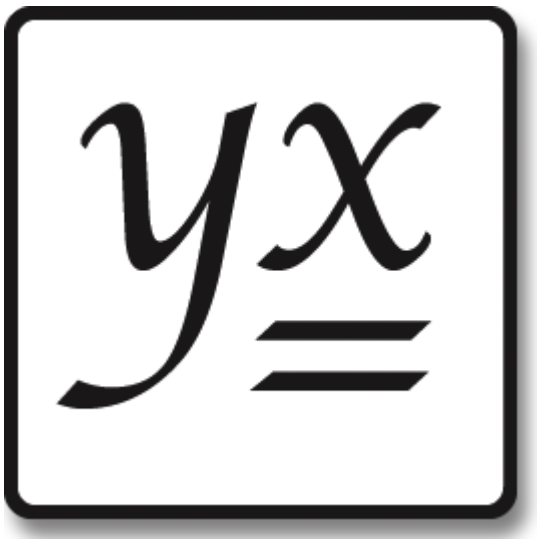


Absyntax Excel Add-in

User Guide
February 2014

Table of Contents

Introduction

The **Absyntax Excel Add-in** for Microsoft Excel integrates Microsoft Excel spreadsheet software and the **Absyntax framework**. The add-in may be considered a universal add-in to the extent that it allows a Microsoft Excel user to invoke multiple user-defined operations, typically on ranges of data in Microsoft Excel worksheets. In this context, "user-defined operation" is a synonym for an Absyntax project.

The Absyntax Excel Add-in is most useful in any of the following circumstances:

- you need to perform operations that are not supported by Microsoft Excel;
- you need to perform operations for which no add-ins or macros exist;
- existing add-ins or macros are not sufficiently configurable to meet your needs;
- your cell formulas are long, complicated and difficult both to understand and to maintain;
- you repeat the same formula across many cells;
- you have to resort to using SQL to perform data table queries;
- you want to use a Microsoft Excel workbook as the source for data to be passed to Absyntax projects;
- you want to feed the output of Absyntax projects into a Microsoft Excel workbook;
- you would rather use a single add-in than multiple disparate add-ins;
- you do not have the software programming skills necessary to harness the Microsoft Excel developer capabilities.

If you are not familiar with the Absyntax framework, you are advised to read the Introduction and Fundamentals sections of the Absyntax Framework User Guide.

Absyntax Framework User Guide

The Absyntax Framework User Guide describes Absyntax fundamentals as well as providing all the information necessary to create, debug and maintain Absyntax projects. It is available online and for download as a pdf.

System Requirements

The Absyntax Excel Add-in is a Microsoft .NET Framework-based COM add-in for Microsoft Excel. Absyntax itself is a suite of Microsoft .NET Framework-based libraries and applications.

Operating Systems

- Windows XP (SP3) (x86)
- Windows Vista (SP2) (x86 and x64) (recommended)
- Windows 7 (SP1) (x86 and x64) (recommended)
- Windows 8 (x86 and x64) (recommended)

Software Requirements

Microsoft Excel 2007 or Microsoft Excel 2010
Microsoft .NET 4.0 (installed automatically with the product if necessary, subject to your agreement).

Hardware Requirements (guideline only)

- 2GHz processor
- 512MB RAM
- 1GB (32-bit) or 2GB (64-bit) available hard disk space for Microsoft .NET 4.0
- 10MB available hard disk space for the Absyntax Framework

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

Installation and Licence Activation

1. Download the zip file containing the latest add-in version from the Absyntax website.
2. Unzip the contained files to a convenient location.
3. Double-click setup.exe to begin the installation.
4. Follow the step-by-step guide. If prompted by a User Account Control dialogue, answer Yes.
5. Once the Absyntax Excel Add-in and, if necessary, the Absyntax framework have been installed you will firstly need to ensure that an activated Absyntax licence is available. Start the Absyntax Licence Manager. A desktop shortcut may exist for this. Alternatively, from the **Start** menu select **All Programs → MII Ltd → Absyntax Licence Manager** or open the Absyntax framework's installation directory and run AbsyntaxLicenceManager.exe. Note: *you must be connected to the internet in order to activate licences*.
6. If you have purchased a licence then enter the purchased licence key and the name of the licensee you specified during the purchase process; then click Add.
7. Activate your purchased licence or, if you do not have one, activate the trial licence that was included with the installation.

Other, less common licensing scenarios are not discussed here. For a full appreciation of all Absyntax licensing options please refer to the Absyntax Framework User Guide.

Fundamentals

This section explains the principles of the Absyntax Excel Add-in.

Project Invocation Rules

Whenever you wish to use an Absyntax project in conjunction with Microsoft Excel (typically to perform some kind of transformation on data that is bound to a workbook), you must define a project invocation rule. Each such rule encapsulates:

1. the target Absyntax project to be invoked;
2. whether the project is reloaded before each invocation;
3. the way in which data is to be acquired from a worksheet and passed into the project;
4. the way in which data emitted from the project is to be written to a worksheet;
5. the amount of time that the Absyntax runtime service will allow the project invocation to complete.

Target Project

The target Absyntax project to be invoked is expressed as the full path to a file with an "apj" extension.

Reloading a Project

For performance reasons it is advantageous to leave a project loaded in memory between invocations because project-loading is a relatively slow process. This is managed automatically. However, once a project is loaded then any changes made to the source project file can only be realised if the project is reloaded. A project invocation rule allows you to specify whether a project should be reloaded before each invocation. This capability is particularly useful if you are in the process of developing or maintaining a project at the same time as using it with the Absyntax Excel Add-in. See Loading and Unloading a Project for more details.

Input and Output Data

You are expected to understand the input and output data requirements of the Absyntax projects you want to invoke. It is not necessary for all projects to require input data and emit output data, and there may be good reasons why you might want to invoke a project that either requires no input data or emits no output data. In any case you must specify whether and how input data are acquired and presented to a project prior to invocation, and whether and how any output data emitted by the project are to be handled.

Time Limit

Each project invocation rule enforces a mandatory time limit. This limit is used to constrain the amount of time that an invoked project has to complete its work. It also ensures that badly formed projects are terminated in a timely manner. The minimum time limit is one second; the default is ten seconds; the maximum is 999 days.

Handling Data

Every Absyntax project has an entry-point and an exit-point. However, it is up to a project's creator to determine whether said entry- and exit-points transmit signals or data. You are expected to know these details for all Absyntax projects you want to invoke using the Absyntax Excel Add-in.

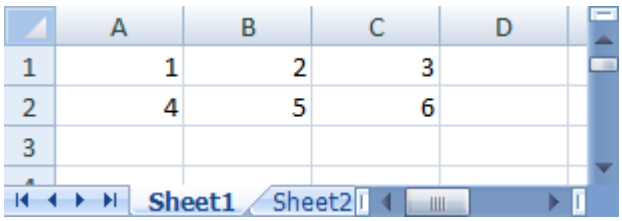
Input Data

If a project to be invoked has an entry-point that requires data, you must define a project invocation rule that specifies the worksheet cell range from which data is to be acquired prior to each invocation. An attempt to invoke a data input project without supplying data to it will result in an error. An attempt to invoke a signal input project with supplied data will result in the data being ignored.

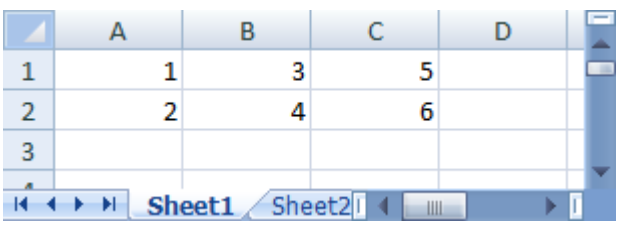
The input data range may refer to a single cell or multiple cells. If it refers to a single cell then the data type of the cell's value must be convertible to the type of data required by the target project's entry-point. If, on the other hand, it refers to multiple cells then the target project's entry-point data type must be either `System.Object` or an enumerable type (such as `System.Collections.IEnumerable`, `System.Collections.Generic.IEnumerable<>` or a one-dimensional array object). Furthermore, if the cells contain values that have different data types, the target project's entry-point data type can only be a strongly typed enumerable if the element type is `System.Object`. Examples of this include `System.Object[]` and `System.Collections.Generic.IEnumerable<System.Object>`.

See the section on data conversion for more details.

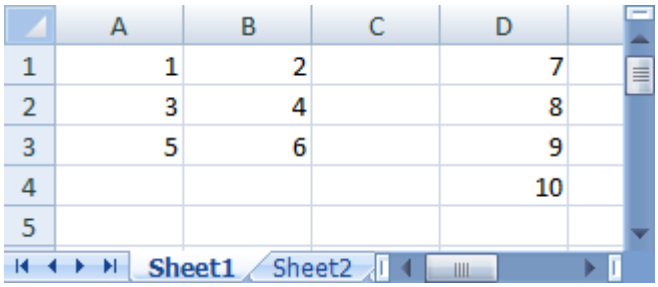
Where an input data range defines multiple cells, the values of these cells are presented to the project's entry-point as a single collection in a sequence determined by the input range order, which can be either "by row" or "by column". For example, an input range defined as A1:C2 and with a "by row" range order would result in data being presented to the target project in the order indicated by the cell values shown below.



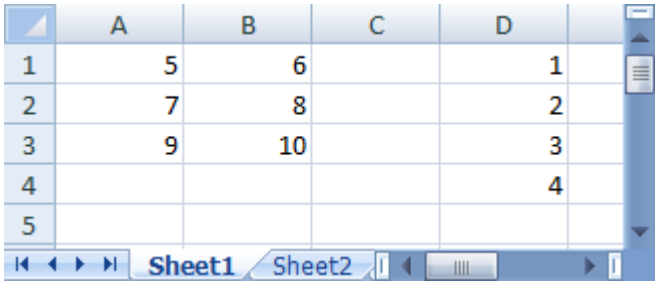
Contrast this with the same range and a "by column" range order. This would result in data being presented to the target project in the order indicated by the cell values shown below.



Where an input data range refers to a worksheet range name, the cells therein may occupy multiple worksheet areas. In such cases the values of these cells are obtained in a sequence determined firstly by the order in which the areas are defined within the range, and secondly by the input range order. In the example below, a named range consists of two areas defined in the order A1:B3 and D1:D4. Assuming a "by row" input range order, this would result in data being presented to the target project in the following sequence.



If, on the other hand, the areas had been defined in reverse order (D1:D4 and A1:B3), data would be presented as follows.

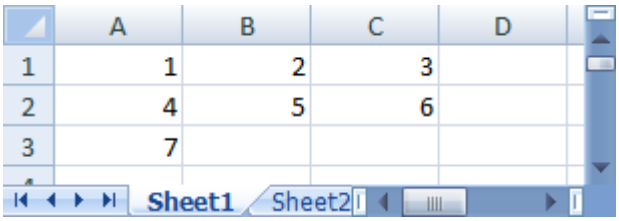


Output Data

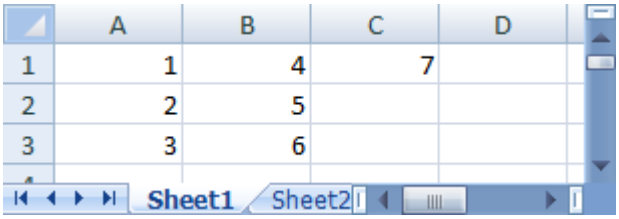
A project may emit data but you can choose to ignore it by suitably configuring the project invocation rule. In such cases all returned data is discarded and no worksheet is updated. Equally, it is acceptable (if, perhaps, confusing) to configure the rule to handle output data for a project that does not emit data. An attempt to invoke such a project will not result in an error but it will result in a single, empty value being written to the top-left cell of the range defined for the selected output worksheet. If this is undesirable then configure the rule to ignore output.

If a rule is configured to handle output data and the project emits non-enumerable data, the Absyntax Excel Add-in will attempt to write a single value to the top-left cell of the first area of the defined output range. If, instead, the project emits enumerable data then an attempt will be made to write each enumerable value to consecutive cells in the defined output range, in accordance with the specified output range order. The defined range does not need to be sufficiently large to accommodate all output values because the Absyntax Excel Add-in will extend the range as necessary. This does, of course, mean that cells outside the defined output range may be overwritten and you should allow for this when identifying the location and ordering of your output range.

For example, if the defined output range is an area of three columns and two rows (i.e. six cells) and the project outputs an array of seven values, all seven values will be written. This is achieved by extending the range in a direction determined by the value of the specified output range order.
In fact, the output data range needs only to be defined in terms of 1 x n columns or n x 1 rows. Consider an output range defined as "A1:C1" with a "by row" output range order: this would result in the first three output values being written to cells A1 through C1, the next three being written to cells A2 through C2 and so on, as shown below.

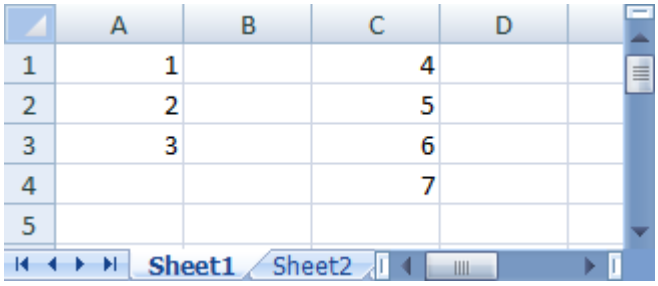


Similarly, an output range defined as "A1:A3" with a "by column" output range order would result in the first three output values being written to cells A1 through A3, the next three being written to cells B1 through B3 and so on.



It is also possible to write output values to non-contiguous cells or cell areas. To do this, define a named range in Microsoft Excel that cross-references the target cells, then select the range name as the output data range for the project invocation rule. The Absyntax Excel Add-in will write any output values to the range areas in the order they are defined within the range while honouring the rule's defined output range order. Note that only the last range area is extended (if necessary): in such cases the behaviour is as described above.

For example, suppose a project invocation rule's output data range refers to a worksheet range name defined as encompassing two cell areas, A1:A3 and C1:C3 (in that order). If the project outputs an array of seven values, all seven values will be written as shown below. Note that the first area (A1:A3) is filled first, and the last area (C1:C3) is extended to accommodate the extra output values.



Note that although Microsoft Excel allows you to define named ranges that span multiple worksheets, the Absyntax Excel Add-in does not recognise such ranges.

Data Conversion

Converting Input Data

An Absyntax input data project has an entry-point that will accept a single object of a specific type. Every time a project invocation rule is invoked that defines a requirement for input data, the Absyntax Excel Add-in is faced with the task of fetching and packaging this data into a form that will be accepted by the target project.

In Microsoft Excel, the value of any non-empty cell in a project invocation rule's input data range that is not a character string or a Boolean will be passed by the add-in to the target project as a double-precision floating-point number, irrespective of any cell formatting. If your project needs to work with other types of numeric value (e.g. integers) or date/time values then it must have an entry-point that supports either a double-precision floating-point number (`System.Double`) or a collection of such numbers and it must convert these numbers as necessary.

Furthermore, while it is sometimes possible to convert single values from one type to another, the same cannot be said for converting a collection of such values. If you do not wish to be concerned with the nuances of packaging multiple cell values from Microsoft Excel into an object that can be converted to the data type of an Absyntax project's entry-point, you can define your project as having an entry-point data type equal to any of the following:

- `System.Array`
- `System.Object[]`
- `System.Collections.IEnumerable`
- `System.Collections.Generic.IEnumerable<System.Object>`

Such projects guarantee that they will be able to receive one or more values of different types from the Absyntax Excel Add-in. The disadvantage of this approach is that your project will need to perform conversions so that it can then process each item correctly. However, this is the only choice when you need to input different types of data to a project.

If all cell values in the input data range are of the same type (i.e. either Boolean, text or numeric), you can define your project as having a "strongly typed" enumerable entry-point such as:

- `System.Double[]`
- `System.Collections.IEnumerable<System.String>`

Empty Cells

An empty Microsoft Excel cell is a cell for which no value has been specified. The way in which empty cells in a project invocation rule's input data range are handled by the Absyntax Excel Add-in depends on whether there are multiple cells in the range and the types of the non-empty cell values.

Single-cell range?	Non-empty cells contain text?	Non-empty cells contain other data?	Action
Yes	-	-	The range contains one empty cell. An attempt will be made to pass a single null value to the target project. This currently results in an error (missing startup data).
No	No	No	All cells are empty. An attempt will be made to pass an object array of null references to the target project.
No	Yes	No	Cells are either empty or contain only text. An attempt will be made to pass a string array of all cell values, null or otherwise, to the target project.
No	No	Yes	Cells are either empty or contain only non-text values. If all non-text values are of the same type (i.e. they are all either numeric or Boolean) then an attempt will be made to pass an array of that type, *excluding* empty values, to the target project. If the non-text values include both Boolean and numeric types then an attempt will be made to pass an object array of all cell values, null or otherwise, to the target project.
No	Yes	Yes	An attempt will be made to pass an object array of all cell values, null or otherwise, to the target project.

Conversion Strategy

The Absyntax Excel Add-in does not know what a project's entry-point data type is (or even if a project requires input data at all). This means that, initially, it must guess before packaging worksheet data and invoking a project with it. If Absyntax is unable to convert the presented data, the add-in will try packaging the data in up to two alternative ways, re-invoking the project each time until the project accepts the data. This sequence of attempts is referred to as the conversion strategy. The add-in remembers the form of the data for the successful conversion and uses it the next time the same rule is invoked. If Absyntax rejects all attempts to convert the data, this is considered an error.

The following table defines the various strategies that the add-in can employ when attempting to package one or more cell values into an object that will be accepted by the target project's entry-point.

Last Form of Data	Single-cell range?	Conversion Strategy
None	Yes	1. Single value 2. Strongly-typed array 3. Object array
None	No	1. Strongly-typed array 2. Object array
Single value	Yes	1. Single value 2. Strongly-typed array

		3. Object array
Single value	No	1. Strongly-typed array 2. Object array
Strongly-typed array	Yes	1. Strongly-typed array 2. Single value 3. Object array
Strongly-typed array	No	1. Strongly-typed array 2. Object array
Object array	Yes	1. Object array 2. Single value 3. Strongly-typed array
Object array	No	1. Object array 2. Strongly-typed array

Note that if a project's entry-point data type is changed and the associated project invocation rule is configured to reload the project, there is no guarantee that the last successful form of data will still work and the add-in may have to cycle through the relevant strategy when it next invokes the rule. It is also worth noting that the same rule may be changed to target a different project, for which there is also no guarantee.

Single-string Input Data Projects

Absyntax projects with entry-point data types equal to `System.String` can sometimes appear to behave strangely when they are supplied with data from Microsoft Excel. If you create a project invocation rule for such a project and said rule defines an input data range consisting of multiple cells, upon invocation the target project will receive a string value equal to one of the following:

- "Boolean[] Array"
- "String[] Array"
- "Double[] Array"
- "Object[] Array".

With reference to the conversion strategy described above, this is because the Absyntax Excel Add-in attempts to pass to Absyntax either a strongly-typed array or an object array (depending on the last form of data and the data types of the various cell values). Because *any value is convertible to a string*, Absyntax accepts the incoming array and presents to the project the name of the array's data type (because that is how arrays are converted to strings).

This kind of behaviour is rarely required, so ensure that your project invocation rule's input data range extends only to a single cell when targeting single-string input data projects.

Receiving Output Data

Microsoft Excel will reject attempts to write data items to worksheet cells if any item is of a type that Microsoft Excel does not recognise. However, Microsoft Excel is proficient at receiving string values and treating them flexibly. For example, if the string representation of the Boolean value "True" is written to a cell, it will be stored as a Boolean value. Likewise if the string representation of a numeric value is written to a cell, it will be stored as a numeric value. For this reason, when requiring an Absyntax project to output data, it is advisable to use a project that defines its exit-point data type as either of the following:

- `System.String[]`
- `System.Collections.Generic.IEnumerable<System.String>`

The latter form is recommended as it is more versatile from the perspective of Absyntax project development and maintenance.

Outputting project data as strings has an added bonus in that the string representations of date/time values are automatically formatted when written to worksheet cells, meaning that a date/time appears as it is meant to rather than as a double-precision floating-point number that needs subsequent, manual formatting.

Executing a Run

When you execute a run you are instructing the Absyntax Excel Add-in to invoke all configured, valid and enabled project invocation rules. Depending on the chosen execution mode, rules are invoked either synchronously or asynchronously. Synchronous rules are invoked in the order in which they are defined.

By invoking rules synchronously you allow the output of each target Absyntax project to be written to a Microsoft Excel worksheet before the next rule is invoked, meaning that the next target project can, if required, be supplied with data updated as a result of a previous invocation. By invoking rules asynchronously, on the other hand, you can execute a run more quickly because all rules are invoked at the same time. Asynchronous runs can be particularly useful if you have several long-running projects to invoke or you have several projects which are configured to reload before each invocation.

To execute a run you must have a suitable, active Absyntax licence.

Loading and Unloading a Project

The Absyntax Excel Add-in is designed to facilitate the speedy invocation of Absyntax projects. To do this it makes use of the capabilities of the Absyntax framework to load projects into computer memory and, once they are loaded, invoke them on demand. This approach avoids the time-consuming overhead of unnecessarily reloading a project from file.

The first time a project invocation rule is invoked, Absyntax will need to load the target project before invoking it. When a rule is invoked subsequently, the Absyntax Excel Add-in determines whether any of the rule's material details (specifically, its time limit and the project path) have changed since the last time it was invoked. If they have, or if the rule is configured to reload the target project, and if there is a loaded project associated with said rule, the currently associated project is unloaded before an attempt is made to load the rule's target project. Equally, when a run is executed and the add-in identifies that a loaded project no longer has an associated rule (because the rule has been deleted), the orphaned project will be unloaded.

The following situations will also result in a loaded project being unloaded.

- an invocation fails to supply the project with required data;
- an invocation fails to convert supplied data to to a type required by the project's entry-point;
- an invocation does not complete in the time allotted by the associated project invocation rule;
- a run is aborted while a project invocation is in progress;
- there are loaded projects associated with a Microsoft Excel workbook that is closed;
- the Absyntax host process is terminated (manually via the Windows Task Manager, for example).

Project State

Subject to the above, a loaded Absyntax project will remain in computer memory between invocations. This means that the state of all of a project's contained features will remain intact. If a project has not been designed with repeated invocation in mind, this can cause unexpected behaviour upon subsequent invocation. If you want a target project to behave as it would if it was repeatedly loaded, invoked and unloaded then you may need to design it to initialise or reset itself.

For example, consider a project that increments a counter periodically during an invocation. If the counter is not reset at any time then its value is cached and the next invocation will make use of this cached value. This may, of course, be the desired behaviour but often it is not. Be mindful of project state when you design projects for use with the Absyntax Excel Add-in.

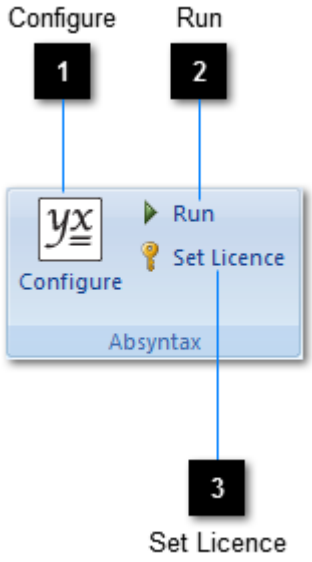
Multiple Rules, Same Project Path

It is possible for multiple project invocation rules to target the same Absyntax project. Note that each such rule, when invoked for the first time, will result in a separate instance of the project being loaded. In other words, project instances are not shared between rules. In turn, this means that instances of the same project may have different states.

Aborting a Run

A run may be aborted at any time prior to its conclusion. Be aware that if you do abort a run, you risk leaving your Microsoft Excel workbook in an indeterminate state because some data may already have been written (in the event that the run is invoking multiple projects).

User Interface

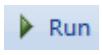


The Absyntax Excel Add-in appears as a group of buttons in the Microsoft Excel ribbon's **Data** tab.

1 Configure

Click this button to open the Absyntax Project Configuration dialogue window and maintain project invocation rules.

2 Run



Click this button to execute a run, invoking all configured, valid and enabled project invocation rules. The Absyntax Execution Coordinator dialogue window is displayed.

When this button is disabled, its super-tip will give clues as to why it is disabled. Hover the mouse over the button to reveal the super-tip.

If you see...	Then...
The host process is not yet open. This may indicate an issue with your Absyntax licence.	Depending on how much time has passed since Microsoft Excel was started, this can mean that no suitable, active Absyntax licence was found.

	If you have an active third-party execution licence, this may mean that the client identifier either has not been specified or is incorrect. Click the Set Licence button in order to specify the correct identifier. Alternatively, if you have recently started Microsoft Excel (i.e. within the last few seconds) then wait a little longer for the Absyntax host process to open.
Please wait while the host service starts.	This indicates that the Absyntax host process is open but the runtime service has yet to start. It should normally start within a few seconds of the host process opening.
No valid, enabled projects have been defined. Use the Configure button to create and maintain project rules.	This means that the Absyntax host process is open and the runtime service has started, but the active workbook does not have a single project invocation rule that is both valid and enabled. In other words, there is nothing to invoke.

3 Set Licence

Click this button to open the Absyntax Licence dialogue window and specify which Absyntax licence to use.

If you want to prevent users from accessing this dialogue, you must change the add-in's configuration file. To do this, navigate to the add-in's installation directory and open the file "AbsyntaxExcelAddIn.dll.config" using a plain-text editor such as Notepad. Then change:

```
<add key="AllowLicenceAccess" value="True" />
```

to

```
<add key="AllowLicenceAccess" value="False" />
```

Save the file and restart Microsoft Excel.

Absyntax Project Configuration

The Absyntax Project Configuration dialogue window allows you to define the Absyntax projects to be invoked during a run.

1 Toolbar

The toolbar supports actions in respect of individual project invocation rules.

2 Project invocation rule list

The list of defined project invocation rules for the active Microsoft Excel workbook.

3 Selected rule

The selected project invocation rule is the target for specific toolbar operations.

4 Disabled rule

Disabled project invocation rules are highlighted with a maroon background. Such rules are ignored during a run.

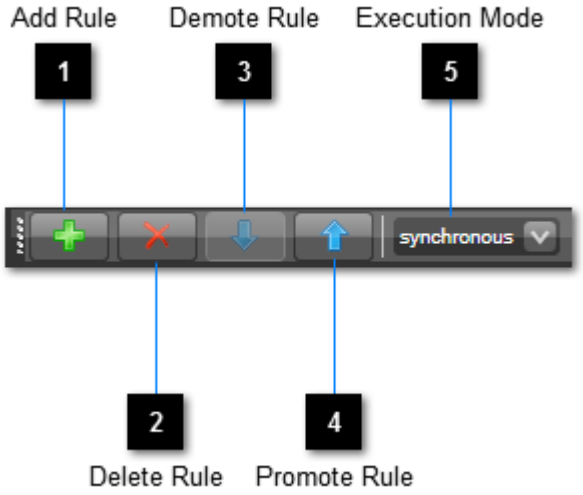
5 OK

Commits all changes to the active Microsoft Excel workbook and closes the dialogue window. If you subsequently save the workbook, all project invocation rule data is saved with it.

6 Cancel

Discards any changes and closes the dialogue window.

Absyntax Project Configuration toolbar



1 Add Rule

Adds a new, default project invocation rule to the bottom of the list.

2 Delete Rule

Deletes the selected project invocation rule.

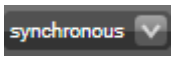
3 Demote Rule

Demotes the selected project invocation rule. Note that rule ordering is only relevant when the execution mode is set to "synchronous".

4 Promote Rule

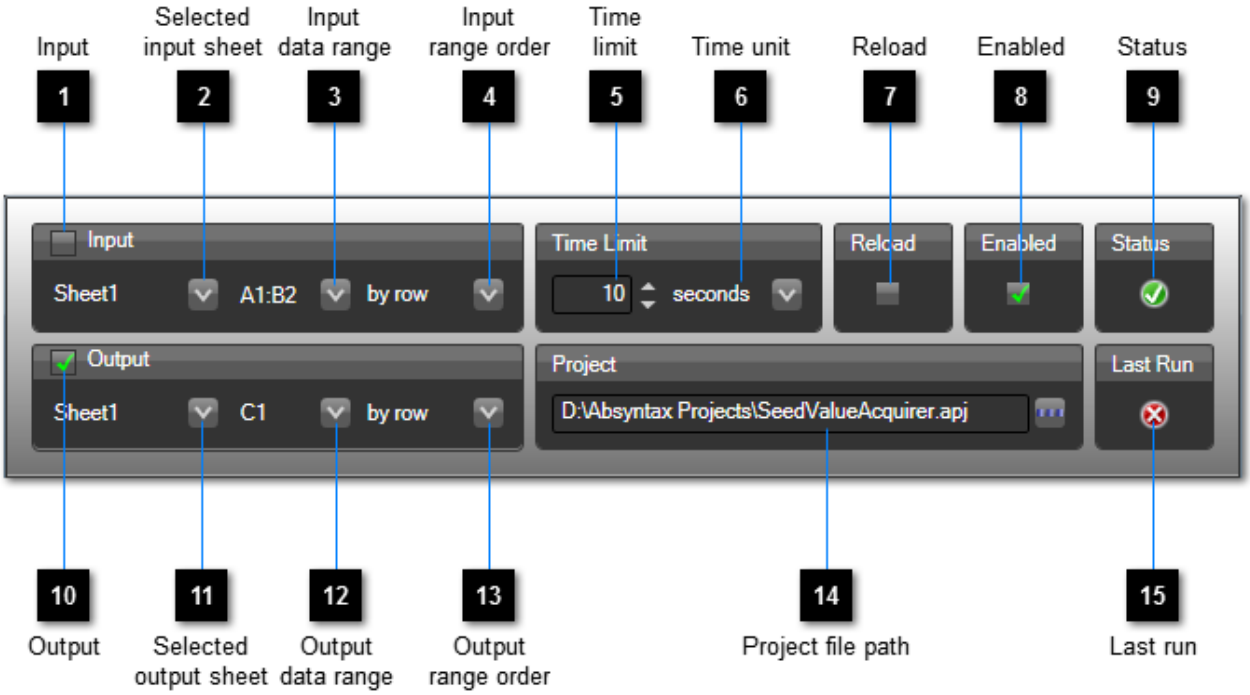
Promotes the selected project invocation rule. Note that rule ordering is only relevant when the execution mode is set to "synchronous".

5 **Execution Mode**

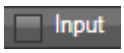


Determines whether all valid, enabled projects will be invoked synchronously or asynchronously.

Anatomy of a project invocation rule

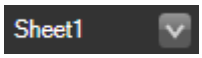


1 Input



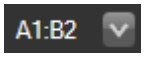
Check this box if data is to be acquired from a worksheet in the active workbook and passed to the target project upon invocation.

2 Selected input sheet



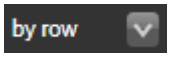
The worksheet from which input data will be acquired (if the Input check box is checked) prior to invoking the target project.

3 Input data range



Defines the range of cells belonging to the selected input worksheet from which input data will be acquired (if the Input check box is checked) prior to invoking the target project. This can be either a single cell (e.g. "A1"), a pair of cells defining the diagonally opposed corners of a single area (e.g. "A1:B2") or a range name defined in the active Microsoft Excel workbook.

4 **Input range order**

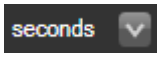


Defines the order in which worksheet cell data from the input data range is presented to the target project (if the Input check box is checked).

5 **Time limit**

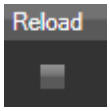
A value which, when combined with the time unit, determines the amount of time that Absyntax will allow for a project invocation to complete before terminating the invocation. The minimum value is 1, the maximum value is 999.

6 **Time unit**



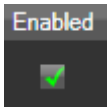
A quantity of time which, when combined with the time limit, determines the amount of time that Absyntax will allow for a project invocation to complete before terminating the invocation.

7 **Reload**



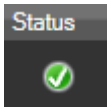
When checked, the target project is reloaded from file prior to the next invocation.

8 **Enabled**



When checked, an attempt will be made to invoke the target project on the next run (assuming it is in a valid state – see Status). When unchecked, the target project will not be invoked.

9 **Status**

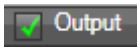


Indicates the validity of the project invocation rule.

- Indicates that all information is correct and the target project file exists.
- Indicates one or more of the following:

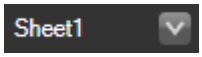
- there is no selected input sheet and the Input check box is checked;
- the input data range is invalid and the Input check box is checked;
- there is no selected output sheet and the Output check box is checked;
- the output data range is invalid and the Output check box is checked;
- the file path of the target Absyntax project is either blank or does not point to an existing file.

10 Output



Check this box if any output data emitted by the project is to be written to a worksheet in the active workbook. Uncheck it if no project output data is to be written.

11 Selected output sheet

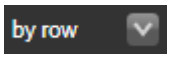


The worksheet to which data emitted by the invoked project will be written (if the Output check box is checked).

12 Output data range

Defines the range of cells belonging to the selected output worksheet to which data emitted by the invoked project will be written (if the Output check box is checked). This can be either a single cell (e.g. "A1"), a pair of cells defining the diagonally opposed corners of a single area (e.g. "A1:B2") or a range name defined in the active Microsoft Excel workbook.

13 Output range order

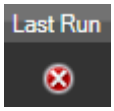


Defines the cell order in which data emitted from the invoked project is written to the output data range (if the Output check box is checked).

14 Project file path

The full path of the file containing the Absyntax project to be invoked. Click the ellipsis button to open a file browser dialogue that will help you find Absyntax projects.

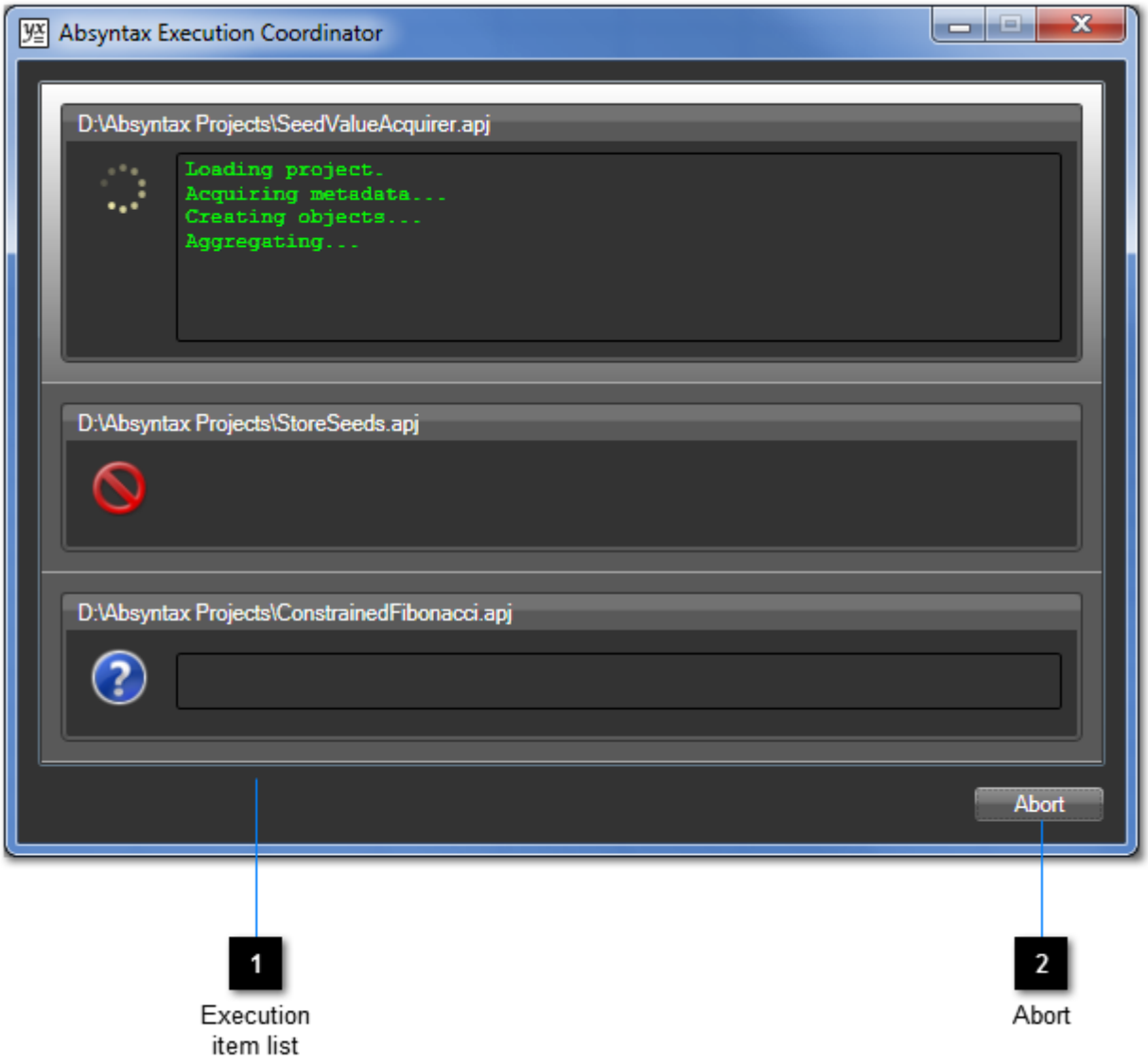
15 Last run



Indicates the outcome of the last invocation of the target project.

- Indicates a successful invocation.
- Denotes a project that was not invoked.
- Indicates that a non-specific problem arose during invocation.

Absyntax Execution Coordinator



The Absyntax Execution Coordinator dialogue window is opened whenever you execute a run. It provides feedback for Absyntax projects while they are being invoked. It also indicates the pre- and post-invocation state of each item.

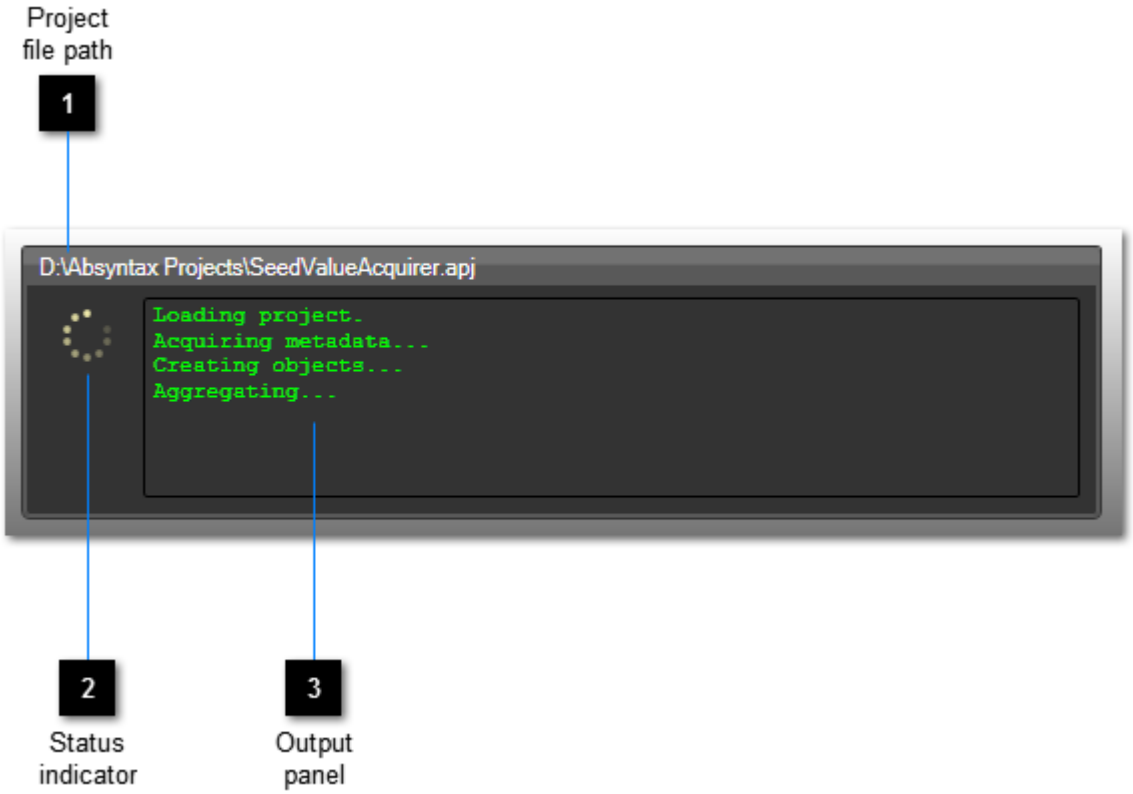
1 Execution item list

Displays a list of items representing project invocation rules that either have been invoked, are in the process of being invoked, are awaiting invocation or will not be invoked.

2 Abort

Click this button to abort the current run. When the run concludes, this button's caption changes to "Close".

Anatomy of an execution item



1 Project file path

D:\Absyntax Projects\SeedValueAcquirer.apj

Shows the full file path of the file containing the target Absyntax project.

2 Status indicator

Indicates the state of the execution item. The symbol shown indicates that invocation of the target project has started and has not yet finished. Other symbols are:

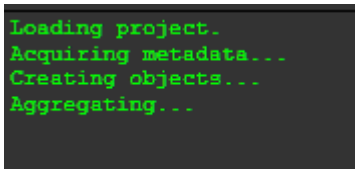
- Indicates that a project will not be invoked because the associated project invocation rule is either disabled or invalid.
- Indicates that a project is awaiting invocation.
- Indicates that a project was invoked successfully.
- Indicates any of the following:

- invocation timed out before the project was able to signal its exit-point;
- invocation was aborted;
- a project was invoked successfully but not all of its output data could be written to the target Microsoft Excel worksheet (because either the output data included an item of a type that Microsoft Excel did not recognise or because there was insufficient room in the worksheet).

Indicates that an attempt was made to invoke a project but errors occurred during the operation such as:

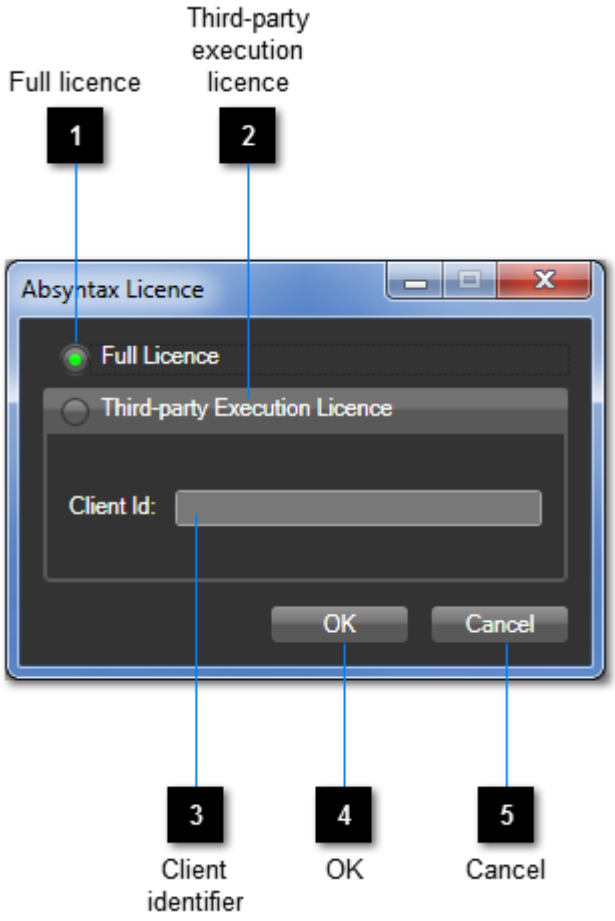
- Absyntax could not load the target project;
- required input data was missing;
- there was a problem converting the supplied input data.

3 **Output panel**



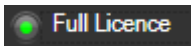
Displays all output messages received from the target project, together with any messages issued by the Absyntax Excel Add-in for information.

Absyntax Licence



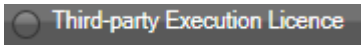
The Absyntax Licence dialogue window allows you to identify the type of Absyntax licence to be used by the Absyntax Excel Add-in. Note that the information managed by this dialogue is saved under your roaming user account.

1 Full licence



Select this option if you want the Absyntax Excel Add-in to make use of a full Absyntax licence. This is the default option.

2 Third-party execution licence



Select this option if you want the Absyntax Excel Add-in to make use of a third-party execution licence. The Client Id text box is enabled only when this option is selected.

3 Client identifier

When the third-party execution licence option is selected, use this field to enter the unique client identifier you were supplied with when you acquired your third-party execution licence. You will not be able to click the OK button until you have entered a syntactically valid identifier.

4 OK

Commits all changes and closes the dialogue window.

5 Cancel

Discards any changes and closes the dialogue window.

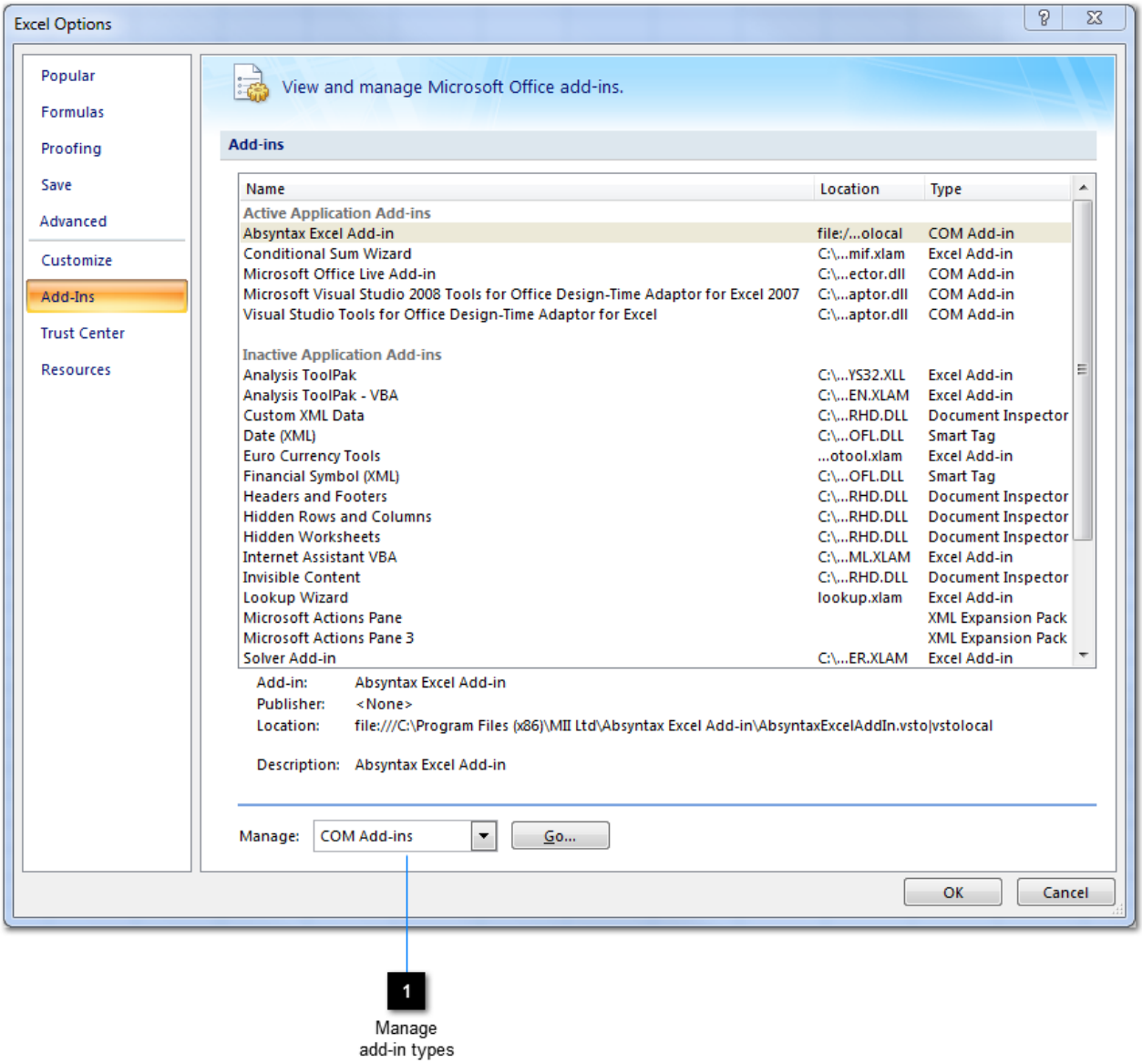
Warnings and Errors

The following table explains some of the warning and error messages that may appear in the output panel of an execution item.

Message	Description
⚠ *The project has aborted.*	This indicates that you aborted the project while it was being invoked.
⚠ *The project has aborted. The operation has timed out.*	This indicates that the project invocation rule's time limit has been exceeded, resulting in the project being aborted by the Absyntax Excel Add-in. It may be that insufficient time has been allocated to the operation. Alternatively there may be a problem with the project's design: you should ensure that all of the project's execution paths result in its exit-point being signalled.
⚠ *Failed to write output data to* [sheet]![range].	This indicates either that the invoked project returned a data item of a type unrecognised by Microsoft Excel, or that there was insufficient space in the target worksheet in which to write all the returned data. If an unrecognised data type is the problem, change the project so that it emits only Microsoft Excel-compatible data (refer to the section on data conversion). Otherwise, change the output data range so that it is further away from either the rightmost or bottommost edge of the target worksheet (in accordance with the project invocation rule's output range order).
⚠ *The project has ended.*	This typically indicates that, as part of an invocation, a project required loading but was aborted before it could be invoked. Absyntax considers an abort request as a request to unload a loaded project. If a project is not actually being invoked at the time the unload request is received then no "aborted" message is issued.
⊗ *The project failed to start because required startup data was not supplied.*	This indicates that either the associated project invocation rule's Input check box is unchecked or the input data range identifies a single cell whose value is empty.
⊗ *The project failed to start because the startup data could not be converted to the type required by the project's entry point.*	Refer to the sections describing data handling and data conversion.

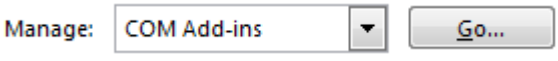
The byte source could not be deserialised.	When Absyntax is attempting to load a project, this indicates one of the following problems: • the target project file is not an Absyntax project file; • the target project file contains a version of an Absyntax project that is no longer supported; • the target project file contains references to types that cannot be found in the Absyntax framework. The last of these may indicate a missing Absyntax add-in (not to be confused with a Microsoft Excel add-in). You should contact the project's owner to find out whether the project uses Absyntax add-ins and, if it does, ensure that these add-ins are available to Absyntax by locating them in the Absyntax installation directory's AddIns sub-folder.
The communication channel to the service host has faulted.	This indicates that the Absyntax host process has closed unexpectedly. It is designed to restart in such circumstances, but you may need to wait a few seconds before being able to execute another run.
The server did not provide a meaningful reply; this might be caused by a contract mismatch, a premature session shutdown or an internal server error.	See above.

Excel Options



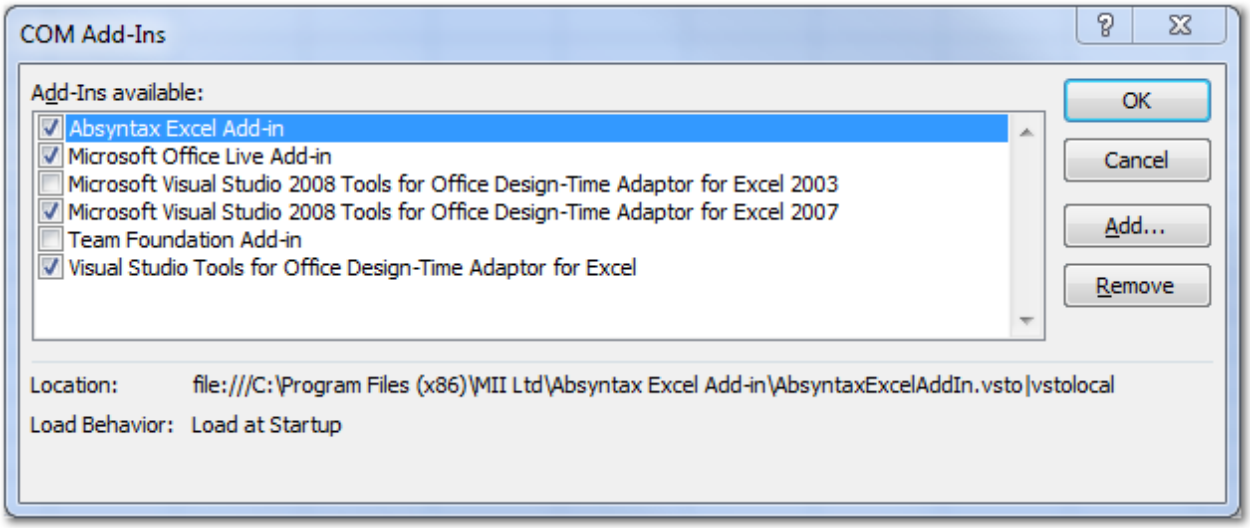
To show the Excel Options dialogue window, from the **Office Button** select **Excel Options**. Then select Add-Ins from the left-hand tab panel. You can use the Add-ins list to see whether the Absyntax Excel Add-in is loaded. If it is, there will be an entry for it in the Active Application Add-ins group.

1 Manage add-in types



To show the list of installed COM add-ins (of which the Absyntax Excel Add-in is one), select "COM Add-ins" and click the Go button.

COM Add-Ins



In order for the Absyntax Excel Add-in to work, it must appear in the the list of available COM add-ins and it must be checked. You can open this dialogue from the Excel Options dialogue.

