in which your existing project is located.

To close a project:

- If you are using the FAC Edition, save your work and close your browser window.
- If you are using the Desktop Edition, select **Project** from the top-level menu and choose **Close Project**.

Backing up and Restoring Projects - FAC Edition Only

If you are using FICO® Platform, your projects are automatically backed up.

To review the status of your backups, click **Admin** on the main Xpress Workbench menu. The **Backup status** page is displayed for the current component—Xpress Workbench. It presents a list of backups, their **Start times**, their **Status** and their **Durations**.

Backups are only made when it is detected that a project changes. They occur daily for up to the last 3 days, and then weekly for up to the last 5 weeks. If you have been working consistently on several projects, you are likely to see up to eight backups per project.

Click **View logs** to see a detailed log of a backup's execution - if an error has occurred, it will contain further information.

Click **Backup Archive** in the left-hand menu and select your project in the **Project** drop-down list to see a list of backups for a specific project. Click **Download backup** to download the backup file, a gzipped tar file.

Expand the archive in a convenient location on your local machine and the backup files are available for your use.

Displaying the Dashboard - FAC Edition Only

From the **Project** page, you can return to the **Dashboard** page by clicking the **Home** link in the page header.

CHAPTER 4

Editing Source Files

You can perform a full set of life-cycle operations on source files.

You can open, edit, and rename files in the Desktop edition. In addition, you can transfer files (upload and download) in the FAC edition.

Opening Files

Complete the following steps to open a file:

1. Locate the file you want to open in the file tree.
2. Choose one of the following ways to open a file:
   - Double-click the filename.
   - Right-click the filename and click **Open** on the menu that appears.

Opening Files with the FAC Edition

If a file cannot be opened within the Xpress Workbench IDE, you may be offered the option to download it for local editing.

Opening Files with the Desktop Edition

Depending on the file type, you may be presented with an option to open it in an external editor. For example, an XLSX file might be opened in Excel.

You can also open Mosel or VDL source files (MOS or VDL files) in Xpress Workbench directly from your operating system's file explorer-right click on the file and choose **Open**.

Creating Files

When it comes to creating files, Xpress Workbench offers four options:

- Using the file tree
- Using the file menu
- Using a template
- Using the View Designer

Creating Files from the File Tree

Complete the following steps to create a new file from the file tree:

1. In the file tree, locate the folder where you want to create the new file and right-click it. If you want to create a file in the root of the project, right-click the project name in the file tree.
2. Click **New** on the menu that appears.
3. In the submenu that appears, click the file template that you want to use.
4. Type the filename for the file and press **Enter**.

Creating Files from the File Menu

Complete the following steps to create a file from the **File** menu:

1. Open the **File** menu and click **New**.
2. In the submenu that appears, click the file template that you want to use.
3. Open the **File** menu and click **Save**.
4. In the **Save As** dialog, type the filename and select the folder where you want to create the new file.
5. Click **Save**.

Creating Files from a Template

Complete the following steps to create a file from a predefined template:

1. Open the **File** menu and click **New**.
2. In the submenu that appears, click the file template that you want to use.
3. Open the **File** menu and click **Save**.
4. In the **Save As** dialog, type the filename and select the folder where you want to create the new file.
5. Click **Save**.

Creating Files from the View Designer

Complete the following steps to create a new file using the New File icon in the View Designer:

The **New File** icon is displayed in the area above the artboard, to the right of the most recent tab.

Figure 1: New File icon

1. Click the new file icon.

2. Click **New** on the menu that appears, then in the sub menu, select the type of file you wish to create.

   The file is created and displayed in the central pane, surrounded by appropriate context sensitive tabs.

3. When you have finished editing the file, click `File > Save`.

   The **Save As** dialog is displayed.

4. Select the location for the file and click **SAVE**.

Renaming Files

Complete the following steps to rename a file:

1. Locate the file you want to rename in the file tree.
2. Right-click the filename.
3. Click **Rename** on the menu that appears.
4. Type the new filename.
5. Press **Enter**.

Uploading Files - FAC Edition Only

Note: To upload a large project containing many files, you can put them into a ZIP file and then specify the ZIP file when creating your project in Workbench. For more, see Creating Projects with the FAC Edition on page 12.

To upload a file from your computer:

1. Locate the file using your computer's file explorer.
   - On Windows, locate the file using File Explorer.
   - On macOS, locate the file using Finder.
2. In the Xpress Workbench file tree, locate the folder where you want to upload the file.
3. Drag the file from your computer's file explorer onto the target folder in the Xpress Workbench file tree.

   Note: In Google Chrome, you can drag an entire folder from your computer into Xpress Workbench.

Downloading Files - FAC Edition Only

You can download a single file or multiple files.

Download files in one of the following ways:

1. Download a single file or folder:
    1. Locate the file or folder you want to download in the file tree.
    2. Right-click the filename.
    3. Click **Download** on the menu that appears.
2. Download several files or folders as a zip or tar.gz archive:
    1. Use Ctrl-click to select multiple files. Select each file or folder you want to download in the file tree.
    2. When all the files are selected, right-click one of them.
    3. Click **Download** on the menu that appears.

On Windows, multi-file selections are downloaded as a zip archive.

On macOS, they are downloaded as a tar.gz archive.

Downloading the Project

Complete the following steps to download the entire project as a ZIP or `tar.gz` archive:

- Open the **File** menu and click **Download Project**.

Note: On Windows, the project is downloaded as a ZIP archive. On macOS, it is downloaded as a `tar.gz` archive.

Note: To create an app archive file that can be imported into Xpress Insight, see Exporting Xpress Insight Apps on page 73.

Viewing File Revision History - FAC Edition Only

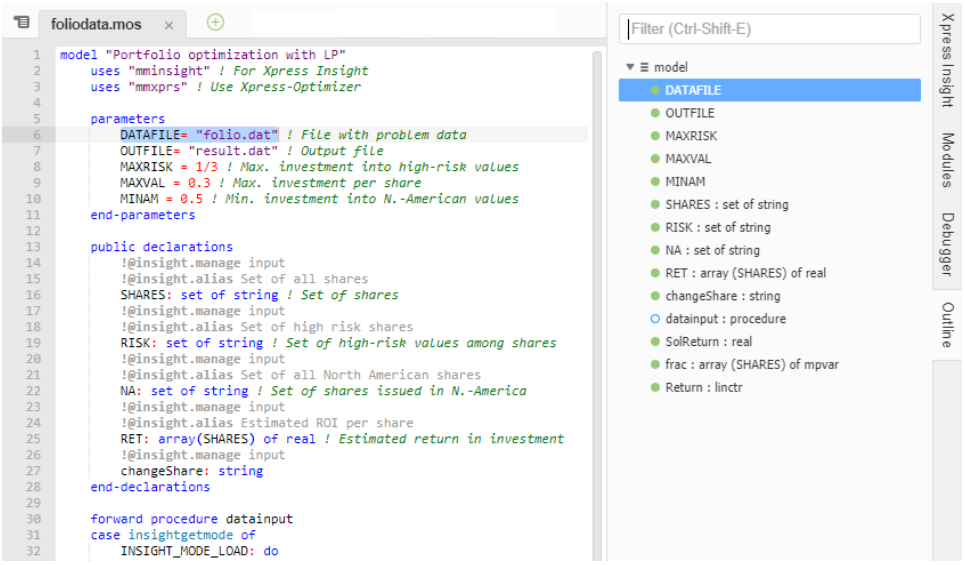
Xpress Workbench keeps a history of the edits that have been made to a file. Complete the following steps to revert the file to a previous version:

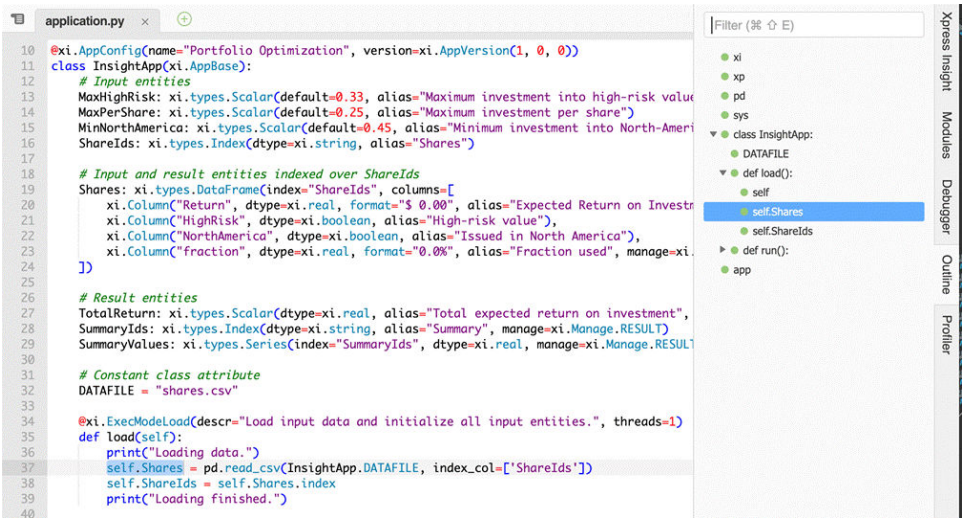
1. Open a file by double-clicking its filename.
2. Open the **File** menu and click **Show File Revision History**.
3. Drag the slider until the correct revision of the file is shown.
4. (Optional) Click the **Revert** button.

Coding Assistance

As a fully integrated IDE, Xpress Workbench offers syntax highlighting, the ability to insert templates for common functions (snippets) via a keyboard shortcut, search for symbols, and is capable of automatically completing keywords and statements.

The structure of an open Mosel or Python model is displayed in the **Outline** pane. Clicking on a displayed element will highlight the associated code in the code editor including those declared in the Mosel code as *include* files (see Jump to Definition on page 28).





For Mosel projects, the **Modules** panel in the right-hand menu displays the available Mosel modules, and the subroutines, types, and global symbols that they expose.

Additional features are explained in the following topics:

- Jump to Definition
- Finding Text
- Highlighting Syntax
- Automatically completing words
- Automatically completing statements
- Inline documentation
- Folding code

Jump to Definition

You can quickly navigate to the definition of a method or function call from the **Outline** pane or the code editor.

Use any of the folllowing options to jump to the definition:

From the **Outline** pane:

- Click the function or method call in a file, including those declared in the Mosel code as *include* files.

From the **code editor**:

- Select a symbol and press **F3**. Press *F3* immediately afterwards to return to the previous location.
- Right-click the call in the code editor and select **Jump To Definition** from the menu. Press *F3* immediately afterwards to return to the previous location.

Note: ■ With Python files, you can jump to a definition even if the variable in question is imported from another file. If the import is imported using a `from` clause (such as from my_module import x, possibly aliased using the `as` directive), the cursor jumps to the `from` clause in the current file. Selecting **Jump to Definition** from the pop-up menu on the name or alias in the `from` clause switches to the appropriate file and shows the definition there.

Finding Text

The **Find**, **Find in Files**, and **Replace** menu options open a dialog in the text editor pane that contains some common tools.

Item	Description
	Enable Regular expression mode
	Enable case sensitivity
	Match whole words only

Item	Description
	Wrap the search to the start of the document

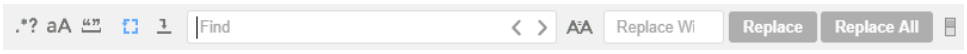
In addition, each menu option features a unique tool.

Function	Tool icon	Description
Find		Search the selected text only
Find in Files	$_>	Show the search results in the console
Replace		Attempt to match the case of the text being replaced

Complete the following steps to find a text string in a given file:

1. Open the file by double-clicking its filename.
2. Open the **Find** menu and click **Find**.
   - On Windows, you can instead press **Ctrl-F**.
   - On macOS, you can instead press **Cmd-F**.

   The **Find** box is displayed below the editor.



3. Type the text string you want to find and select options to configure the search.

   Hover over the options to see the related tool-tip.
4. Press **Enter**.
5. (Optional) To find the next occurrence click the greater-than symbol to the right of the **Find** box.
   - On Windows, you can find the next occurrence by pressing **Ctrl-K**.
   - On macOS, you can find the next occurrence by pressing **Cmd-K**.
6. (Optional) To find the previous occurrence, click the less-than symbol to the right of the **Find** box.

Replacing Text

Complete the following steps to replace one text string with another:

1. Find the text string you want to replace, by following the instructions in the previous section.
2. In the right most text box, type a replacement text string.
3. Click **Replace**.

   The currently selected occurrence is replaced, and the editor highlights the next occurrence. The currently selected occurrence is replaced, and the editor highlights the next occurrence.

4. (Optional) Repeatedly click **Replace** to continue replacing further occurrences.
5. (Optional) To replace all occurrences in the file, click **Replace All**.

Finding Text Across Many Files

Complete the following steps to search in all project files for a text string:

1. Open the **Find** menu and click **Find in Files.**
   - On Windows, you can instead press **Ctrl-Shift-F**.
   - On macOS, you can instead press **Cmd-Shift-F**.
2. In the **Find** box that appears below the editor, type the text string you want to find.
3. (Optional) In the box to the right of the find button, enter a pattern for the filenames you want to match. Several patterns can be separated by a comma, and files can be excluded by beginning a pattern with a minus sign.

   By default, all files are matched except those whose names begin with a dot.
4. Click the **Find** button.

   The **Search Results** window appears below the editor.
5. Double-click a line in the **Search Results** window to open the source file in the editor.

Replacing Text Across Many Files

To replace all occurrences of a text string in all project files with another text string:

1. Open the **Find** menu and click **Find in Files.**
   - On Windows, you can instead press **Ctrl-Shift-F**.
   - On macOS, you can instead press **Cmd-Shift-F**.
2. In the **Find** box that appears below the editor, type the text string you want to find.
3. (Optional) In the box to the right of the find button, enter a pattern for the filenames you want to match. Several patterns can be separated by a comma, and files can be excluded by beginning a pattern with a minus sign.

   By default, all files are matched except those whose names begin with a dot.
4. In the **Replace With** box, to the right of the **Find** box, type the string with which you want to replace the found text.
5. Click **Replace**.

Highlighting Syntax

If Xpress Workbench supports the language of a source file, it highlights the source code according to the syntax of the language. Supported languages include:

- Mosel
- Python

- XML
- VDL
- HTML
- JavaScript

The language of a source file is inferred by Xpress Workbench using the filename extension. Complete the following steps to override the language for the open source file:

1. Open the **View** menu and click **Syntax**.
2. Click the language you want to use for the source file.

Automatically Completing Words

When editing a source file, Xpress Workbench tries to automatically complete the word that you are typing based on words that have appeared elsewhere in the same source file or in other source files in the project.
Perform the following steps to autocomplete a word:

1. Begin typing the name of a variable or subroutine that appears elsewhere in a source file.

   Xpress Workbench displays a list of words that begin with the characters you typed. If you are editing a Mosel or Python file, this list is based on the variables and properties in the current scope.
2. Use the cursor keys to select the word that you wish to autocomplete.
3. Press **Enter** or **Tab**.

Automatically Completing Statements

When editing a source file, Xpress Workbench automatically inserts the closing tag for a typed statement. (This is less relevant in Python files, where indentation rather than closing tags are used to indicate syntactic structure.)

Complete the following steps to autocomplete a statement:

- Begin typing the name of a statement. The Xpress Workbench IDE displays a list of words that begin with the characters you typed.
- Use the cursor keys to select the statement that you wish to autocomplete.
- Press **Enter** or **Tab**—the statement is added to the Mosel file.

Tip: Press **Tab** while the modifying the statement to navigate into the next editable area in the statement.

Xpress Workbench will autocomplete the following Mosel statements:

- model/end-model
- package/end-package
- parameters/end-parameters

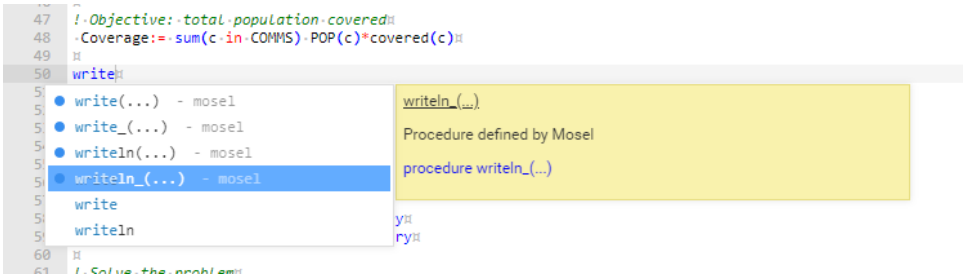
- declarations/end-declarations
- record/end-record
- requirements/end-requirements
- initializations/end-initializations
- forall do/end-do
- while do/end-do
- do/end-do
- repeat/until
- if/then/else/end-if
- case/end-case
- procedure/end-procedure
- function/end-function

Inline Documentation

When editing a source file, the Xpress Workbench IDE displays the help text for standard Mosel library subroutines, and for certain Python libraries used for solving a model (in particular, those for Xpress Insight).

In an open Mosel file, when you begin typing one of the available subroutines, the autocomplete dialog will display all the different variations of that subroutine. If you highlight one of the options in this window, either by moving the keyboard cursor or hovering the mouse over an option, the inline documentation window will display any inline help text for the highlighted item.

Note: If you highlight an item with the keyboard and hover the mouse, the mouse hover has priority.



Folding Code

In supported source file types, code blocks in the file can be folded into a single line, hiding the contents of the block and making the file easier to read.

Folding Code with the Mouse

Complete the following steps to fold or unfold a code block in a Mosel source file:

1. Hover the mouse in the narrow grey area where the line numbers for the source file are displayed, and look for the small arrows which appear to the right of the line numbers.
2. Click an arrow to fold or unfold the code block that begins on that line.

Folding Code with the Keyboard

Complete the following steps to fold and unfold code blocks using the keyboard:

1. Position the cursor anywhere within a code block that you want to fold.
2. Press **F2**
3. To unfold a code block, position the cursor anywhere on a folded line, and press **F2** again.

Formatting Code

You can reformat Python code according to a set of consistent layout rules.

- To reformat a Python file, click the **Edit** menu and select from the **Formatting** options.

  Note that the entire Python file is formatted, irrespective of how much code is selected in the editor. This is because a selection might not be syntactically valid on its own, so the entire file is required for context.

  The Python formatter respects the project settings regarding the use of tabs or spaces in a file. It does not currently have any other configuration options.

  Tip: If you are working on the same Python file in another IDE besides Xpress Workbench, be aware that the other IDE might have different formatting rules, or its rules might be configured differently. To avoid making unnecessary cosmetic changes that might make it harder to compare code versions, use only one IDE consistently for formatting.

Collaborating with Other Users - FAC Edition Only

Xpress Workbench is a multi-user application with features for collaborating with other users.

When another user opens a project that you have open in your browser, a notification icon appears in the top right of the screen showing you their name. If this user opens a file that you also have open in the project, the position of their cursor within the file is shown on your screen. If the user edits the file, their edits appear on your screen in real time. Xpress Workbench highlights the lines of the file that were edited by each user who is collaborating on the project.

Note: In this version of **Xpress Workbench**, all projects are public and can be opened and edited by any other user who has an account on the server.

Sending Messages to Users

To send a message to another user who has the same project open:

1. Open the **Collaborate** sidebar by clicking its name.
2. Type your message into the text box at the bottom of the sidebar.
3. Press **Enter** to send the message.

   All other users who have the project open are notified with the message.

   When a user opens the **Collaborate** sidebar, they see a history of the messages that were sent by all users of the project.

CHAPTER 5

Running and Debugging Models

Xpress Workbench allows you to run models in a standalone environment. Mosel models are compiled, loaded, and run when they are executed in Workbench. When a Mosel model is compiled, the syntax of the model and the libraries implemented in Mosel are checked but no operation is executed. The result of the compilation is a BInary Model (BIM) that is saved in a second file. In this form, the model is ready to be executed and the source file is not required any more. To actually 'run' the model, the BIM file must be read in again by Mosel and then executed.

Running a Mosel Xpress Insight app in Workbench, outside of Xpress Insight, can be faster and simpler than publishing the complete app to an Xpress Insight server whenever you want to test it. When you run the Mosel model in Workbench, it is also re-compiled if the source has changed.

Running Models

Complete the following steps to run a model in Xpress Workbench:

1. In the **Run Configurations** drop-down list, select the source file that you want to run.

   The **Run Configurations** drop-down list is located to the right of the main menu bar, near the top of the screen.

2. Click the  **Run** button or press Ctrl+F5.

   Note: Only source files that declare a Mosel model can be executed. Source files that declare a Mosel package cannot be run directly; instead, run the main Mosel source file, which begins and ends with the **model** and **end-model** keywords.

3. Check the **Output** window that opens at the bottom of the screen to see the output printed by the model. If the model expects some input, you can type in the output window, too.

   Xpress Workbench compiles the source file and any packages that it imports before running the model. Any syntax errors are highlighted in the code window, and compilation errors are displayed in the **Output** window. Packages can be compiled but not run, as a debugging step.

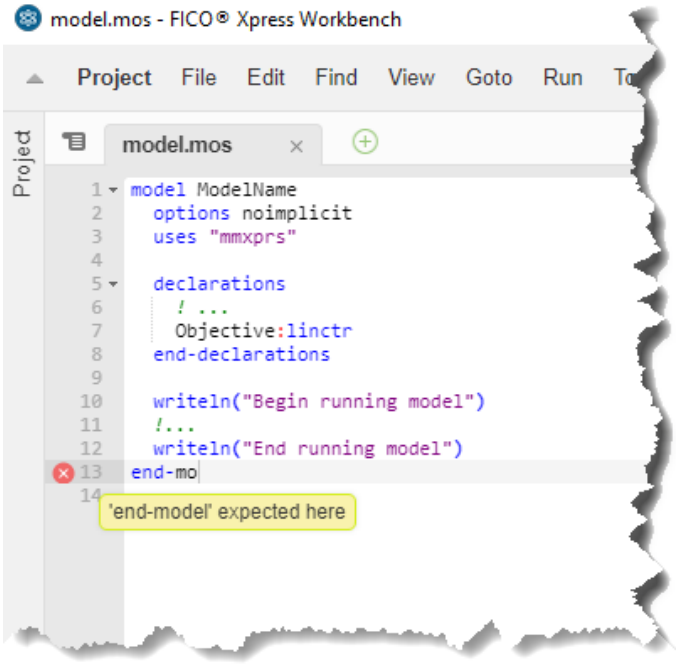


Figure 2: Highlighted syntax error

Note: If you are running a model on DMP, you might see the following message in the **Output** window:

```
warning  libmamba 'root_prefix' set with default value:
```

This message can safely be ignored.

Stopping a Model

A model that is executing in Mosel can be stopped at any time. Complete the following steps to stop a model:

1. Click the **Stop** button in the **Output** window.
2. Wait for the model to finish executing.

Note: If a model is being debugged, this stops the debug session immediately. You cannot review the final state of the model before it exits.

Changing the Working Directory

By default, the working directory of the running Mosel model is the directory containing the source file. If the model reads any input files or writes any output files, they are read from or written to this directory.
Complete the following steps to change the working directory:

1. Run the model by following the instructions above.
2. Either wait for the model to finish executing or click the **Stop** button to interrupt it.

3. When the model has finished executing, click the **CWD** button in the **Output** window.
4. Choose the working directory for the model and click **Select**.
5. Run the model again. This time it runs in the newly selected working directory.

   Note: If the project contains a top-level directory named **model_resources**, the working directory is set to this directory by default. This makes the runtime environment for an Xpress Insight app similar to how it is when the app runs on an Xpress Insight server, when Xpress Insight copies the contents of the **model_resources** directory into the working directory.

Debugging Models

Xpress Insight provides a fully-featured Mosel debugger. The debugger allows you to:

- Set and clear breakpoints.
- Step over the next program statement.
- Step into a subroutine call at the current statement.
- Step out of the current subroutine call.
- View the current call stack.
- View the values of local variables within the current subroutine call.
- View the values of global variables in the main model.
- View the values of global variables defined in each package.
- Evaluate expressions as you step through the program.

Starting an Xpress Insight Debug Session

Ensure that:

- You have installed Python.
- You have installed the debugpy, xpress, and xpressinsight modules from pip or conda.
- You have set the worker environment variables geared towards your Python installation from PyPI or conda as outlined in the Xpress Insight Python module documentation.
  - Key elements are around setting MOSEL_RESTR and PYTHON_EXE as well as other environment variables in the Insight worker's application.properties file.

Complete the following steps to run an Insight scenario in debug mode:

1 Open the Xpress Insight pane in Xpress Workbench and right-click on a scenario. Selection the execution mode (LOAD or RUN) you want to debug.

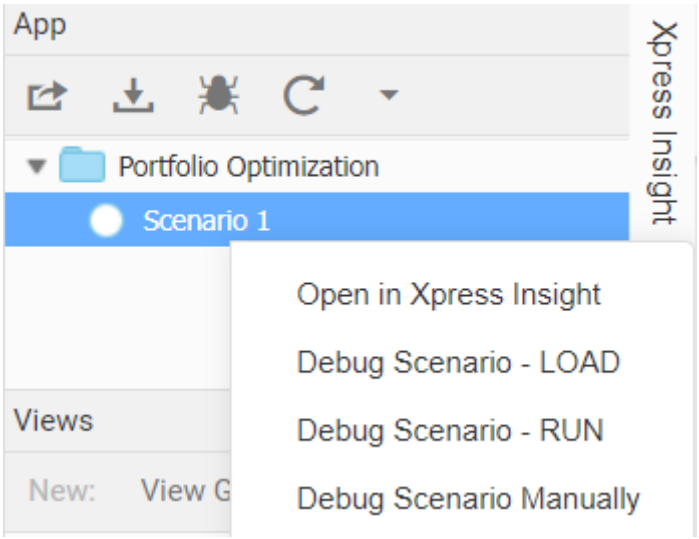


Figure 3:

When selecting **Debug Scenario Manually**, you must either click on **Click to Open Xpress Insight** in the green banner, or open Xpress Insight directly and execute the scenario manually by clicking a button for any custom execution mode, or by using the built-in LOAD and RUN options from the scenario pill drop-down.

2 During debugging, the **Debugger** sidebar opens if breakpoints have been set.

See the next section for how to set breakpoints.

3 When the model has finished executing, the debugger exits and the **Debugger** sidebar closes.

Note: Once the debugger has exited, you can no longer see the values of any model variables. Make sure you have set breakpoints accordingly. This is in contrast to the Mosel debugger, which automatically suspends just before exit.

Setting and Removing Breakpoints

Breakpoints are set by clicking in the breakpoint area, which is the narrow grey area where the line numbers for the source file are displayed. When you hover the mouse to the left of the line numbers, the cursor changes to a hand icon to show that clicking in that spot sets a breakpoint.

1 To set a breakpoint, click in the breakpoint area to the left of the line where you want to set the breakpoint.

A red disk appears to show that a breakpoint was set.

2 To remove a breakpoint, click on the red disc icon in the breakpoint area to the left of the line where you want to remove the breakpoint.

The red disc icon disappears from the breakpoint area.

3. To disable a breakpoint, open the **Debugger** sidebar if it is not already open, by clicking on its name and in the breakpoints section, clear the checkbox for the breakpoint that you want to disable.

   The debugger no longer suspends at this breakpoint, until you enable it again.

4. To disable all breakpoints in the model, open the **Debugger** sidebar if it is not already open, by clicking on its name and click the **Toggle Breakpoints** button, which is the fifth button from the left in the **Debugger** sidebar.

   Note: Conditional breakpoints are not supported in this version of Xpress Workbench.

   Note: Breaking automatically on exceptions is not supported in this version of Xpress Workbench.

Suspending and Resuming the Model

The model can be suspended at any time while it is executing, allowing you to inspect its state. Complete the following steps to suspend the model:

1. Open the **Debugger** sidebar, if it is not already open, by clicking on its name.
2. Click the **Pause** button, which is the first button from the left in the **Debugger** sidebar.

   While the model is executing, this button has a pause icon.
3. To resume execution, open the **Debugger** sidebar, if it is not already open, by clicking on its name.
4. Click the **Resume** button, which is the first button from the left in the **Debugger** sidebar.

   Alternatively, press `F8`.

   While the model is suspended, this button has a **play** icon.

Stepping through the Model

While the model is suspended, you can step through to the next statement, depending on which action you choose.

1. Open the **Debugger** sidebar if it is not already open, by clicking on its name.
2. Step through a model using any of the following steps:
   - To step to the next statement in the current subroutine, click the **Step Over** button, which is the second button from the left in the **Debugger** sidebar.

     Alternatively, press `F10`.
   - If the current statement contains a call to a subroutine, you can step into the subroutine. The debugger suspends at the first statement of the subroutine. Click the **Step Into** button, which is the third button from the left in the **Debugger** sidebar.

     Alternatively, press `F11`.
   - If the current statement is within a subroutine, you can step out of the subroutine. The debugger suspends at the statement after the calling statement.

Click the **Step Out** button, which is the fourth button from the left in the **Debugger** sidebar.
Alternatively, press `Shift-F11`.

Examining Variables

While the model is suspended, you can inspect its data using the **Variables** section in the **Debugger** sidebar.

The variables section contains the following entries:

- An entry for each local variable defined in the current subroutine
  If the current statement is not within a subroutine, no local variables are available.
- Global symbols
  Expand this entry to see variables that are declared at the top level of the model.
- Parameters
  Expand this entry to see runtime parameters defined in the model.
- Constants
  Expand this entry to see any constant declarations in the model.
- An entry for each namespace defined in the model
  Expand these entries to see the global variables defined in each namespace.

The **Variables** section shows the name, type, and value of each variable. If a variable holds structured data, such as a set, array, list, or user-defined record, the entry for that variable can be expanded. Complete the following steps to expand a variable:

1. Click the arrow to the left of its name.
   The entries within the data structure appears.
2. Continue to expand the entries until you see the data that you require.

Note: You can also see the value of a variable by hovering the mouse over a reference to the variable in the active source file.

Examining Arrays

When inspecting array data in the debugger, Workbench shows a maximum of 20 array entries in the variables tree.

Complete the following steps to view the full contents of the array in a table:

1. While suspended at a breakpoint, open the **Debugger** sidebar.
2. In the **Variables** tree, find the array you want to inspect and right-click its name.
   Sets and lists can also be inspected in tables.
3. In the menu that appears, click **Inspect in table**.
   A new tab opens containing a table. The table has a row for every entry in the array, a column for each index set of the array, and a column for the array values.

4 You can inspect the table in the following ways:

- Click a column header to sort the table by that column.
- Filter the table by entering a filter value in one or more of the filter cells, which appear at the top of the table directly under the column headings.
- To inspect the array elements in more detail, hover the mouse over the table cell until a tooltip appears.
  - If the table cell contains a record, you can expand the tooltip to show its member values.
  - If the table cell contains a nested array, you can expand the tooltip to show its elements.

Note: You can increase the number of array elements displayed in the variables tree by changing the **Maximum Collection Items to Show in Debugger** setting in your user preferences.

Note: Arrays with more 10,000 entries cannot be filtered or sorted for performance reasons. To sort or filter large arrays you must save the data as a CSV file and open it in another application. The limit can be increased by changing the **Maximum Collection Table with Sorting/Filtering** setting in your user preferences, but this may have a significant impact on the performance of the application.

Saving Array Data as CSV Files

Array data can be saved as a CSV file so that it can be analyzed using other applications.

To export an array as a CSV file:

1 Open the array in a table.
See Examining Arrays on page 40.

2 Click **Save as CSV**.

3 Choose where you would like to save the CSV file and click **Save**.
If you are using the FICO Analytic Cloud edition of Workbench, your browser may download the file automatically without prompting you to choose where to save it.

Evaluating Expressions

In the **Watch Expressions** section of the **Debugger** sidebar, you can evaluate any number of variables and expressions in the context of the current subroutine.

The values are updated each time you step through the program.
You can add or remove watch expressions.

1 Open the **Debugger** sidebar, if it is not already open, by clicking on its name.

2. Complete the following steps to add a watch expression while the debugger is suspended:
   a. Double-click on the text **Type your expression here** in the **Watch Expressions** section.
   b. Enter the name of a variable or an expression involving one or more variables.
   c. Press **Enter**.

   Tip: The pop-up menu options that are available in the **Variables** tree view are also available for evaluated expressions.
3. Complete the following steps to remove a watch expression:
   a. Right-click the watch expression you want to remove.
   b. Click **Remove Watch Expression** on the menu that appears.

Compiling Models

Complete the following steps to compile a Mosel source file to a BIM file:

1. From the **Run Configurations** drop-down list (located to the right of the main menu bar, near the top of the screen), select the Mosel source file that you want to compile.
2. Click the **Compile** button, which is the button directly to the right of the **Run Configurations** drop-down list.
   Alternatively, press `Ctrl-B`.
3. Check the **Output** window for any compilation errors.
   If compilation is successful, a BIM file appears with the source file in the file tree.

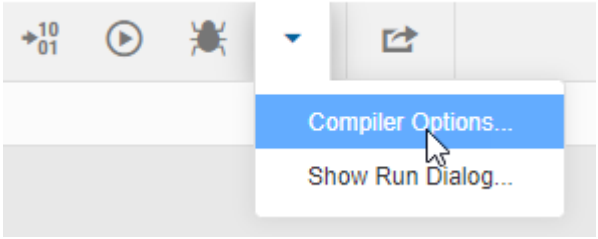
Note: If the source file loads any packages, Xpress Workbench compiles the packages before compiling the selected source file. Intermediate BIM files are placed in a folder named `out` by default.

Configuring Compiler Options

You can change the options that are used when compiling source files.

To view or change the compiler options, follow these steps:

1. In the toolbar, click the **Tools** icon and then select **Compiler Options** from the drop-down menu.



2 In the **Compiler Options** window, configure the options that should be used when compiling source files.

Note: For more information about Mosel compiler options, see the Mosel Reference Manual.

You can configure the following settings:

BIM File Output Folder — The folder within the project where intermediate BIM files are stored during compilation.

Custom Python Path — The full path to the Python environment to use for this project. If you do not specify a custom path, the system default Python environment is used.

Tip: You can use a custom Python path, along with custom arguments, to select a Python runtime environment with the `conda` package. For more information, see Choosing a Python Environment Using Conda on page 44.

Custom Python Arguments — Any command-line arguments you want to pass to the Python interpreter (for example, to set optimization or debug levels). Keep in mind that Xpress Workbench will append additional arguments to the command string at run time.

Debugging Information — The information to include in the generated BIM file. Select one of the following options:

- **Remove private object names**: The names of private variables, constraints, and subroutines are not saved to the BIM file.
- **Add debugging information**: The `-g` command line flag is passed to the Mosel compiler. Use this option to locate a runtime error.
- **Add debugging and tracing information**:
  The `-G` command line flag is passed to the Mosel compiler. This option is required to run the model with the debugger.

By default, private declarations in Mosel are stripped on compilation. Declarations are private unless explicitly prefixed with the `public` keyword. Any entities in your Mosel code that need to be visible to Xpress Insight must be declared public.

Implicit Declarations	How to handle symbols that are not explicitly declared in a `declarations` block. Select one of the following options: ■ **Allow implicit declarations**: No command line flags are passed to the Mosel compiler. Implicit declarations are silently accepted. ■ **Forbid implicit declarations**: The `-ni` command line flag is passed to the Mosel compiler. Implicit declarations cause the compilation to fail with an error. ■ **Warn about implicit declarations**: The `-wi` command line flag is passed to the Mosel compiler. Implicit declarations are accepted, but the compiler prints a warning.
Treat Warnings as Errors	Whether to pass the `-we` command line flag to the Mosel compiler. Select this option if you want the compiler to treat warnings as errors, such that any warning causes the compilation to fail.
Generate Doc Annotations	Whether to pass the `-D` command line flag to the Mosel compiler. Select this option if you want to generate documentation annotations in the resulting BIM file. (By default, documentation annotations are ignored.)
Enable Xbim Extension	Whether to pass the `-I` command line flag to the Mosel compiler. Select this option if you want to enable the xbim extension, which causes additional symbol information (such as array index names) to be stored in the generated BIM file.

Note: When you run a model from Xpress Workbench, the `-g` option is automatically added so that stack traces can be displayed if an error occurs. When you debug a model from Xpress Workbench, the `-G` option is automatically added.

Choosing a Python Environment Using Conda

You can specify a Python runtime environment using the `conda` package.

If you use `conda` to manage Python packages and environments, you can configure Xpress to select a runtime `conda` environment for your project.

To do this, configure the compiler options so the Python command string invokes the `conda` executable instead of invoking the Python interpreter directly, and also specifies the environment you want to use:

1. Open the compiler options. (For more information, see Configuring Compiler Options on page 42.)
2. In the **Custom Python Path** field, type `$CONDA_EXE`. (You can also specify the full path to `conda.exe`.)

3. In the **Custom Python Arguments** field, type the following arguments:

   ```
   run -n env python
   ```

   where *env* is the name of the environment you want to use.

4. (Optional) If there are additional arguments you want to pass to the Python interpreter, type them after the `python` command in the **Custom Python Arguments** field.

Running Submodels and Parallel Solving

Xpress Workbench supports running Mosel models that load and run submodels. Before your master model can load and run a submodel, you must ensure that the submodel has been compiled. Complete the following steps to run a model that uses submodels:

1. Open the submodel source file by double-clicking its filename.
2. Open the **Run** menu and click **Build**.

   If compilation is successful, a BIM file appears alongside the source file in the file tree.
3. Ensure that the submodel BIM file is located in the working directory of the master model.
4. Select the master model source file in the **Run Configurations** drop-down list.
5. Click the **Run** button, which is the second button to the right of the **Run Configurations** drop-down list.
6. If a submodel encounters a runtime error, the master model will be suspended on whichever line was being executed when the error occurred. Click **Resume** to continue executing the master model.

   If you change the submodel source file, make sure you recompile it before running the master model.

   - **Note:** Manually compiling the submodel is not needed if your master model compiles it programmatically using the **compile** function in Mosel.
   - **Note:** You cannot debug a submodel while it is running within the master model. Breakpoints set in submodel source files are ignored. To debug a submodel in standalone mode, select the main source file for the submodel in the **Run Configurations** drop-down list and run it as usual.
   - **Note:** (FAC edition only) Remote instances of Mosel are not supported. Submodels must be loaded and run in the main Mosel process.

Preparing Template Files for Mosel Deployment Wrappers

It is straightforward to execute Mosel models from applications written in C, C#, Java, VB.NET and VBA.

To assist with this, Xpress Workbench contains—for each of those languages—template source files that can load and execute a pre–compiled Mosel model.

To deploy a Mosel model in an application:

1. Compile your Mosel model in Xpress Workbench.
2. Open the **File** menu, click **New** then choose the appropriate Mosel deployment file template.
3. Find the constant with the value `test.bim` near the top of the file and change its value to the name of your BIM file.
4. Save the file then copy this source file and your BIM file into your project in Microsoft Visual Studio, Eclipse IDE or other development environment.

   Note: Xpress Workbench cannot compile or run source files written in C, C#, Java, VB.NET or VBA. To compile and run the generated source file, use an IDE tool which supports these languages, such as those listed above.

For more information on how to interact with Mosel from C, C#, Java, VB.NET and VBA, see the *Mosel User Guide* and the *Mosel Libraries Reference Manual*.

Tuning Mosel Models

For a given optimization problem, setting suitable control parameters frequently results in improved performance such as solution time reduction. The FICO® Xpress built-in tuner can help a user to identify such a set of control settings that allows the Xpress solver to solve problems faster than by using defaults.

See section *Using the Tuner* in the Xpress Optimizer Reference Manual for more information on how to tune optimization problems.

Starting a Tuning Session

Complete the following steps to start tuning a Mosel model:

1. In the **Run Configurations** drop-down list, select the Mosel source file that you want to tune.
2. Click the **Run Tools** button (the fourth button to the right of the **Run Configurations** drop-down list).
3. In the menu that appears, click **Show Run Dialog**.
4. In the **Run Model** dialog, ensure that the **Run Mode** field is set to **Tune the Model** and configure the settings for the tuning session.
   See Tuner Settings on page 48 for more information.

5 Click **Run**.

The tuner logs from will appear in an **Output** window at the bottom of the screen.

6 When you see the message **New tuner results** at the top of the screen, click the link.

A new tab will open, displaying the tuner results in a table.

Note: Tuner result files and log files are written to a directory named `tuneroutput` within the project. You can view the tuner results table for a previous tuner session by locating the XTR file in the Xpress Workbench file tree and double-clicking it.

The columns of the tuner results table are as follows:

Column	Description
Problem	The name of the tuning session, as specified in the **Tune Model** dialog.
Type	The type of problem that was solved, e.g. MIP.
Xpress Version	The version of the Xpress that was used for the tuning session.
Timestamp	The date and time of each tuner run, in the form `yyyymmdd.hhmmsss.micros`.
Logs	Click in this cell to open the Xpress log file for the tuner run.
Tuner Threads	The number of times the tuner solved the problem as part of each run, including permutations.
one or more columns...	A column for each Xpress control which the tuner tried during the session.
Code	Click this cell to see the Mosel code which sets the Xpress controls as they were for this tuner run.
Status	The status of the tune run, e.g. whether or not the problem was solved within the time limit.
Run Time	The total time taken to solve the problem.
Objective	The value of the objective function at the end of the tuner run.

For MIP problems, the following columns are also present:

Column	Description
Bound	The best bound at the end of the tuner run.
Gap	The gap at the end of the tuner run, as a percentage.
Integral	The integral at the end of the tuner run.
Solutions	The number of solutions found during the tuner run.
Nodes	The number of tree nodes searched during the tuner run.

Tuner Settings

In the **Run Model** dialog box, you can configure the following settings that affect the tuner's behavior:

- **Session Name**: A name for the tuning session. It will determine the directory where tuner result files are written. The name will also appear in the tuner table. If you plan to combine the data from several tuner sessions, the name can serve to differentiate the results from different sessions.
- **Method File**: A path to a tuner method file. A tuner method consists of a list of controls for the tuner to try with. You can create a template tuner method file by opening the **File** menu, clicking **New** and choosing **Tuner Method**.
- **Tuner Time Limit**: The maximum time in seconds to spend tuning the model. If set to zero, no time limit will be imposed.
- **Permutations**: The number of times that the problem will be randomly permuted and solved while tuning it. This can improve stability by avoiding lucky controls that show improved performance by coincidence rather than structural reasons. See section *Tuner with Problem Permutations* in the Xpress Optimizer Reference Manual.
- **Tuner Threads**: The number of different control settings that will be run simultaneously.

Manually Enabling the Tuner

In some cases, the **Run Model** dialog is not sufficient, and you must modify the source code of your model to enable the tuner:

- If your model executes a submodel using Mosel's `run` subroutine, the tuner will not be enabled for the submodel.
- If your model solves more than one optimization problem, the tuner will be enabled for all problems defined in the model, but the custom tuner settings specified in the **Run Model** dialog will only apply to the top-level problem.

There are two ways to manually enable the tuner. The first is to pass the `XPRS_TUNE` option to the `minimize` or `maximize` call. For example, to tune a submodel when it is executed from a master model, the submodel could contain the following code:

```
minimize(XPRS_TUNE, Objective)
```

The second way is to set the `xprs_tunermode` parameter:

```
setparam("xprs_tunermode", 1)
minimize(Objective)
```

If you want to tune the main problem in your Mosel model, but disable the tuner for the subproblems, you can set `xprs_tunermode` to zero when solving the subproblem, and then set it to 1 when solving the main problem.

```
with subprob do
  setparam("xprs_tunermode", 0) ! Disable the tuner
  minimize(SecondaryObjective)
done
setparam("xprs_tunermode", 1) ! Enable the tuner
minimize(PrimaryObjective)
```

To customize the tuner settings when tuning a subproblem, the relevant parameters must be set inside the with block which defines the subproblem. For example:

```
with subprob do
  setparam("xprs_tunermethodfile", "tunermethod.xtm")
  setparam("xprs_tunerthreads", 4)
  minimize(SecondaryObjective)
done
```

For more information about parameters that affect the tuner's behavior, see *Using the Tuner* in the Optimizer Reference Manual. See the Mosel Language Reference Manual for more information about working with multiple problems in Mosel.

Profiling Models

Running a model with the profiler provides detailed information on what part of the code is actually executed and how much time each statement requires. This information can be helpful for optimizing the model (by locating hot spots where the code is using a great deal of computer time) and also for building test suites (by checking whether the data sets used in the test set exercise all statements of a given model).

Starting a Profiling Session

Complete the following steps to start profiling a model:

1. In the **Run Configurations** drop-down list, select the Mosel or Python source file that you want to profile.
2. Click **Run Tools** (the fourth button to the right of the **Run Configurations** drop-down list).
3. In the menu that appears, click **Show Run Dialog**.
4. In the **Run Model** dialog box, set the **Run Mode** field to **Profile the Model**.
5. Click **Run**.

   When the model has finished running, the **Profiler** tab opens, showing the profiling results in a table. The results are also written to a file in the same directory as the source file; this file has the same name as the source file, with the extension `.prof`.

   Note: If a Mosel model loads multiple packages, a separate `.prof` file is written for each package. For Python models, only a single `.prof` file is written.

   The columns of the table on the **Profiler** tab differ depending on the language of the source file.

   - **Mosel source files:**
     - **Line #**: The line number of the statement in the source file.
     - **Count**: The number of times the corresponding statement has been executed.

  - **Time (s)**: The total amount of time (in seconds) spent on this particular line (this measure is not valid if the statement is a recursive call).
  - **Latest (s)**: The elapsed time (in seconds) between the beginning of the execution and the last time the line was executed (the line number, then the first three columns before the Mosel source).

- **Python source files:**
  - **Line #**: The line number of the statement in the source file.
  - **Name**: The name of the function that was executed.
  - **Count**: The total number of times the executed function was called.
  - **Total Time (s)**: The total amount of time (in seconds) spent executing the function.
  - **Average Time (s)**: The average amount of time (in seconds) spent executing the function on each call.
  - **File**: The file or module name of the executed function.
  - **Thread**: The name of the thread executing the function.

You can double-click a row in the table jump to the line in the source file.

Note: If the source has been updated since the `.prof` file was created, it might take longer to go to the correct line.

You can sort the profiler table by clicking on the header of the column you want to sort by.

Collecting Coverage Data

The profiler can also collect coverage data for Mosel models. The difference between this and profiling the model is that the generated files have the extension `.cov`, and they only indicate the number of times each statement has been executed.

Additionally, existing `.cov` files are updated instead of being replaced, such that iteration counts of each statement are added up.

After execution, the total execution time and source file coverage information is displayed in the output pane and an annotated version of each source file is written containing the number of times each statement has been executed. The coverage logs will tell you whether two executions of a model take the same path through the source code, independent of the optimization results which might vary if the solver version changes.

Complete the following steps to collect coverage data for a Mosel model:

1. In the **Run Configurations** drop-down list, select the Mosel source file for which you want to collect coverage data.
2. Click **Run Tools**, the fourth button to the right of the **Run Configurations** drop-down list.
3. In the menu that appears, click **Show Run Dialog**.
4. In the **Run Model** dialog box, set the **Run Mode** field to **Collect Coverage Data**.

5 Click **Run**.

Tracing Model Execution

Mosel can execute a model in trace mode. When executing in trace mode, the line number of each executed line is written to a trace file named `sourcefile-timestamp.trac`. This feature can be used to identify the point of failure if a model terminates unexpectedly. Trace mode output can be controlled via annotations, for more see Running Mosel.

Complete the following steps to execute a Mosel model in trace mode:

1 In the **Run Configurations** drop-down list, select the **Mosel source file** for which you want to collect coverage data.

2 Click **Run Tools**, the fourth button to the right of the **Run Configurations** drop-down list.

3 In the menu that appears, click **Show Run Dialog**.

4 In the **Run Model** dialog box, set the **Run Mode** field to **Trace Model Execution**.

5 Click **Run**.

CHAPTER 6

Debugging Python Models

The debugger allows you to:

- Set and clear breakpoints.
- Set conditional breakpoints.
- Step over the next program statement.
- Step into a subroutine call at the current statement.
- Step out of the current subroutine call.
- View the current call stack and navigate through it.
- View the values of local variables within the current subroutine call.
- View the values of global variables in the current model.
- View the values of global variables defined in each Python module.
- Evaluate expressions via a new input field above the Output pane as you step through the program.

Note:

- You must have an Xpress Insight project to debug Python models.
- The debugging inspection tools do not currently support full traversal of object hierarchies.
- Objects like pandas DataFrames have a default textual representation showing the data in a tabular format.
- Data can be viewed as static text, in contrast to Mosel's live Inspect In Table option.
- As with Mosel, the Python debugger does not support editing the source during debugging.
- Only single-threaded debugging is currently supported. For example, debugging the sklearn module's multithreaded cross_validate method may lead to the debug process freezing.

Starting a Python Debug Session

Ensure that:

- You have an Xpress Insight project.
- You have installed Python.
- You have installed the debugpy, xpress, and xpressinsight modules from pip or conda.
- You have set the worker environment variables geared towards your Python installation from PyPI or conda as outlined in the Xpress Insight Python module documentation.
  - Key elements are around setting MOSEL_RESTR and PYTHON_EXE as well as other environment variables in the Insight worker's application.properties file.

Complete the following steps to run an Insight scenario in debug mode:

1. In the **Run Configurations** dropdown list, select the Python source file to run.
2. Add any breakpoints to lines by clicking next to the line number. A red circle should appear showing that the line has a breakpoint.
3. Click the **Debug** button.

   The model will run, and a debugger sidebar opens.

   Note: Once the debugger has exited, you can no longer see the values of any model variables. Make sure you have set breakpoints accordingly. This is in contrast to the Mosel debugger, which automatically suspends just before exit.

Setting and Removing Breakpoints

Breakpoints are set by clicking in the breakpoint area, which is the narrow grey area where the line numbers for the source file are displayed. When you hover the mouse to the left of the line numbers, the cursor changes to a hand icon to show that clicking in that spot sets a breakpoint.

1. To set a breakpoint, click in the breakpoint area to the left of the line where you want to set the breakpoint.

   A red disk appears to show that a breakpoint was set.
2. To set a conditional breakpoint, right-click in the breakpoint area to the left of the line and select **Set Condition**.

   A blue disk appears to show that a conditional breakpoint was set.
3. To remove a breakpoint, click on the red disc icon in the breakpoint area to the left of the line where you want to remove the breakpoint.

   The red disk icon disappears from the breakpoint area.
4. To disable a breakpoint, open the **Debugger** sidebar if it is not already open, by clicking on its name and in the breakpoints section, clear the checkbox for the breakpoint that you want to disable.

   The debugger no longer suspends at this breakpoint, until you enable it again.

5 To disable all breakpoints in the model, open the **Debugger** sidebar if it is not already open, by clicking on its name and click the **Toggle Breakpoints** button, which is the fifth button from the left in the **Debugger** sidebar.

Note: Breaking automatically on exceptions is not supported in this version of Xpress Workbench.

Suspending and Resuming the Model

The model can be suspended at any time while it is executing, allowing you to inspect its state. Complete the following steps to suspend the model:

1 Open the **Debugger** sidebar, if it is not already open, by clicking on its name.

2 Click the **Pause** button, which is the first button from the left in the **Debugger** sidebar.

While the model is executing, this button has a pause icon.

3 To resume execution, open the **Debugger** sidebar, if it is not already open, by clicking on its name.

4 Click the **Resume** button, which is the first button from the left in the **Debugger** sidebar.

Alternatively, press `F8`.

While the model is suspended, this button has a **play** icon.

Stepping through the Model

While the model is suspended, you can step through to the next statement, depending on which action you choose.

1 Open the **Debugger** sidebar if it is not already open, by clicking on its name.

2 Step through a model using any of the following steps:

- To step to the next statement in the current subroutine, click the **Step Over** button, which is the second button from the left in the **Debugger** sidebar.

  Alternatively, press `F10`.

- If the current statement contains a call to a subroutine, you can step into the subroutine. The debugger suspends at the first statement of the subroutine. Click the **Step Into** button, which is the third button from the left in the **Debugger** sidebar.

  Alternatively, press `F11`.

- If the current statement is within a subroutine, you can step out of the subroutine. The debugger suspends at the statement after the calling statement. Click the **Step Out** button, which is the fourth button from the left in the **Debugger** sidebar.

  Alternatively, press `Shift-F11`.

Examining Variables

While the model is suspended, you can inspect its data using the **Variables** section in the **Debugger** sidebar.

The variables section contains the following entries:

- An entry for each local variable defined in the current subroutine - by navigating through the call stack, the local variables will adapt accordingly
  - If the current statement is not within a subroutine, no local variables are available. Any variables in nested scopes will be listed as local variables if they are used in the local scope.
- Global symbols
  Expand this entry to see variables that are declared at the top level of the module.
- If there is a local variable called 'self', for example the Insight app object, then this object's properties can be inspected.

The **Variables** section shows the name, type, and value of each variable. For structured Python entities, a string representation similar to output of 'repr()' function is shown. For some entities of the xpressinsight Python package with tabular data such as in DataFrames, the variables can be expanded.

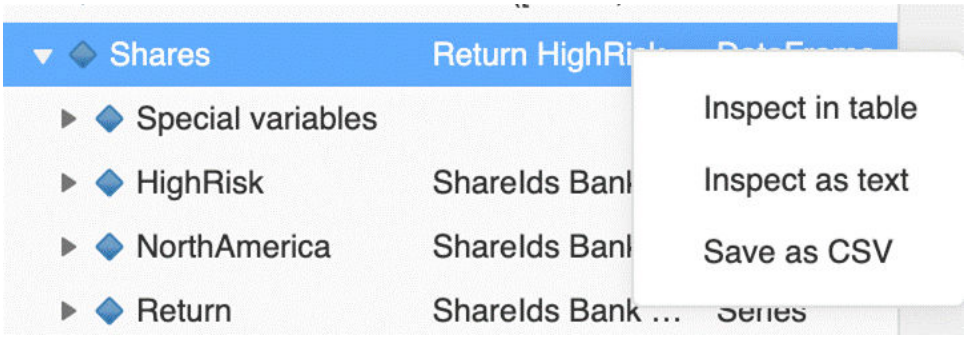
- ▶ To expand a variable, click the arrow to the left of its name.
  The entries within the data structure appears.

Examining Tabular Data

When inspecting tabular data, such as DataFrames, in the debugger, Workbench shows a maximum of 20 array entries in the variables tree.

Complete the following steps to view the full contents of the array in a table:

1. While suspended at a breakpoint, open the **Debugger** sidebar.
2. In the **Variables** tree, find the variable you want to inspect and expand it to see its properties.
3. Right-click on the variable to see the available actions.



4. Select the option you want to use to inspect the variable:
   - Select **Inspect in table** to see the contents of the variable in a table, in the same manner as for a Mosel set or array.

     Note: This option is available only for types that Xpress Workbench recognizes as tabular in nature (including built-in Python sequence types and the types used to represent entities in Xpress Insight models).
   - Select **Inspect as text** to see the textual representation of the variable, as determined by the Python `repr()` function.

     Tip: Some types (such as `pandas.DataFrame`) have a textual representation that is a user-friendly tabular view. Depending on the type and the data, this view might be more informative than the **Inspect as Table** view.
   - Select **Save as CSV** saves the data layout from the **Inspect in Table** view as a comma-separated value (`.csv`) file. This option is available only for types for which CSV representation is supported.

Evaluating Expressions

In the **Watch Expressions** section of the **Debugger** sidebar, you can evaluate any number of variables and expressions in the context of the current subroutine.

The values are updated each time you step through the program.
You can add or remove watch expressions.

1. Open the **Debugger** sidebar, if it is not already open, by clicking on its name.
2. Complete the following steps to add a watch expression while the debugger is suspended:
   a. Double-click on the text **Type your expression here** in the **Watch Expressions** section.
   b. Enter the name of a variable or an expression involving one or more variables.
   c. Press **Enter**.

      Tip: The pop-up menu options that are available in the **Variables** tree view are also available for evaluated expressions.
3. Complete the following steps to remove a watch expression:
   a. Right-click the watch expression you want to remove.
   b. Click **Remove Watch Expression** on the menu that appears.

Using the Python Debug Prompt

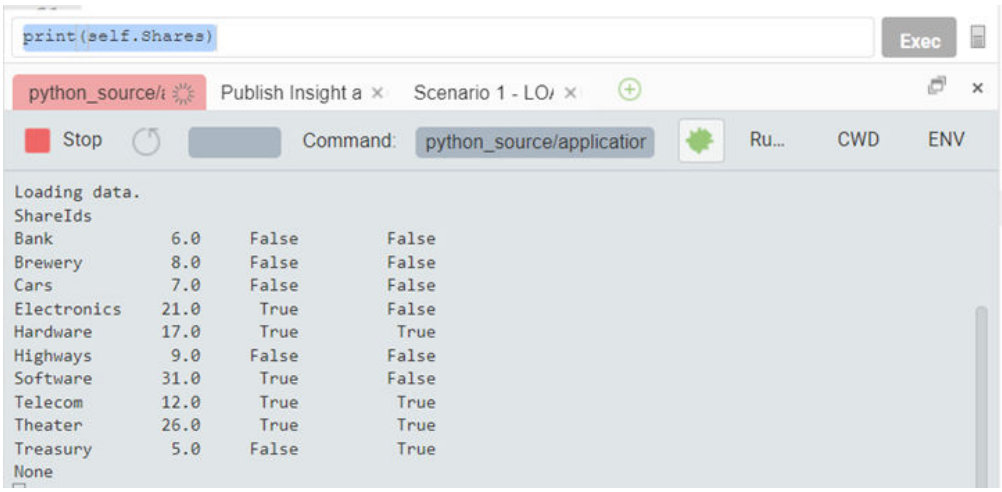
The debugger sidebar and Variable inspection limits the number of values displayed for performance reasons.

A debug prompt for Python models has been added located above the Output pane which allows you to enter arbitrary expressions, whose results will be displayed on the

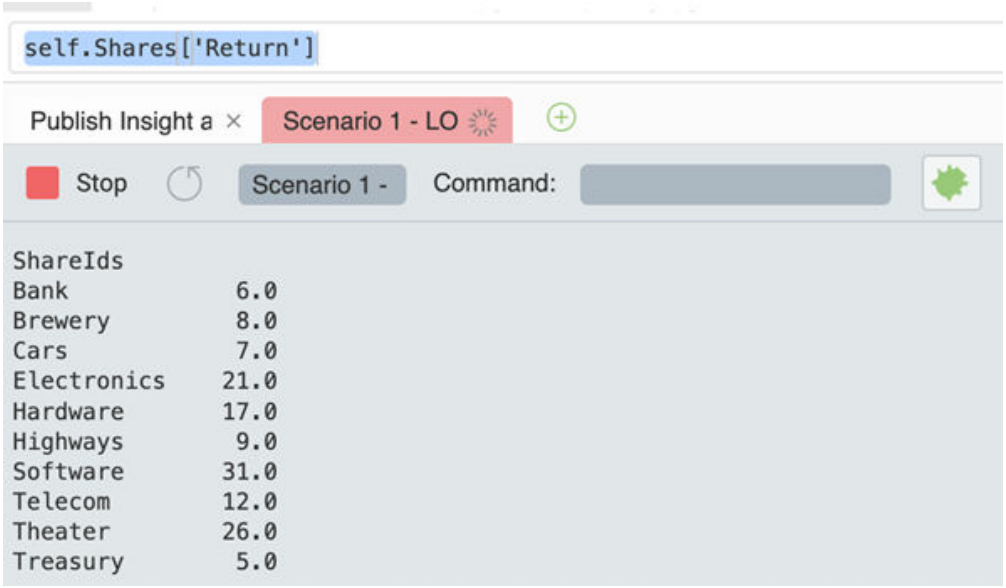
Output pane. It can be used to conveniently print any variable in the current local stack frame or from global scope.

The prompt supports evaluation of expressions and statements. Specifically, if the value of the expressions is not the Python "None" value, it will be displayed. For example:

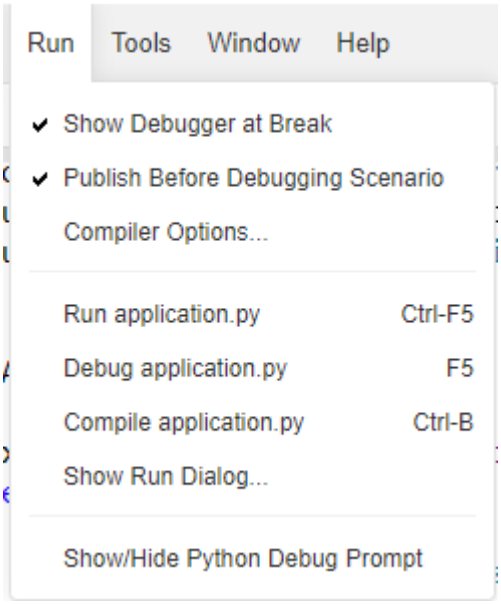
- '1.0==1' will display as 'true'
- 'print(15)' and '15' both result in the same output; '15'
- A = 15 results into "None" and won't be displayed. However, the value is set and 'A' or 'print(A)' in a subsequent command will display '15' on the Output pane.



This can also be used to show a subset of the data:



If the debug prompt does not appear, click on the **Run** menu, and select **Show/Hide Python Debug Prompt**:


Run
Tools
Window
Help
Show Debugger at Break
Publish Before Debugging Scenario
Compiler Options...
Run application.py Ctrl-F5
Debug application.py F5
Compile application.py Ctrl-B
Show Run Dialog...
Show/Hide Python Debug Prompt

CHAPTER 7

Working with Xpress Insight

Xpress Workbench can assist you with many tasks during the development of Xpress Insight apps, including:

- Publishing your project to Xpress Insight
- Building an Xpress Insight app archive
- Debugging scenarios running on an Xpress Insight server
- Downloading new and modified workbooks from Tableau Server into your project
- Autocompletion in VDL source files

Configuring an Xpress Insight Server - Desktop Edition Only

You need to configure an Xpress Insight server before you can publish to or otherwise interact with it.

The first time you try to publish to Xpress Insight, you will encounter a dialog box prompting you to provide configuration information.

In the dialog, review the list in the left-hand pane and select the appropriate server. If your planned server is not in the list, click **Add** and follow the prompts to add a new server. ifComplete the other fields and click **Save** to save your settings.

By default for Xpress Insight 4, if a locally-installed instance of Xpress Insight is active at the URL `http://localhost:8860`. You will need to enter your credentials in the form fields. Trial version user credentials are `admin/admin123` - contact your system administrator for details of other login options.

For Xpress Insight 5, the default URL is `http://localhost:8080`. After adding and selecting an Xpress Insight 5 URL, if Workbench detects that this Insight server is using a later version, the username and dialog boxes are removed and the credentials are picked from the OS credential store. See Providing the REST API Credentials.

Troubleshooting Xpress Insight HTTPS Connections

If the Xpress Insight server is configured to use TLS connections (`https:` rather than `http:`), some circumstances might cause a certificate error when you try to publish or otherwise connect from Xpress Workbench to Xpress Insight (see Publishing to Xpress

Insight). This results in the error message `unable to verify the first certificate`. If this happens, try the following steps:

1. Make sure the security certificate, and any intermediate certificates for the Xpress Insight server, are present in the OS certificate store on the machine where Xpress Workbench is installed.
2. If that does not resolve the error, extract those certificates from the Xpress Insight server into a single `.pem` file (in ASCII text rather than binary format) and copy that to the file system of the machine where Xpress Workbench is installed. Then set the environment `NODE_EXTRA_CA_CERTS` to the full path to the `.pem` file. This will enable Xpress Workbench to find the certificates at run time.
3. If you still see the same error, or a "self-signed certificate" error, you can use a Workbench-specific override. Set the environment variable `WORKBENCH_EXTRA_CA_CERT_FILES`. For the value, specify a list of full paths to files, each one containing one certificate in ASCII format. These must be fully resolved paths and separated using the correct delimiter for your operating system (semicolon for Windows and colon for other OSs).

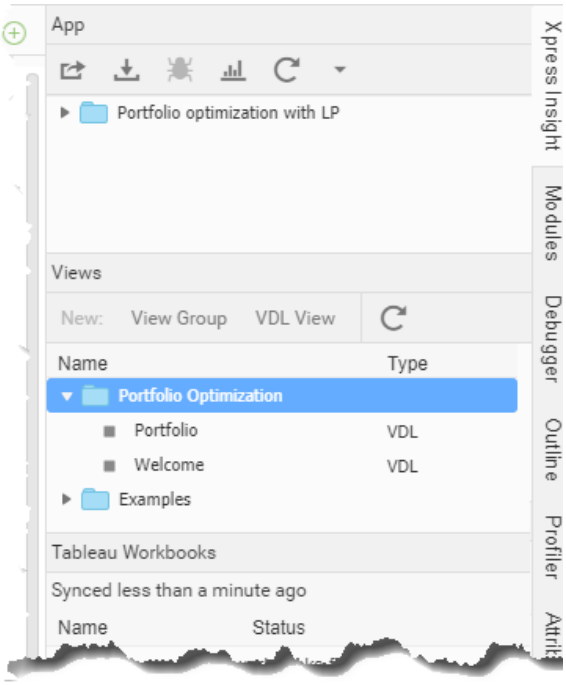
   Note: If you are deploying Workbench in its own Docker container, add the certificate files to the `workbench-config` volume. The environment variable must be set in the `.env` file, with the paths specified as `/workbench-config/`*`certificate_filename`*. Separate the paths using colons.

Managing Insight Views

The Xpress Insight sidebar in Xpress Workbench features a **Views** pane that enables you to create, re-arrange, group, and delete views.

Views pane

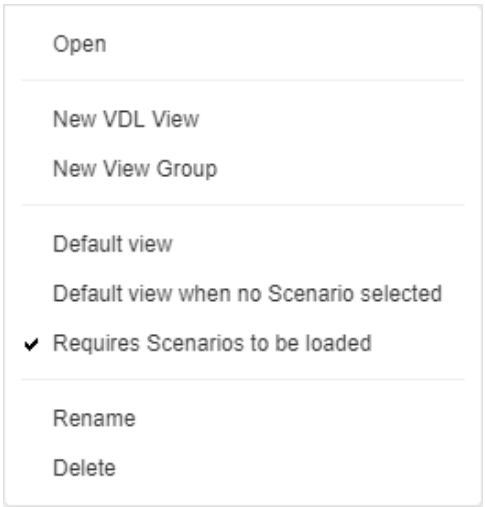
The **Views** pane is displayed when the Xpress Insight sidebar is open.



You can double click on any file displayed in the Views pane to open it. There are three buttons at the top of the **Views** pane:

- **View Group**—Create a new View Group, for more see Managing View Groups.
- **VDL View**—Add a new VDL View, for more see Managing Views.
- Refresh (icon)—Refresh the display in the **Views** pane

You can perform several view management tasks by right clicking on a folder or view in the Views pane—The menu content reflects the options available to the item you clicked on.



Managing Views

The process of creating Views is also described in Creating a New View on page 64.

There are three ways to create a new view:

- In Xpress Workbench, you can click **File > New > Insight View(VDL)**.
- In the View Designer, click the green plus icon in the tab area.
- In the View Designer, **Xpress Insight> Views** pane, click **VDL View**.

The **New View Wizard** is displayed. Proceed through each page of the New View wizard before clicking **Finish**. The new view is displayed on the artboard.

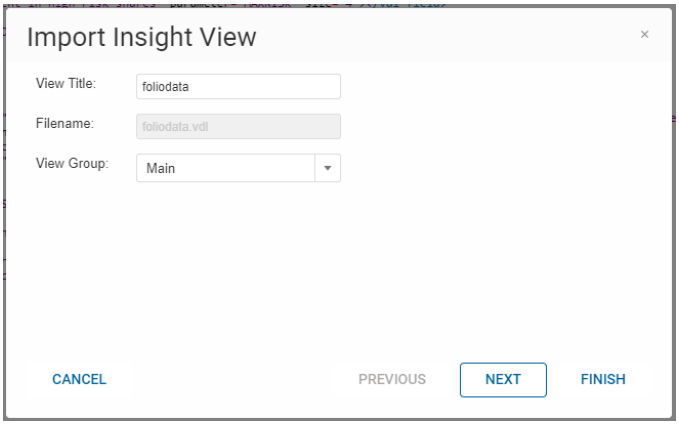
Importing Views

Existing Views can be imported into the project using the Import Insight View option.

1. Copy a suitable source file into the Project folder, either through the operating system explorer software, or by dragging and dropping the file onto the Project pane in Workbench.

   You can import `.html` or `.vdl` into the `client_resources` folder, or `.twb` or `.twbx` into the `tableau` folder.

2. Right-click on the file to open the **Import Insight View** Wizard-This wizard is only displayed if an appropriate file type is selected.



3. Enter a Title for the View, and select the View Group. Click Next.
4. Select the View Options.

Menu option	Definition
Requires Scenario To Be Loaded:	This option is enabled by default-It can be disabled if you are creating a view containing static introductory information, or a view used to configure how scenarios are loaded.
Set As Default View:	The view will be displayed in the Xpress Insight web client when a scenario selected-This will replace any view previously set as default view in your app.

Menu option	Definition
Default When No Scenarios Selected:	The view will be displayed by default until a Scenario is selected.

5 Click **FINISH**.

The new view is displayed on the artboard.

Deleting Views

To delete a View:

- In the **Views** pane, right click on the View and select **Delete**—This will remove the file from the view. Confirm the deletion in the next dialog.

  You will also be given the option to delete any associated source files from the project.

Managing View Groups

View Groups provide a way to organize the views when they are displayed in Xpress Insight.

Creating a View Group will create a new category in the **View Navigator** drop down selector in Xpress Insight.

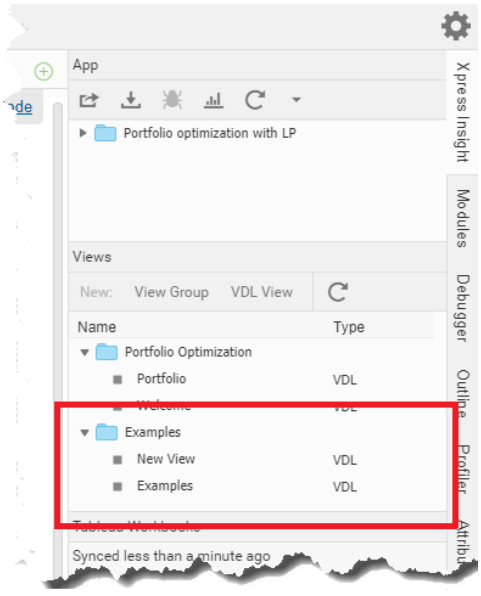


Figure 4: View Group in the Views pane

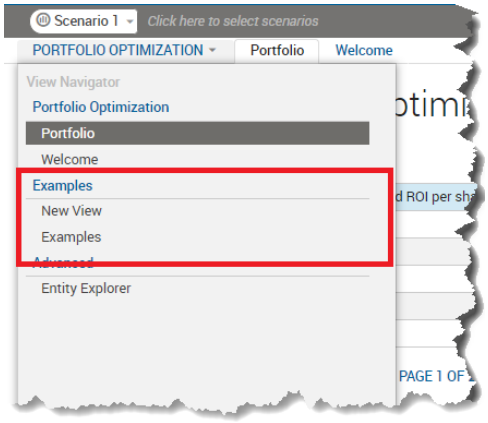


Figure 5: View Group in Xpress Insight

There are two ways to create a View Group. In the **Xpress Insight** > **Views** pane:

1 Click the **View Group** button.

2 Right click in the list of Views and select **New View Group**.

A new View Group is created, shown as a folder in the Views list.

Deleting View Groups

To delete a View Group:

- In the Views pane, right click on the View Group and select **Delete**. The View Group and all Views in that group are deleted.

Changing the View order

When views are published to Xpress Insight they are displayed across the browser in the order they are arranged in the Views pane.

To change the display order, open the Xpress Insight sidebar. In the **Views** pane, click and drag the views and folders displayed in the list to re-order them—The Views in the list are displayed from left to right in Xpress Insight.

Note: View Groups cannot be nested.

Creating a New View

Perform the following steps to begin creating the user interface.

To create a new View using **Xpress Workbench 3.1** or higher:

1. Click **Open Project**.
2. Either:
   - Click **File > New > Insight View (VDL)**.
   - Click the **New View** button on the tab bar.

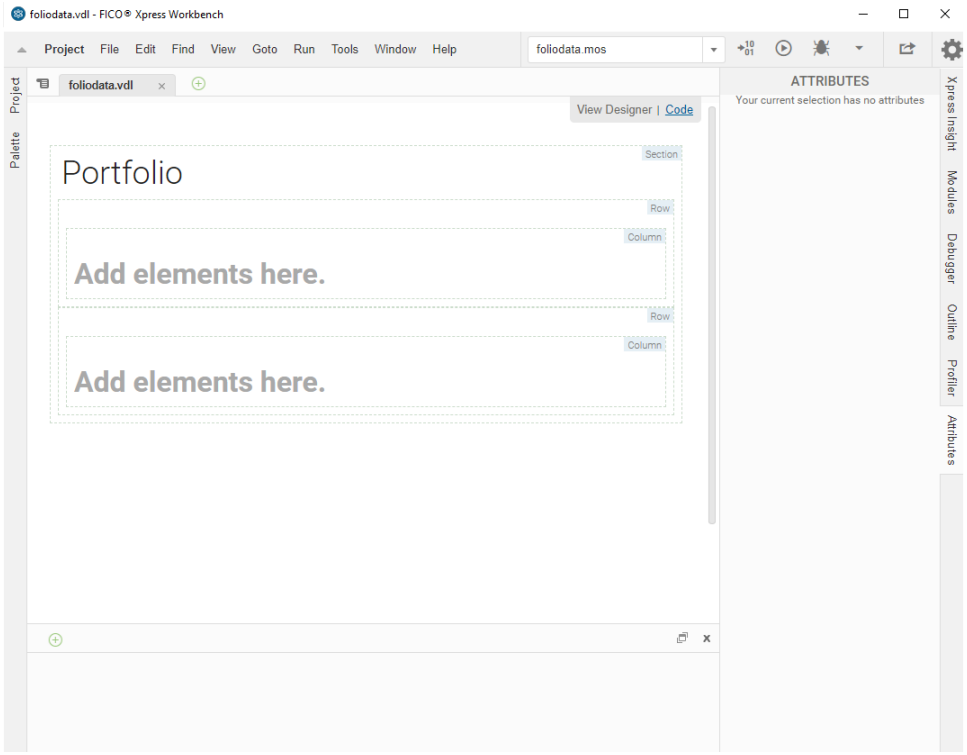
     

     Figure 6: New View button
   - In the **Xpress Insight** tab, click **Views > VDL View**.
3. Proceed through each page of the **New View** wizard.
   1. In the first wizard page, enter Portfolio in the **View Title** field—This text is displayed as the view label by Insight and sets the default value for the **Filename**. Change the Filename value to "foliodata.vdl".Click **Next** to proceed.
   2. The second wizard page enables you to select a page layout: the options are a Basic view, Two columns, or Three columns. Click **Next**.
   3. In the final wizard page, select from the following options:

Menu option	Definition
Requires Scenario To Be Loaded:	This option is enabled by default-It can be disabled if you are creating a view containing

Menu option	Definition
	static introductory information, or a view used to configure how scenarios are loaded.
Set As Default View:	The view will be displayed in the Xpress Insight web client when a scenario selected- This will replace any view previously set as default view in your app.
Default When No Scenarios Selected:	The view will be displayed by default until a Scenario is selected.

4 Click **FINISH**. The new view is displayed on the artboard.



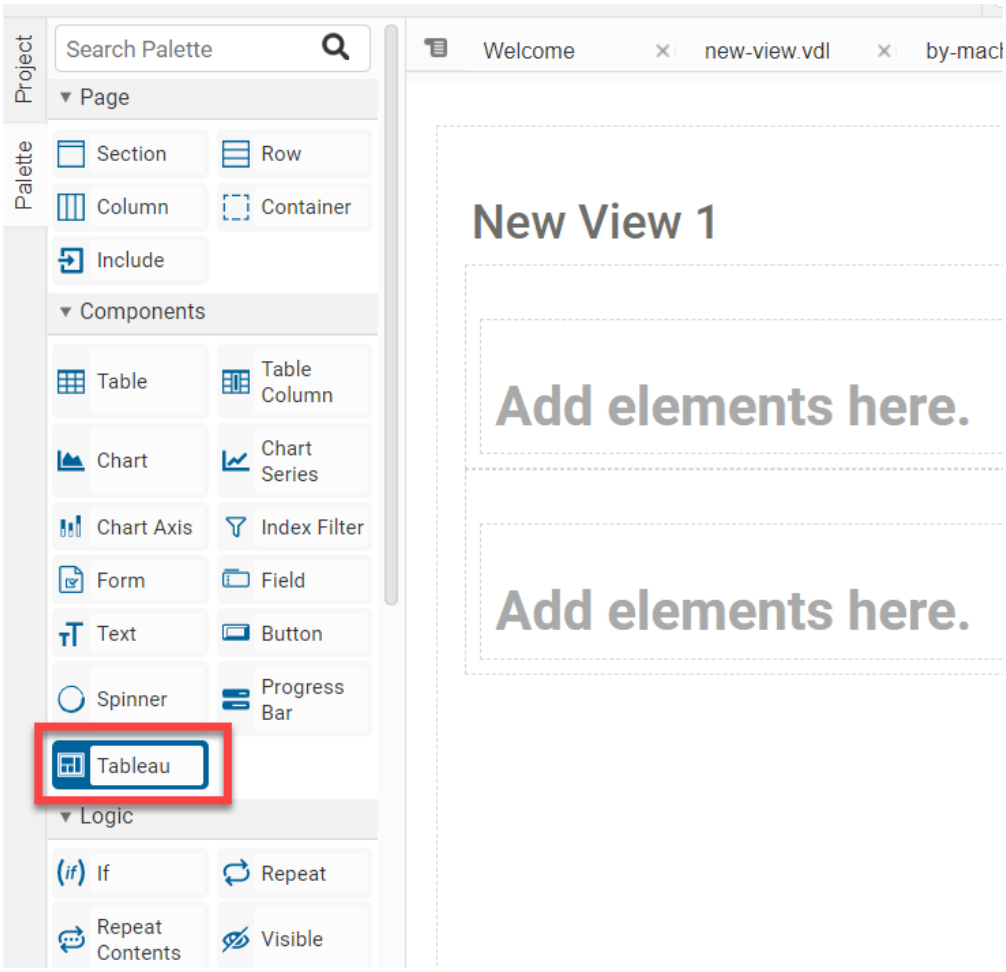
5 If desired, switch to the code editor view using the link at the top right of the artboard.

View Designer Overview

View Designer is a visual editor available in Xpress Workbench that makes it easy to design and develop the views made available in Xpress Insight apps.Use View Designer to compose a view by dragging interface elements into a symbolic layout and configuring the behavior of those elements with wizards.

View Designer presents a symbolic layout that represents the structure of the final VDL file, it does not behave as a WYSIWYG editor. For advanced tasks you can switch between View Designer and the Code editor to work directly with the underlying markup. Your view can be quickly and easily previewed in Xpress Workbench.

In View Designer, a **Palette** pane is displayed on the left, the artboard is shown in the central area, and an **Attributes** pane is on the right.



Views for Xpress Insight apps are composed of various elements: Layouts (such as Section, Row, and Column), Components (such as Tables and Charts), and Logical elements (such as Repeat and Visible). Drag elements from the **Palette** to the artboard to add them to the view.

Note: Views are regularly automatically saved while they are open in View Designer.

Hovering over the elements on the artboard will highlight their boundaries. Elements can be selected by a left click.

The most commonly used attributes of the currently selected element are displayed in the **Attributes pane**; The selected element type is shown at the top of the **Attributes pane**.

Elements placed on the artboard can be moved relative to each other, duplicated, or deleted by utilizing the action toolbar that displays when a placed element is selected. Elements can also be cut, copied, and pasted.

Using the Palette

Elements are organized on the **Palette** in groups. Click and drag the element you wish to use onto the artboard. As you pass over placed layout elements, they will change color to indicate if the dragged element can be dropped there.

- blue shading indicates a valid position.
- red shading indicates an invalid position and the element will not be accepted at this position.

The five panes in the Palette and the elements they contain are:

LAYOUT	Section	Block of page content with padding top and bottom.
	Row	Each row is placed under existing content and can hold up to 12 columns.
	Column	Divides a row into vertical blocks with a configurable width.
	Container	Defines an area in the view and is used to hold other elements.
	Include	Include another VDL source file at this point in the page.
COMPONENTS	Table	Placeholder for a table of array entities that are defined in its Table Columns.
	Table Column	Defines the array entities that provide the content for a Table.
	Chart	Placeholder for a chart that displays model entity data or custom data.
	Chart Series	Set of entity or data information supplied to the Chart element.
	Index-Filter	Filter an index out of a VDL chart or table.
	Form	A container for field elements that controls their behavior.
	Field	Used to enter text or values in a Form.
	Button	Can be placed anywhere in the view to set a value and run the scenario.
	Text	Text in the view can be static or updated using expressions.
	Spinner	A spinner image.
	Progress Bar	A progress bar that can indicate % complete.
	Tableau	An element that contains a Tableau visualization.
LOGIC	If	Use an expression to control whether an element is displayed.
	Repeat	Loop around each element in an array, set, element in a JS array, or active scenario.

	Repeat Contents	Used to repeat a block of content.
	Visible	Use an expression to control the visibility of an element.
	Var	Defines a variable name and its value.
	Set Sorter	Change the way a set is sorted from the default settings.
HTML	List	A container for List Item elements that populate a bulleted list.
	List Item	A container for an item in a list, each list item must be populated with content such as a text component or an html image container.
	Image	Defines the source and size of an image in the view.
ACTIONS	Action	The base action used to call other Actions or Javascript functions.
	Message	Display a message.
	Confirm	Show a confirmation dialog.
	Execute Model	Queue the model for execution.
	Drawer	Open a drawer page in the current view.
	Open View	Navigate to another view.
	Manage Scenarios	Open the Scenario Manager.
	Job Message	Send a message to an executing scenario.
	Cancel Job	Cancel the job for an executing scenario.
	Set Var	Set a dynamic variable.
	Update Entity	Set the value of a model entity.
	Remove Entity	Delete the value of a model entity.
	Add Label	Add labels from a label array to data.
	Aggregate Data	Aggregate data as an array of numbers, or an array of objects that contain a *value* property.
	Push Array	Push a value to a VDL variable that has been defined as a dynamic array.
	Get Entity Data	Fetch data for one or many entities from single or multiple scenarios.
	Get Index Set	Return the set names associated with the given entity name
	Group Data	Group data by one or more of its keys.
	Table Export	Trigger the export function of one or more DataGrid, TreeGrid, or Pivot Table components.
	Create Attachment	Create a new attachment.
	Edit Attachment	Edit a existing attachment.

	Manage Attachment	Manage an attachment.
	Edit Attachment Properties	Open a dialog to change the properties of an attachment.
	Download Attachment	Download an attachment.
	Upload Attachment	Upload an attachment.
	Tableau Export	Export the tableau viz in one of the accepted formats.
	Tableau Clear Global Filters	Clear Tableau global filters from browser storage and update filters in any active visualizations in the view.

Using the View Designer and VDL to Create Custom Views

The View Designer creates Views in VDL, an extensible markup language that is a super-set of HTML and follows the same syntax, along with additional elements defined by the VDL language.

The VDL code created by the View Designer utilizes reusable components containing extensive functionality to define configurable interactive views.

Content can be augmented with customization via HTML, CSS and JavaScript. This chapter explores the underlying VDL code that makes up the View in greater detail.

VDL Versioning

When a VDL file is created in View Designer, a version of the VDL language is defined at the beginning of the file using the `<vdl version="">` attribute. When defining the VDL version, you are notifying the Xpress Insight Server of the major version and optionally the minimum minor version that your VDL view requires.

You can define either a major and minor version, for example `<vdl version="5.1">` or a major version only, for example `<vdl version="5">`. In both cases, the Xpress Insight Server will provide the latest compatible minor version and always match the major version. If no compatible minor version is available, an error message is shown when accessing the view.

If you are using components in a view that require a newer version of VDL than is defined in the VDL file, View Designer will show a warning.

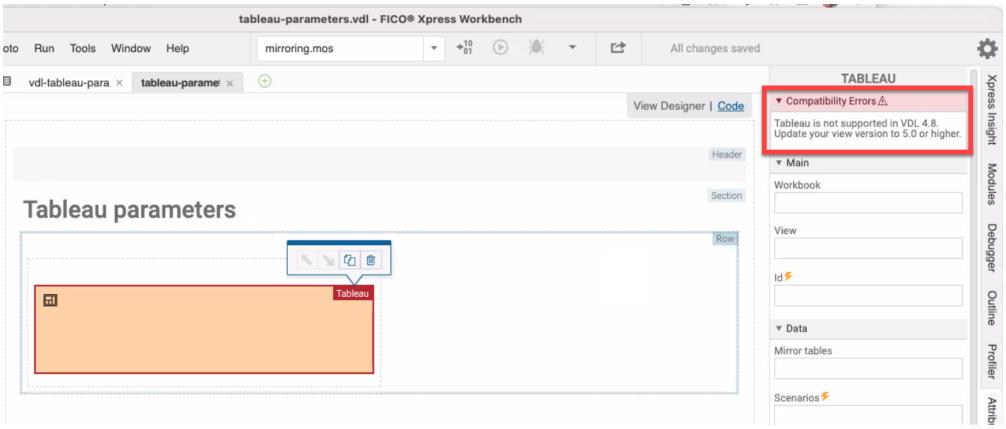


Figure 7:

To update the VDL version, access the code editor as described below, and update the `<vdl version="5.1">` property to the required version.

Accessing the Code Editor

You can quickly switch between the code editor and the View Designer using the links displayed in the top right corner of the artboard.

Figure 8: View Designer link

This section provides examples in both the View Designer and code editor when the feature being described is accessible by both methods.

The View Designer integrates with Xpress Workbench entities and scenarios, generating the required VDL code to provide simple, direct bindings to your app's model data, to aid rapid view development.

VDL Examples App

Your Xpress Workbench installation contains an app that showcases all the elements within VDL, it is located at `<installdir>/examples/insight/vdl/vdl_language.zip`. You can upload it to Xpress Workbench and click on its tile to launch it within the Xpress Workbench environment.

Open Source Extensions

In addition to the developed VDL components, the View Designer also supports the integration of additional open source components. FICO has collected several examples

and documented the installation process for Xpress Workbench to enable them to be used in Xpress Insight views. The available examples include:

- `vdlx-sidebar`, the main sidebar extension.
- `vdlx-toolbar`, a simple toolbar at the top of view.
- `vdlx-dropdown-button`, for a drop-down list.
- `vdlx-tabs`, to create tabs or pills for inner view navigation; and `vdlx-modal`, a modal dialog wrapper that can contain custom VDL.

These examples can be downloaded as a single zip archive from https://github.com/fico-xpress/vdl-custom-extensions.

Javascript API

FICO also offers a JavaScript API. For more information, see Using the JavaScript API to Create Custom Views.

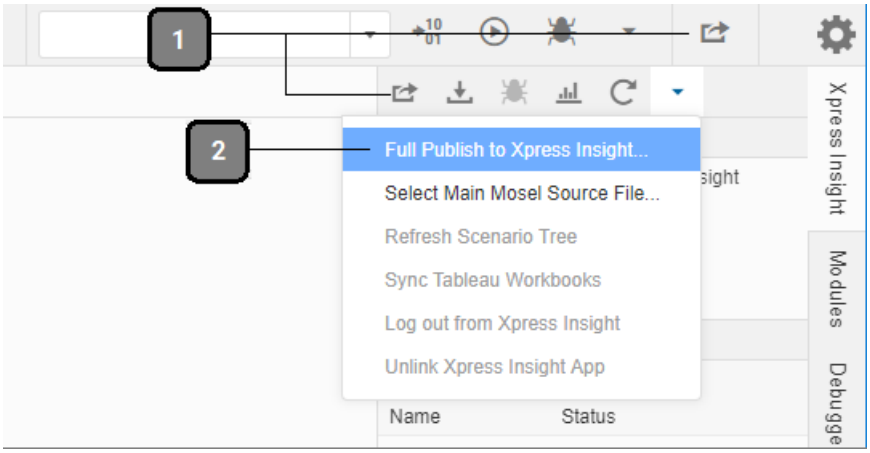
Publishing to Xpress Insight

Creating or updating an app in Xpress Insight involves packaging the app files into a ZIP file and uploading the ZIP file using the Xpress Insight user interface. Xpress Workbench simplifies the process by performing these tasks automatically.

Before publishing to Xpress Insight, you must ensure that your project follows the structure of an Xpress Insight app:

- It must contain a Mosel source file that uses the **mminsight** module.
- It should contain a companion file at the top level of the project, containing the configuration for an Xpress Insight app, including the custom views exposed by the app. For more information, see Insight 5 Companion File XML Reference.
- Any custom view files should be placed in a **client_resources** folder at the top level of the project.
- Any input files needed by the Mosel model should be placed in a **model_resources** folder at the top level of the project.
- Any shared attachments needed by the app can be placed in a **attachments** folder at the top level of the project.

All Xpress Workbench features that relate to Xpress Insight are accessed from the **Xpress Insight** sidebar. You can find this sidebar on the right side of the Xpress Workbench **Project** page. There are two options when publishing to Xpress Insight, Full or Quick.



1. Quick Publish - Use this option to include only the files that have been changed since the last publish
2. Full Publish - Always use this option when publishing to Xpress Insight for the first time-additionally it is recommended you use this option when adding a new view, and as the final publish prior to release.

Complete the following steps to publish your project to Xpress Insight:

1. Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.
2. Use the required **Publish** option.
3. Use the **Xpress Insight Server** drop-down to select the server where you want to publish the app.
4. Select the Xpress Insight app that you want to update or select **Create a new app**.

   Note: On the FICO® Analytic Cloud, the list of Xpress Insight app may take several seconds to load.
5. Click **Publish**.
6. Check the **Output** window, which opens at the bottom of the screen, to see any errors or warnings.

   A warning is shown if there are files or folders in the project that are not permitted within an Xpress Insight app archive.

Updating Xpress Insight Apps

Xpress Workbench will remember the Xpress Insight server and app that you last published to. To publish again and update the same app:

1. Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.
2. (Optional) If you want to change the main Mosel source file for the app, click **Settings**, which is the rightmost button in the **Xpress Insight** sidebar, and click **Select Main Mosel Source File**.

3. Publish an app by doing one of the following:
   - Click **Publish** button which is the first button from the left within the sidebar.
   - If you want to publish to a different Xpress Insight server or app, click **Settings**, which is the rightmost button in the **Xpress Insight** sidebar, and click **Publish to Xpress Insight**.
4. Check the **Output** window, which opens at the bottom of the screen, to see any errors or warnings.

Opening Xpress Insight

Once a project is published to Xpress Insight, it is linked to the Xpress Insight app, and the app's folders and scenarios are displayed in the **Xpress Insight** sidebar.

Complete the following steps to open the app in Xpress Insight:

1. Open the **Xpress Insight** sidebar if it is not already open, by clicking its name.
2. In the scenario tree, right-click the Xpress Insight app name.
3. Click **Open in Xpress Insight** on the menu that appears.

   A new browser tab is opened, showing the app within Xpress Insight.

   Note: To refresh the list of folders and scenarios, click the **Refresh Scenario Tree** button in the rightmost position of the Xpress Insight toolbar.

Exporting Xpress Insight Apps

Xpress Workbench can automatically publish a project to any Xpress Insight server within the same network environment.

To publish to an Xpress Insight server that is located elsewhere, you must create an app archive file that you can then upload using the Xpress Insight user interface. Complete the following steps to create an app archive file from your project:

1. Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.
2. Click the **Download** button, which is the second button from the left within the sidebar.
3. Check the **Output** window that opens at the bottom of the screen to see any errors or warnings.

   A warning is shown if there are files or folders in the project that are not permitted within an Xpress Insight app archive.

   Important: The save process is handled outside the Workbench framework. If the save operation fails, it might not be reported by Xpress Workbench.

   Note: The Xpress Insight app archive file only includes those files and folders that are allowed by Xpress Insight. To download all of the files and folders in the project, open the **File** menu and click **Download Project**, however this method may not provide a valid Xpress Insight app archive.

Debugging Scenarios

Xpress Workbench can be used to debug scenarios while they are executing on an Xpress Insight server.

First you must have published the project to an Xpress Insight server. You can then start a debug session within Xpress Workbench. You can debug any type of scenario execution, whether the scenario is being loaded or run, or is executing in a custom run mode.

Complete the following steps to debug a scenario:

1. Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.
2. Expand the scenario tree and select the scenario you want to debug.
3. Click **Debug**, which is the third button from the left within the sidebar.
4. In the menu that appears, select the execution mode that you want to use for the debug session.

   The app is published to Xpress Insight before the debug session starts.
5. You can now debug the model as usual. See Debugging Models on page 37 for more information.

   Note that:

   - If you set a breakpoint, the line with the breakpoint is highlighted to show that model execution is suspended at that line.
   - Otherwise, the model runs to completion and suspends at the last line, so you can review the final state of the model.
   - The **Debugger** sidebar opens, where you can see the call stack and the values of the variables defined by the model.

Once you have debugged a scenario, it will be added to the **Run Configurations** drop-down list in the main menu bar. You can quickly repeat the same debug session again by clicking the **Debug** button to the right of the **Run Configurations** drop-down list.

Manually Executing Scenarios

Because there are so many ways to run a scenario, and they are often specific to the Xpress Insight app, Xpress Workbench can also start a debug session without automatically executing the scenario. After starting the debug session, you must open Xpress Insight and run the scenario manually.

Complete the following steps to debug a scenario manually:

1. Open the **Xpress Insight** sidebar, if it is not already open, by clicking its name.
2. Expand the scenario tree and select the scenario you want to debug.
3. Click the **Debug** button, which is the third button from the left within the sidebar.
4. In the menu that appears, select **Debug Scenario Manually**.

   The app is published to Xpress Insight before the debug session starts.

5 When you see the message `Click to open Xpress Insight`, click the link.

If the message disappears before you click it, you can open Xpress Insight manually.

Alternatively, you can right-click the scenario in the **Xpress Insight** sidebar and click **Open in Xpress Insight** in the menu that appears.

A new browser tab opens showing the app in Xpress Insight, and the scenario appears on the shelf.

Note: You may need to log in to Xpress Insight before the app is opened.

6 In Xpress Insight, execute the scenario. This could be done using the scenario menu or by using a button on a custom view.

The scenario appears to be continually executing

7 Switch back to Xpress Workbench.

8 You can now debug the model as usual. See Debugging Models on page 37 for more information.

Note: When a debug session has started and Xpress Workbench is waiting for the scenario to be executed, the debugger only attaches if the scenario is run by the same user who started the debug session. If another user logs into Xpress Insight and runs the scenario, the debugger does not attach.

Cancelling Debug Sessions

A debug session can be cancelled, either before the scenario is executed or during execution.

If the session is cancelled before the scenario is executed, the scenario executes as normal when it is next run, and the debugger does not attach.
Complete the following steps to cancel a debug session:

1 Click **Stop** in the **Output** window.

2 Wait for the model to finish executing

The debug session stops immediately. You cannot review the final state of the model before it exits.

Unlinking Projects from Xpress Insight

Xpress Workbench will remember the Xpress Insight server and app that you last published to. When you do not wish to continue publishing the project to this app, you can unlink the project from Xpress Insight.

Complete the following steps to unlink the project from the Xpress Insight app:

1 Click the **Settings** button, which is the rightmost button in the **Xpress Insight** sidebar.

2 Click **Unlink Xpress Insight App**.

Command-Line Tool for Common Operations

In addition to the interactive user interface, Workbench provides a simple command-line interface for common operations. The tool is called `xpwinsightcmd`, and its filename and location vary by platform.

- On Windows, the `xpwinsightcmd.bat` file is in the `workbench` folder.
- On macOS and Linux, the `xpwinsightcmd.sh` file is in the `bin` folder.

On Windows, the tool can be invoked as `xpwinsightcmd` or `xpwinsightcmd.bat`, and on macOS and Linux as `xpwinsightcmd.sh`. The Windows installer and the macOS `xpvars.sh` script update your `PATH` environment variable to include the folder locations of the tool.

The output of `xpwinsightcmd` is the same as you would see in the terminal windows when performing the equivalent actions in the Workbench UI. Running the tool without any arguments prints the available options, and the process exits with an error code.

```
No action supplied
xpwinsightcmd options:
xpwinsightcmd build [--base path/to/project] [-g | -G] path/to/main/file
xpwinsightcmd publish [--base path/to/project] path/to/main/file --to URL
        [--app-name app-name] [--app-id app-id] [--create-if-missing]  [--full] [--prod]
        [--python path/to/python | --python-conda] [--python-args args]
        [--exec-env name] [--server-credentials-file path/to/credentials]
```

Note: The output in the example above is from macOS and uses forward slashes (`/`) in its paths; on Windows backslashes (`\`) should be used. Likewise, if paths or other command-line arguments contains spaces or other complex characters, those arguments should be escaped or quoted in an OS-appropriate manner.

Note: This tool is focused on operations related to the development and deployment of Insight apps. For command-line access to actions based around Xpress itself and no relation to Insight (e.g. compiling or running a Mosel model), use the Xpress command-line tools such as mosel (.exe)

Command-Line Tool Options and Arguments

The **`--base`** option is shared by all actions and is used to specify the root directory of the Workbench project (the parent of the `source` or `python_source` folder). If this option is omitted, then the current working directory is used as the base. Other file path arguments are relative to this base directory.

Each action requires the argument specifying the path to the main Mosel file that contains the model directive or the Python source code that defines the Insight app class. For Python projects, the main file is `python_source/application.py`. Other possibilities may be added in future.

build action

The **build** action creates an app `.zip` file suitable for uploading to Insight. This is created in the project root folder, overwriting any previous `.zip` file. By default, the code is compiled for production deployment with debugging disabled.

The syntax of the `build` action is as follows:

```
xpwinsightcmd build [--base path/to/project] [-g | -G] path/to/main/file
```

Options:

- `--base` *`dir`*: (optional) the project root directory, defaults to working directory (see above).
- *`path/to/main/file`*: (required) the main file (see above).
- `-g` or `-G`: the debugging level flag to pass to the Mosel compiler.

publish action

The `publish` action will create or update an app on the Insight server. By default, Mosel code is compiled for debugging rather than production use by passing the `-G` option to the compiler; you can use the `--prod` option to override this default.

By default, the action assumes the target Insight is version 5 or later and will acquire credentials from the OS as described in Equipping Workbench with Credentials to Publish. If the Insight server is version 4.x, then this must be specified using the `--insight-4` option, and the credentials passed in the command line.

Insight credentials on Windows and macOS are handled in the same way as they are with the desktop Workbench application. On Linux, either the `--server-credentials-file` option or the `WORKBENCH_CREDS_FILE` environment variable must be used.

The syntax of the `publish` action is as follows:

```
xpwinsightcmd publish [--base path/to/project] path/to/main/file --to URL
                [--app-name app-name] [--app-id app-id] [--create-if-missing]
                [--full] [--prod] [--exec-env name]
                [--server-credentials-file path/to/credentials]
```

Options:

`--base` ***`dir`***	(Optional) The project root directory, defaults to working directory (see above).
`path/to/main/file`	(Required) The main file (see above).
`--to` ***`URL`***	(Required) The root URL of the Insight server.
`--app-name` ***`name`***	(Optional) The name of the app to create or update (see `--create-if-missing`).
`--app-id` ***`id`***	(Optional) The UUID of the app to update.
`--create-if-missing`	(Optional) When used with `--app-name`, updates any existing app of that name; otherwise creates a new app.

`--full`	(Optional) Perform a full rather than partial publish.
`--prod`	(Optional) Publishes for production use rather than debugging. This option overrides the `-G` option that is passed to the compiler by default. **Note:** A partial publish might skip compiling some files. If you have built previously using different debug levels, use the `--full` option along with `--prod` to make sure all published content has the same debug level.
`--exec-env` ***`name`***	(Optional, Insight 5.8 and later only) Update the name of the app's execution environment.
`--server-credentials-file` ***`filename`***	(Linux only) Path to file containing client ID and secret of Insight API tokens. You can use the `insight-credentials.template` file as a starting point for this file. Alternatively, you can set the environment variable `WORKBENCH_CREDS_FILE` to this path instead of adding it to the command line.

Specifying an existing app (if any)

If only the required arguments are used, a new app will be created each publish. To specify an existing app, the simplest method is to use the `--app-name` option. If this is combined with the `--create-if-missing` option, then the app will be created if it does not already exist. The two options can be used together so the first publish will create the app, and subsequent ones will update it in-place.

Note: If the name of the app as specified by the model source or companion file does not match the supplied name, then upon publish the name of the app on the server will be updated to match it. This means subsequent publishes will then be able to find the app by name.

Alternatively, the user can also identify the app to update by app id. The app id is immutable and provides a stable reference when the name may have changed (for instance via Insight's auto-rename to avoid a name collision).

To obtain the id of the app, open the app in the Insight web client (so the URL looks something like <server>/insight/#eyJwc…), open the Developer tools on the web browser, and enter the following at the Console:

```
console.log(JSON.parse(atob(location.hash.substring(1))).project)
```

This will print out the app id.

Full vs partial publish

By default, `xpwinsightcmd` will behave like the Workbench UI and use its last known publish timestamp to determine what to compile and what to include in the publish payload. If you specify the `--full` option, then everything will be recompiled, and all resources will be re-published.

Insight Execution Environment

This can be specified via the `--exec-env` argument. See Configuring the Execution Environment for details about Insight Execution Environments.

CHAPTER 8

Authoring Tableau Views for Insight Applications

You can use Xpress Workbench to create and edit Tableau workbooks.

The instructions in this section assume that your project is linked to an Xpress Insight app. See Publishing to Xpress Insight on page 71.

Creating Workbooks

Tableau workbooks are created and edited using Tableau Server. After creating a workbook, it must be associated with the Xpress Insight app by adding an entry in the companion file.

Before creating a Tableau workbook, ensure that your companion file defines at least one mirror table and data source. The following example defines a mirror table containing a single entity, and a data source containing the mirror table:

```
<database-mirror>
  <mirror-tables>
    <mirror-table name="my_table">
      <entity name="my_entity"/>
    </mirror-table>
  </mirror-tables>
  <data-sources>
    <data-source name="my_data_source">
      <mirror-table-ref name="my_table"/>
    </data-source>
  </data-sources>
</database-mirror>
```

Note: A Tableau workbook can only utilize a single data source. This data source may contain several mirror tables, but the tables may have overlapping index sets so that they can be joined together into a single SQL query. See the Xpress Insight Mosel Developer Guide or Python Developer Guide for more information about how to configure mirror tables and data sources.

Complete the following steps to create a new Tableau workbook:

1. Open the companion file by double-clicking its filename.
2. Ensure that the companion file contains a valid Tableau data source.

3. Choose a name for the new workbook and add a new `tableau-workbook` entry to the companion file, specifying your chosen workbook name and setting the `managed` attribute to `true`.

   See the Xpress Insight Mosel Developer Guide or Python Developer Guide for more information about how to define custom views in the companion file.

4. Save the companion file and publish the project to Xpress Insight.
5. Open the Xpress Insight sidebar by clicking its name.
6. Click **Tableau**, which is the fourth button from the left in the sidebar.

   The Tableau Server launch page opens in a new browser tab.

   You may be prompted to log into Xpress Insight before this page appears.

   If the mirror database is empty, you are prompted to mirror a scenario. This is to ensure that the workbook data source contains some data.

7. (If prompted) Select the scenario to mirror, and click **Continue**.

   Tableau Server is launched.

8. Open the `Default` project in Tableau Server.
9. Click the **Data Sources** tab.

   A list of data sources appears. The data sources defined in the companion file is prefixed with the mirror table prefix for the app.

10. Click the name of the data source that you want to use in the workbook. Only data sources that belong to the current app should be used.

    A list of workbooks linked to the data source appears.

11. Click **New Workbook** to the right of the datasource name.
12. Create the content of the workbook.
13. Click **Save As** in the page header.
14. Enter the workbook name exactly as you entered it in the companion file.
15. Click **Save**.
16. The workbook is downloaded to Xpress Workbench.

    Important: New workbooks created in Tableau Server will not appear in Xpress Insight until you log out of Xpress Insight and log back in. Log out by clicking your username in the top right of the screen and then clicking **Log Out**.

Editing Workbooks

Complete the following steps to edit an existing Tableau workbook:

1. Open the Xpress Insight sidebar by clicking its name.
2. Click the **Tableau** button, which is the fourth button from the left in the sidebar.

   The **Tableau Server** launch page opens in a new browser tab.

   If the mirror database is empty, you are prompted to mirror a scenario. This is to ensure that the workbook data source contains some data.

3. (If prompted) Select the scenario to mirror, and click **Continue**.

   Tableau Server is launched.

4. Open the `Default` project in Tableau Server.
5. Click the workbook you want to edit.
6. After your changes are saved, the workbook is downloaded to Xpress Workbench.

   Note: Tableau Server can only be used to edit workbooks whose data sources are defined externally, in the companion file. If the Xpress Insight app contains any workbooks with embedded data sources, the system will prevent Tableau Server from being launched.

Downloading Workbooks into Xpress Workbench

Xpress Workbench automatically downloads workbooks that are created or modified in Tableau Server. This happens in the background.

1. After saving a workbook in Tableau Server, switch back to the Xpress Workbench tab.
2. Wait up to 60 seconds for Xpress Workbench to synchronize the workbooks.

   If there were warnings while synchronizing the workbooks, a message appears in Xpress Workbench.

Viewing the Workbook Status

Complete the following steps to see the status of the workbooks that Xpress Workbench is synchronizing:

1. Open the Xpress Workbench sidebar by clicking its name.
2. Look in the **Tableau Workbooks** section, below the scenario tree.

   The status of each workbook appears.

   Note: Xpress Workbench only synchronizes managed workbooks, which are configured in the companion file with the `managed` attribute set to `true`. Without this setting, the workbook will be ignored by Xpress Workbench.

APPENDIX A

Contacting FICO

FICO provides clients with support and services for all our products.

FICO Customer Support

FICO Customer Support offers technical support and services for your FICO solutions ranging from self-help tools to direct assistance with a FICO support engineer. Support is available to all clients who have an active maintenance contract.

The FICO Customer Self-Service Portal (support.fico.com) is a secure web portal that allows users to open, review, and update their support cases; manage their organization's portal users; find solutions to common problems in the FICO Knowledge Base; and view the availability of their cloud applications 24 hours a day, 7 days a week.

You can find support contact information and a link to the FICO Customer Self-Service Portal on the Global Support Center landing page (www.fico.com/en/product-support).

FICO Community

The FICO Community is a great resource to find the experts and information you need to collaborate, support your business, and solve common business challenges. You can get informal technical support, build relationships with local and remote professionals, and improve your business practices. For additional information, visit the FICO Community (community.fico.com/welcome).

Documentation

FICO continually looks for new ways to improve and enhance the value of the products and services we provide.

If you have comments or suggestions regarding how we can improve this documentation, let us know by sending your suggestions to techpubs@fico.com. Please include your contact information (name, company, email address, and optionally, your phone number) so we may reach you if we have questions.

FICO Learning

FICO Learning is the principal provider of platform and solutions training for our clients and partners. FICO Learning offers instructor-led classroom courses, web-based training, seminars, and training tools for both new user enablement and ongoing performance support.

For additional information, visit the FICO Learning home page (www.fico.com/en/product-training) or email producteducation@fico.com.

About FICO

FICO (NYSE:FICO) is a leading analytics software company, helping businesses in 90+ countries make better decisions that drive higher levels of growth, profitability, and customer satisfaction. Learn more at www.fico.com or contact us at www.fico.com/en/contact-us.

Index

A

apps
    updating 72
array data
    CSV files 41
arrays 40, 55
autocomplete 31, 32

B

breakpoints 17, 38, 53

C

cancelling debug sessions 75
closing projects 13, 21
code
    folding with the keyboard 33
    folding with the mouse 33
collaborating 33
command-line options and arguments 76
Command-Line tool 76
compiler options for Mosel 42
compiling models 42
configuring an xpress insight server 59
Converting Xpress Mosel Models to Xpress Insight apps 11
Converting Xpress Mosel Models with the Desktop Edition 11
coverage data 50
Create a new view 62
creating
    files 24
    projects 12, 13, 19, 20
Creating a View Group 63
Creating projects 19

D

dashboard
    displaying 22
debug session 17, 18
debug sessions
    cancelling 75
debugging models 35, 37
debugging scenarios 74
Delete a view 62
deleting projects 21
Deleting View Groups 63
displaying the dashboard 22
downloading
    projects 26
downloading files 26
downloading workbooks 82

E

editing source files 15
evaluating expressions 41, 56
examining arrays 40, 55
expressions 41, 56

F

FICO
    about 84
    community 83
    documentation 83
    education 84
    support 83
    training 84
file menu 23
file tree 23
files
    creating from File menu 24
    creating from file tree 24
    creating from template 24
    creating from View Designer 24
    downloading 26
    opening 23
    renaming 25
    reverting to previous version 26
finding text 28, 30

folding code 33

G

getting started 12

H

highlighting syntax 30

I

IDE 31
Importing Xpress Insight apps to Xpress Workbench with the DMP Edition 11
Inline Documentation 32
inline help 32
Insight tab 64
Installing Xpress Workbench 8
introduction 7

J

Jump to Definition shortcut 28

L

Launching Xpress Workbench 8
logging in 9
logging out 9

M

manually executing scenarios 74
messages
    sending 34
model execution 51
models
    compiling 42
    debugging 35, 37
    resuming 39, 54
    running 35
    running models 35
    stepping through 39, 54
    stopping 36
    suspending 39, 54
    tuning 46
Models
    profiling 49
Modules panel 27
mosel
    profiling 49
Mosel
    compiler options 42
    profiling models 49
mosel deployment wrappers 46
mosel projects 13, 20

N

New View wizard 64

O

Open an Existing Xpress Workbench Project with the Desktop Edition 11
opening files 23
opening projects 13, 21

P

Palette 14
plus icon 62
previous version 26
Profiling models 49
project backup 22
Project pane 14
project restore 22
projects 21
    creating 12, 13, 19, 20
    deleting 21
    downloading 26
    opening 21
    renaming 21
    working with 19
publishing to
    Xpress Insight 72
publishing to Xpress Insight 16, 71
Python
    profiling models 49

R

renaming files 25
renaming projects 21
replacing text 29, 30
resuming models 39, 54
reverting files 26
running models 35

S

sending message 34
source files 15
starting a debug session 37, 53
statement 31
stepping through models 39, 54
stopping models 36
submodels
    parallel solving 45

submodels *(continued)*
    running 45
suspending models 39, 54
syntax
    highlighting 30

T

Tableau Server
    interacting with 80
template files 46
text
    finding 28, 30
    replacing 29, 30
trace mode 51
tuner settings 48
tuning models 46
tuning mosel models 46

U

unlinking projects 75
uploading files 25
Using a template 23

V

variables
    examining 40, 55
VDL
    custom views 69
    orientation 69
VDL View 60
View Designer 14
    artboard 65
    Attributes Pane 65
    custom views 69
View Designer *(continued)*
    orientation 69
        Palette
        COMPONENTS 65
        HTML 65
        LAYOUT 65
        LOGIC 65
View group 60
View Groups 64
Views pane 60, 63, 64

W

words
    automatically completing 31
Workbench IDE 14
workbook
    viewing status 82
workbooks
    creating 80
    downloading 82
    editing 81
working directory 36
working with projects 19

X

Xpress Insgiht
    publishing to 71
Xpress Insight 59
    exporting apps 73
    opening 73
    publishing to 16
    updating app 72
xpress insight projects 13, 20
Xpress Workbench
    description 7