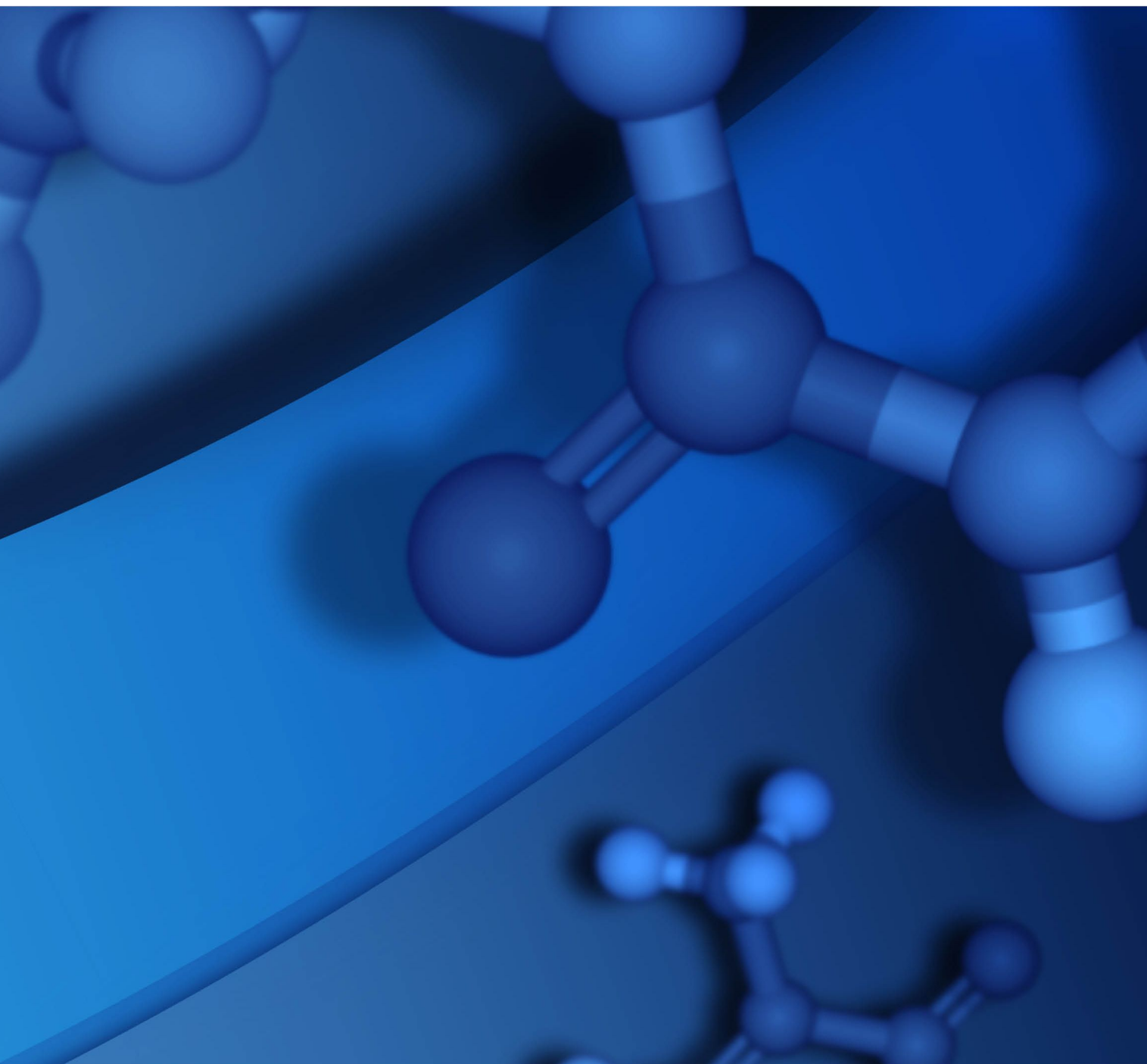


USER GUIDE

MATERIALSSCRIPT

8.0



Copyright Notice

©2014 Dassault Systèmes. All rights reserved. 3DEXPERIENCE, the Compass icon and the 3DS logo, CATIA, SOLIDWORKS, ENOVIA, DELMIA, SIMULIA, GEOVIA, EXALEAD, 3D VIA, BIOVIA and NETVIBES are commercial trademarks or registered trademarks of Dassault Systèmes or its subsidiaries in the U.S. and/or other countries. All other trademarks are owned by their respective owners. Use of any Dassault Systèmes or its subsidiaries trademarks is subject to their express written approval.

Acknowledgments and References

To print photographs or files of computational results (figures and/or data) obtained using BIOVIA software, acknowledge the source in an appropriate format. For example:

"Computational results obtained using software programs from Dassault Systèmes Biovia Corp..
The *ab initio* calculations were performed with the DMol³ program, and graphical displays generated with Materials Studio."

BIOVIA may grant permission to republish or reprint its copyrighted materials. Requests should be submitted to BIOVIA Support, either through electronic mail to support@accelrys.com, or in writing to:

BIOVIA Support
5005 Wateridge Vista Drive, San Diego, CA 92121 USA

Contents

Scripting in Materials Studio	1	Running in legacy standalone mode	19
Introduction	1	Upgrading legacy scripts	19
Further Information	1	Dialogs in scripting	21
Working with scripts in the Materials Visualizer	2	Find and Replace dialog	21
Writing scripts	2	Script Viewer Options dialog	21
Generating scripts	2	General tab	22
Tips and Tricks	3	Printing tab	23
Checking script syntax	3	Colors tab	23
Debugging scripts	3	Script Job Control dialog	24
Keyboard actions in the Script Viewer	4	Script Job Control Options dialog	25
Adding custom menus to run scripts	5	Script Library dialog	26
Setting up a document library	5	Library tab	26
Setting up custom menus	6	Add From Project dialog	27
Sharing custom menus with other users	6	User Menu tab	28
Running scripts from the User menu	6	Choose Document dialog	29
Using a custom menu command which		Import Menu dialog	29
requires no arguments	6	Export Menu dialog	30
Using a custom menu command which		Define Arguments dialog	30
requires arguments	6	Specify Arguments dialog	31
Defining arguments for scripts	7		
Setting up a script to receive arguments	7		
Setting up a custom menu command to			
provide arguments to a script	8		
Using a custom menu command which			
requires arguments	8		
Running scripts on a server	9		
A sample remote scripted job	9		
Using job control in scripting	10		
Launching a job	11		
Job completion information	11		
Managing live updates	11		
If a remote scripted job fails	12		
Running MaterialsScript in standalone mode	15		
Running multiple standalone			
MaterialsScript scripts in the same directory	17		
Running scripts in standalone mode in			
the current directory	18		
Running multiple scripts sequentially or			
concurrently in standalone mode	18		
Sequential running	18		
Windows	18		
Concurrent running	18		
Linux	18		
Windows	18		
Running MaterialsScript in legacy			
standalone mode	19		

Scripting in Materials Studio

Introduction

The Materials Visualizer provides a rich user interface for working with Materials Studio documents and servers. Sometimes, however, it may be desirable to control Materials Studio programmatically, for example to:

- Automate repetitive tasks
- Link several tasks or calculations together in sequence
- Perform calculations that are not available from any of the Materials Studio modules
- Integrate with other software

The MaterialsScript Application Programming Interface enables you to perform tasks such as those listed above by allowing you to manipulate Materials Studio using scripts.

Further Information

For more information about the Materials Studio and other Accelrys software products, visit BIOVIA Support on the Web: <https://community.accelrys.com/index.jspa>

Working with scripts in the Materials Visualizer

The Script Viewer provides an editing environment for script documents within the Materials Visualizer, enabling you to write, edit, and debug your scripts before [submitting them to a Materials Studio server](#). This integrated editing environment allows you to rapidly develop and test scripts until you are confident that they function correctly, all from within the Materials Visualizer.

Writing scripts

Currently, Materials Studio only supports scripts written in Perl. Materials Studio uses an embedded interpreter to run Perl scripts (.pl), so in addition to referencing the MaterialsScript Application Programming Interface (API) Materials Studio scripts can also perform any other functions that are possible in Perl.

Although the Materials Visualizer provides an integrated script editing environment through the Script Viewer, you can use any text editor to create your Materials Studio scripts.

You can also [generate](#) script sections from the Materials Visualizer.

Script documents created within or imported into Materials Studio projects are displayed using the Script Viewer. As well as allowing you to perform standard text editing functions such as cutting, copying, pasting, deleting, etc., the Script Viewer also has a number of features that make it easier for you to write and edit scripts in the Materials Visualizer.

The text of your script can be colored according to the syntax categories of the scripting language being used, making your script easier to read and any errors easier to find. To turn on syntax coloring, check the *Use syntax coloring* checkbox on the [General tab](#) of the Script Viewer Options dialog. You can specify the syntax coloring scheme, along with other elements of Script Viewer rendering, on the [Colors tab](#).


There are a number of formatting options available, including the choice of font, the ability to show or hide white space and line numbers, and automatic indenting of new lines. Toggle these options on and off using the checkboxes on the *General* tab of the Script Viewer Options dialog or, alternatively, use the commands on the Edit and View menus.

Bookmarks are used to mark particular locations in the script, for example, lines containing syntax errors or the point at which a script was stopped; you can also add your own bookmarks manually. Use the commands on the *Edit* menu to toggle bookmarks on and off and to navigate through the script using the bookmarks.

Generating scripts

You can generate sections of MaterialsScript from dialogs in the Materials Visualizer. This captures the current settings on a given dialog and ensures that the generated script will mimic the equivalent behavior of the Materials Visualizer.

To generate script from a dialog

1. Open a dialog with any button that has a *Copy Script* option.
2. Configure the settings on the dialog as required.
3. Click the arrow for the button  and select *Copy Script* from the menu.
The code is now copied to the system clipboard.
4. Open a new or existing Script document.
5. Place the cursor where the code should be pasted.
6. Press the CTRL + V keys or right-click and select *Paste* from the shortcut menu.


A section of script representing the settings on the dialog is pasted into the script document.

Tips and Tricks

- Most generated sections of script make use of variables such as `$doc`. Before running the script, you must define these variables or replace them with your own.

Checking script syntax

Once you have [written](#) a script, it is advisable to check the syntax. Eliminating any syntactic Perl errors, such as unmatched brackets or a missing semicolon at the end of a line, at this stage will speed up the [debugging](#) process, enabling you to concentrate on eliminating any logical errors in your script. The Script Viewer allows you to check the syntax of a script without actually executing it. This enables you to quickly and easily find and correct any problems with the syntax. As Perl is a loosely bound programming language, the script interpreter will not detect MaterialsScript errors, such as invalid function names, until the script is run.

To check the syntax of a script, click the *Check Syntax* button  on the Scripting toolbar or select *Tools / Scripting / Check Syntax* from the menu bar. Any syntax errors are reported in the *Syntax Check* tab of the Script Viewer. For errors that are reported with line numbers, a red diamond-shaped bookmark appears next to the error report and another in the source pane marks the position of the error in the script. Double-clicking on these bookmarks allows you to navigate between the error report and the corresponding error in the script.

The Script Viewer can highlight syntax errors by underlining them with a wavy line, making it easier to detect and locate any errors in your script. Note that some errors lack the information that the Script Viewer requires in order to be able to underline the exact text giving rise to the error, so not all errors may be underlined. To turn on syntax error underlining, check the *Underline errors* checkbox on the [General tab](#) of the Script Viewer Options dialog. You can specify the underlining color, along with other elements of the Script Viewer rendering scheme, on the [Colors tab](#).

The error bookmarks and underlining are automatically cleared when you debug the script or check the syntax again.


Even if your script contains no syntax errors, it might contain other kinds of errors that can only be found when you debug the script or [run it on a server](#). Typically, these 'runtime errors' are caused by logic errors in the script, for example, attempting to use a property or method that is not applicable to a given object.

Debugging scripts


After [checking the syntax](#) of your script, you should run your script to check for MaterialsScript errors. The Script Viewer allows you to execute scripts locally on the Materials Studio client rather than on a server, enabling you to see the output from your script as it runs and, thus, making debugging easier.


Typically, the types of errors that debugging (as opposed to syntax checking) will highlight are:

- Attempting to use an invalid property or method (for example, spelling the name incorrectly)
- Attempting to use a property or method that is not applicable to a given object (for example, trying to create an atom in a study table)
- Attempting to use a value that is invalid for a given property or method (for example, trying to open a document that does not exist)

To start debugging a script, click the *Debug* button  on the Scripting toolbar, or select *Tools / Scripting / Debug* from the menu bar, or press the F5 key. Materials Studio will then invoke the script

interpreter to execute your script on the client. Any documents your script creates or references are automatically opened in Materials Studio. However, you will normally only see changes to the document once the script finishes execution. To see any changes earlier, you need to call the `UpdateViews` function on the document. You can make your script write output to the *Output* tab of the Script Viewer using print statements. The output will appear as soon as the print statement is executed. You can also use print statements to print the values of variables, object properties, etc.

To stop a script while it is running, press the ESC key or click the  icon to the right of the progress indicator on the status bar. Documents will remain in whatever state they are in when the script stops running. Currently, it is not possible to resume execution of a script once it has been halted. Scripts are always run from the beginning.

Tip: You can undo the effects of running a script using the *Undo* button  on the Standard toolbar or by selecting *Edit | Undo Debug Script* from the menu bar. As with other functions in Materials Studio, using *Undo* will not delete any documents that were created by the script.

Keyboard actions in the Script Viewer

The Script Viewer provides a wide range of editing functions. In addition to the standard mouse and keyboard shortcuts for basic functions such as selection, deletion, navigation, etc., a number of keyboard shortcuts are available for more advanced editing functions in the Script Viewer.

The tables below lists various editing operations and their associated keyboard shortcuts.

To	Press
Get context-sensitive help for the word at the cursor	F1
Start debugging mode	F5
Move the cursor to the start of the line	HOME
Move the cursor to the end of the line	END
Move the cursor to the start of the document	CTRL + HOME
Move the cursor to the end of the document	CTRL + END
Move the cursor one word to right (wrapping at line endings)	CTRL + RIGHT arrow
Move the cursor one word to left (wrapping at line endings)	CTRL + LEFT arrow
Move the cursor to the topmost displayed line	CTRL + PAGE UP
Move the cursor to the bottommost displayed line	CTRL + PAGE DOWN
Cut the selected text to the clipboard	SHIFT + DELETE
Copy the selected text to the clipboard	CTRL + INSERT
Paste text from the clipboard	SHIFT + INSERT
Indent selected lines by one tab	TAB
Outdent selected lines by one tab	SHIFT + TAB
Find matched or unmatched pairs of parentheses, braces, or square brackets	CTRL +]

To	Press
Toggle between Insert and Overwrite mode	INSERT
Toggle whether cursor keys only scroll or also move the cursor	SCROLL LOCK

Adding custom menus to run scripts

You can add new items to the User menu of Materials Studio which run custom scripts on the active document in any open project. These User menu items which run scripts are referred to as commands. In order to create a new menu item you must:

- Make the required script document available through the library
- Configure the new command according to the requirements of the script it will run


Setting up a document library

1. Open the [Script Library dialog](#). The [Library tab](#) will be displayed by default.
2. To add a new general location where documents are stored, click the *Add* button in the *Location* section.
A new location will be added to the file list with the default name of "Untitled".
3. For the new location, enter the name to use in the *Name* textbox. The name in the file list will be updated.
4. In the path list click the *<click to add path>* text and enter the absolute location of a shared directory either on the current machine or on a network drive.
Any files or folders in the selected directory will be automatically added to the file list.

Tip: In order to ensure that the files are always available you may need to add multiple paths for the same location, particularly if it is likely that the path is different between server and client.

5. To create a new folder, select an existing location or subfolder and click the *Add* button in the *Folder* section. Enter in a name to use for the folder.
6. To add a file from the current project, select an existing location or subfolder and click the *Add...* button to open the [Add From Project](#) dialog. Select the file in the current project and click the *OK* button.

Setting up custom menus

1. Open the [Script Library dialog](#) and select the [User Menu](#) tab.
2. Optionally, click the *Folder* button to create a new folder to contain your new command. A new item will be added to the menu item list named "My Group" by default.
This is recommended if many commands will be accessed through the User menu and can be useful to group similar commands together.
3. Enter a name for the group to display on the User menu.
4. Select the group where a new item should be created and click the *Command* button. A new item will be added to the menu item list named "My Command" by default.
5. Enter a name for the command in the *Title* textbox and a description if required.
6. Click the  button to open the [Choose Document](#) dialog. Select a script and click the *OK* button.
7. Choose whether to run the command on [Client](#) or [Server](#). The icon displayed on the User menu for the command will indicate whether its script will be run locally or on a server.
8. Select the type of document that the script *Requires*, the command on the User menu will only be available when the appropriate type of document is active.

Tip: When preparing scripts for use by menu commands, you must ensure that the document which they act upon is not explicitly named in the script. Instead, use the `ActiveDocument` property. A script may also require other documents that are the same every time the script is run, for example, fragment documents. Such documents should be placed into the Library and imported in the script using a URL syntax, using the `Import` function.

9. Choose the files in the current project which the script *Uses*.

Sharing custom menus with other users

Custom *User menu* content can be shared with other users. The *Export...* and *Import...* buttons on the [User Menu](#) tab of the [Script Library dialog](#) allow you to create or import a file containing custom *User menu* items.

Tip: Commands shared with other users still reference script documents in your own library. In order to share scripts and menu commands effectively with other users, you must ensure that the library locations chosen reference a network folder with appropriate accessibility for the target users.

Running scripts from the User menu

Using a custom menu command which requires no arguments

1. Ensure that the target document is in focus.
2. Choose the custom command from the User menu.

Using a custom menu command which requires arguments

1. Ensure that the target document is in focus.
2. Choose the custom command from the User menu. The [Specify Arguments](#) dialog automatically opens.
3. Inspect and modify the defaulted argument values.
4. Click *OK* to run the script or *Cancel* to cancel the command.

Note: The set of available commands on the User menu varies according to the document in focus and the *Requires* rule specified on the [User Menu](#) tab of the [Script Library dialog](#). The command on the User menu will only be available when the appropriate type of document is active.

Defining arguments for scripts

Scripts run from the User menu of Materials Studio may be supplied with arguments to control their functionality.

In order to support arguments in a script you must:

- Include statements in the script to process the arguments
- Configure the command on the User menu to supply arguments to the script

Setting up a script to receive arguments

The Perl core module `Getopt::Long` (<http://perldoc.perl.org/Getopt/Long.html>) is the recommended way to receive arguments in MaterialsScript. The following example script can receive four arguments:

```
#!/perl
use strict;
use Getopt::Long;
use MaterialsScript qw(:all);
my %Args;
GetOptions(\%Args, "Target=f", "NumSteps=i", "Name=s", "Flag=s");
print "Target value is: " . $Args{Target} . "\n";
print "NumSteps value is: " . $Args{NumSteps} . "\n";
print "Name value is: '" . $Args{Name} . "'\n";
print "Flag value is: '" . $Args{Flag} . "'\n";
```


The script is able to receive and use these arguments:

1. a numeric argument named "Target",
2. an integer argument named "NumSteps" and
3. a character string argument named "Name".
4. a boolean argument named "Flag" ("Yes" or "No").

Note: In addition, the document in focus is also supplied to the script as an implicit argument, available through the `ActiveDocument` property.

Note: This example uses capitalized argument names. However, by default argument name comparison is performed case-insensitively by the Perl `GetOptions()` function and for Perl hash key lookups.

Setting up a custom menu command to provide arguments to a script

1. Open the [Script Library dialog](#) and select the [User Menu](#) tab.
2. Select the group where a new item should be created and click the *Command* button. A new item will be added to the menu item list named "My Command" by default.
3. Enter a name for the command in the *Title* textbox and a description if required.
4. Click the  button to open the [Choose Document](#) dialog. Select the script which you created above and click the *OK* button.
5. Select the type of document that the script *Requires*. The command on the User menu will only be available when the appropriate type of document is active.
6. Click the *Arguments* button to open the [Define Arguments](#) dialog.
7. Click the *Add argument* button to add one or more arguments.
8. Enter the name and data type for each argument.
9. Enter a default value for each argument.
10. Close the *Define Arguments* dialog.
11. Choose whether to run the command on [Client](#) or [Server](#). The icon displayed on the User menu for the command will indicate whether its script will be run locally or on a server.


Using a custom menu command which requires arguments

1. Choose the command from the User menu. The [Specify Arguments](#) dialog automatically opens.
2. Inspect and modify the default argument values.
3. Click *OK* to run the script or *Cancel* to cancel the command.

Running scripts on a server


Materials Studio allows you to run scripted jobs on remote servers, launching the computation from your Windows desktop PC. When a script calls a module, Materials Studio generates the appropriate input files and transfers these, plus the entire contents of the current folder (and, optionally, any subfolders), to the specified server. When the job is completed, all the files (both input and output) are downloaded to the appropriate Materials Studio project on the client. The scripted job can then resume, if applicable.

To run a script on a Materials Studio server

1. Either import the script from a pre-existing file or use the Script Viewer to [write](#) a new script.
2. [Check the syntax](#) of the script and [debug](#) it using the Script Viewer. Correct any errors that are reported.
3. Once you are confident that your script functions as intended, select *Tools / Scripting / Script Job...* from the menu bar to display the [Script Job Control dialog](#).
4. Choose a server on which to run the scripted job from the *Gateway location* dropdown list. If necessary, specify the *Queue* to which the job will be submitted. Materials Studio will automatically assign a name to the job based on the name of the script document. If you wish to specify an alternative name, uncheck the *Automatic* checkbox and enter the new name in the *Job description* text box.
5. Click the *More...* button to display the [Script Job Control Options dialog](#). Select the documents to be used for live updates and set the behavior of Materials Studio on job completion.
6. Click the *Run on Server* button  on the Scripting toolbar.
7. If you wish, you can examine the [intermediate results](#) to ensure that the calculation is progressing as expected.
8. After the job has finished, view the output files, which will be returned to the project folder in the Project Explorer.

A sample remote scripted job

The sequence of steps that is executed to run a remote scripted Materials Studio job is always the same.

When you click the *Run on Server* button  on the Scripting toolbar or select *Tools / Scripting / Run on Server* from the menu bar, the following happens:

1. Materials Studio communicates with the gateway process to set up a new job.
2. Materials Studio creates the results folder. The results folder name will be generated using the *Job description* specified on the [Script Job Control dialog](#) or, if you have left the *Automatic* checkbox checked, the name will be generated using the name of the document containing the script that is being executed.
3. Materials Studio opens the Job Explorer. This window contains information about the status of the job and its progress.
4. Materials Studio performs a number of checks on the job settings and input. In addition, all job settings are stored in the results directory.
5. Materials Studio temporarily creates a new project in the job scratch area. This project contains all input files and settings for the remote Materials Studio job. The job project files are transferred to the gateway destination you selected on the Script Job Control dialog.
6. When all of the files have been transferred, the gateway starts the job.

7. If you have checked one or more of the *Update structure*, *Update graphs*, or *Update textual results* checkboxes on the [Script Job Control Options dialog](#), Materials Studio will read [intermediate results](#) and update the information on screen and in the relevant file, if applicable. Materials Studio attempts to do this at the intervals specified in the *Update every* field on the Script Job Control Options dialog. However, the actual interval between updates is determined by the time it takes for the server to perform a single iteration step. The update interval can, therefore, be significantly longer than the specified update interval. The exact behavior of Materials Studio on receipt of update information depends on the type of calculation that is being performed.
8. Alternatively, you may view remote files using the Remote view facility of the Job Explorer. The files are located in the `[seedname]_files/Documents` folder and its subfolders.
9. You can continue to work in Materials Studio, exit the program, or even shut down your PC. It will not affect the calculations you are performing (unless, of course, you are using your PC to run the server application as well as the Materials Studio client).
10. Once the job has finished, Materials Studio will transfer the output files back to your PC, where you can view and edit them or use them for further calculations.
11. If you have checked the *Retain server files* checkbox on the Script Job Control Options dialog, Materials Studio will not remove the files from the remote job folder.
12. If you have checked the *Automatically view output* checkbox on the Script Job Control Options dialog, Materials Studio will display the `[seedname].pl.out` file.
13. If you have checked the *Notify on job completion* checkbox and the job log window is not open, Materials Studio will open a job completion window to notify you of the job status.
14. If you stop a scripted job using the *Stop Job* button on the job log window, Materials Studio will offer to download any files produced by the job before it was terminated.

The description given above is somewhat simplified, but provides a reasonable overview of a remote scripted Materials Studio job.

Using job control in scripting

Materials Studio can run scripted jobs as background processes on a server. The following tools are provided to set up and control remote jobs:

- Use the [Script Job Control dialog](#) to select the gateway location and job parameters for future jobs.
- Use the Server Console application to add new servers and to monitor multiple jobs.
- Use the Job Explorer to monitor multiple jobs.

To set job parameters

1. Choose *Tools / Scripting / Script Job...* from the menu bar to display the Script Job Control dialog.
2. Set the *Gateway location* and select the *Queue* to which the job will be submitted.
3. By default, Materials Studio uses the name of the script document as the *Job description*. Additional information is appended to this prefix prior to the job being started. However, you may wish to use an alternative description, to distinguish between separate runs of the same script, for example.
4. If you wish to specify a job name, uncheck the *Automatic* checkbox and enter a new name for the *Job description*.
5. If you want files in subfolders within the current folder transferred to the server with your script, check the *Include subfolders* checkbox. By default, only files within the current folder are transferred to the server; any subfolders and their contents are ignored.
6. Click the *More...* button to display the [Script Job Control Options dialog](#).
7. If required, modify the default parameters that control the [live updates](#) policy and how completed remote jobs are reported.

Launching a job

When you start a remote scripted job, the Job Explorer is displayed automatically. It provides basic information about the job as it progresses. To obtain more detailed information about a server job, double-click on a row in the Job Explorer to open a job log window for the corresponding job.

The job log window contains all of the relevant information about a server job and its current status. It also allows you to stop a job and monitor the transfer of files between the gateway and your PC.

Job completion information

When a remote scripted job finishes, the results files are transferred from the server to the appropriate results folder in the Project Explorer. Materials Studio will report on completion according to the options you have selected.

If the job is small enough that it is completed while the related launch dialog is still open, the completion status will be displayed there, along with any error messages that were generated.

However, if the remote scripted job is more time consuming, you may wish to exit Materials Studio before the job completes. The next time you start Materials Studio and open the project in which the job was launched, the program will automatically re-establish its connection to the remote job. If the job is still running, live updates will resume. If the job has finished, final results will be downloaded, as described below.

Jobs for which job completion notification was requested are automatically listed, showing the completion status (success or failure) of each job.

Results files for all completed remote jobs are transferred from the server and displayed in the Project Explorer. If you checked the *Automatically view output* checkbox on the Script Job Control Options dialog, the results files are displayed. If an error has occurred, Materials Studio will report this and allow you to examine the project log file.

Tip: When Materials Studio is not running, you can open the Server Console from your desktop to check on job status. Access the Server Console by selecting *Accelrys / Materials Studio 8.0 Server Console* from the list of programs on the Windows *Start* menu.

Managing live updates

You can monitor important intermediate results during a remote scripted Materials Studio job.

The [Script Job Control Options dialog](#) allows you to control the generation of intermediate documents and to set the update frequency. Depending on the calculation that is being carried out, you may be able to view one or more chart documents and/or a structure document, allowing you to monitor the progress of the calculation graphically.

If live updates have been requested, Materials Studio will transfer the live update documents, containing intermediate results from the server, to the job results directory. Materials Studio attempts to do this at the interval specified in the *Update every* field on the Script Job Control Options dialog. However, the actual interval between updates is determined by the time it takes to generate new results. The update interval can therefore be significantly longer than the chosen update interval.

If the *Update structure* checkbox is checked on the Script Job Control Options dialog, a structure document, if available, showing the active configuration of the structure on which the calculation is being carried out is returned.

If the *Update graphs* checkbox is checked on the Script Job Control Options dialog, chart documents, if available, showing the evolution of various parameters over the course of the calculation are returned.

If the *Update textual results* checkbox is checked on the Script Job Control Options dialog, Materials Studio will display the results document `[seedname].txt`. Initially, this contains all of the calculation parameters and is updated if any non-fatal errors occur during the calculation.

Notes:

- All the selected live update documents are transferred to the project folder at the interval specified. For clarity, not all available live update documents are opened automatically. You can select and open any live update documents whose contents you wish to monitor using the Project Explorer.
- All live update documents remain in the project folder in the Project Explorer when the job is completed.
- If you exit Materials Studio during a remote scripted job and restart it at a later point, the intermediate update documents will be updated once Materials Studio reconnects to the running job.

To manage live updates

1. Choose *Tools / Scripting / Script Job...* from the menu bar to display the [Script Job Control dialog](#).
2. Click the *More...* button to display the Script Job Control Options dialog.
3. If you would like to receive structure updates (if available), check the *Update structure* checkbox.
4. If you would like to receive chart updates (if available), check the *Update graphs* checkbox.
5. If you would like to receive textual updates, check the *Update textual results* checkbox.
6. Set the minimum time between live updates by adjusting the value in the *Update every* field.

If a remote scripted job fails

Materials Studio checks most of the data and settings required to perform a remote scripted job prior to launch. If it cannot start the job, error messages are generated detailing the reasons.

However, jobs may sometimes fail for reasons which cannot be checked prior to launch. In such cases, more detailed information about the error may only be available in the `[seedname].pl.out` report file produced by the job or in the `MatStudioLog.htm` file located in the server-side project directory, `[seedname]_Files`. Other files stored in the directory on the server, for example, `MatStudioServer.log`, may also contain further clues. To view the server-side files, you can use the Remote View facility of the Job Explorer.

Below is a list of the most common reasons for remote scripted jobs to fail. It may help you to identify and fix any problems you have with your remote scripted jobs.

Tip: Select *View / Project Log* from the menu bar to see if any error or warning messages have been reported.

Common reasons for a remote scripted job to fail to start

1. Gateway and network communication problems
 - No gateways registered on client.

The default installation of Materials Studio on a client does not set up the gateway addresses for remote server programs. You can check whether you have gateways registered on your PC using the Server Console. You can also use the Server Console to add gateways.

- No gateway registered on client that has a suitable server available.
Scripted jobs require the appropriate server to be installed. A server gateway that provides the correct environment must be selected. You can view the servers available through a particular gateway using the Server Console.
- Local gateway information out of date.
The information about which servers are available through a gateway is kept on the gateway itself. When a new gateway is registered on the PC using the Server Console, the list of servers is copied to the client PC. But when a new server is made accessible or removed from a gateway, you will have to refresh the information stored on your PC. You can use the Server Console to refresh the gateway information.
- Gateway not available.
If the computer that is running the gateway is switched off or the gateway program is not running, you will not be able to launch jobs on it. Select a different gateway or contact the system administrator of the gateway you want to run the job on. You can test whether a gateway is running using the Server Console.
- Network failure.
Materials Studio needs to be able to communicate with a gateway. If this is not possible, it will not be able to launch a job. You can test whether a gateway is running using the Server Console. If you can launch the same job on the same gateway from a different PC, then it may be a problem with your PC. In this case, try rebooting your PC or consult your system administrator.
- Disconnected Server Manager program stopped responding.
The communication between Materials Studio and the gateways is performed by the Disconnected Server Manager (DSMgr . exe). This program runs in the background and, normally, no output can be observed. However, it may sometimes stop responding, in which case the following symptoms may be observed:
 - No jobs can be started from your PC, but from a different PC the same job can be started on the same gateway.
 - The network connection to the gateway seems OK and you can access the gateway information from an Internet Explorer window with the URL `http://<gateway>:<port>`, for example, `http://numbercruncher.accelrys.com:18888`.
 - It is not possible to obtain information about any gateway using the Server Console.
 To fix this problem, you will have to exit Materials Studio and end the DSMgr task using the Windows Task Manager. When you restart Materials Studio, the DSMgr will be started automatically.
- User authentication failure for a gateway.
When a gateway is set at the password level of security, a valid user name and password are required to run a job on such a gateway. If the user name and password supplied cannot be authenticated against the set of users registered for the gateway, the server job will fail to start.
- Job launch failure - network timeout.
Slow network communications can result in job failure if the gateway timeout setting is not high enough. Try increasing the gateway timeout value if network speed could be causing the job launch failure. Refer to the Server Console help for details on setting the timeout.

2. Client-side problems

- Disk full on client PC.

Materials Studio creates some files that contain input information and job details. If these files cannot be created because the disk is full, the job cannot be started.

Note: These files are usually created in a temporary file folder, for example, C:\TEMP. You must free up disk space so that these files can be created.

- Job filename too long.

If a results directory is deeply nested within a number of folders in a project, the default job filename automatically generated by Materials Studio may become longer than the Windows limit of 259 characters. If this happens, you will need to manually assign a shorter filename to the job on the *Job Control* tab before you can run it.

3. Server-side problems

- Disk full on server.

Materials Studio servers need to create files to store the job information and job output data. If these files cannot be created because the disk is full, the job cannot be started. Contact your system administrator to ensure that enough disk space is available on the gateway computer.

- License failure.

The use of the server programs is licensed and might only allow a limited number of jobs to be executed concurrently. In this case, you may want to try to run your job on a different gateway or wait until licenses are available again and restart your job.

- Server process - resources failure.

If the server has too few resources to execute your job, the job may fail. This can be caused by too many jobs being executed on the server. In this case, you may want to try to run your job on a different gateway or wait until the resources are free again. If the problem occurs frequently, you may want to notify the system administrator of the gateway computer of this problem.

Common reasons for a remote scripted job to fail to finish successfully

1. Server-side problems

- Out of disk space.

If the server machine runs out of disk space while a scripted job is running, various errors may occur. You may have to ask the system administrator of the gateway to free up some disk space before trying to run another job.

- Out of memory.

If the server machine runs out of virtual memory or swap space when trying to start up a process or while running, various errors may occur. These errors depend on the operating system of the server machine.

- Communication failure.

Slow network communications can result in job failure if the gateway timeout setting is not high enough. Try increasing the gateway timeout value if network speed could be causing the job to fail. Refer to the Server Console help for details on setting the timeout.

Common reasons for a remote scripted job to fail to download results

1. Scripting process ended without writing the results.

When the server program encounters problems, it may exit without producing the required output files. You may want to look very carefully at the `[seedname]_Files/Documents/[seedname].txt` file produced by the job for indications of the problem, or, if that file does not exist, at the `[seedname]_Files/MatStudioLog.htm` or `MatStudioServer.log` files, which may contain further clues. To view any remote files, use the Server Console. If neither file exists, you may want to ask the system administrator of the gateway to check whether the disk on the server is full.

2. Results folder removed.

When Materials Studio launches a remote scripted job, it creates a subfolder that will contain the results of the scripted calculation once the job is finished. This folder must not be renamed, moved, or deleted, since the results cannot then be downloaded. If the files cannot be downloaded, they will not be removed from the server and you can retrieve them individually using the Server Console (or by using the *Save As...* facility of your browser if you are accessing gateway information using the URL).

3. Disk full on client PC.

There needs to be enough free disk space on your PC to store the results files from the remote scripted job. If there is not enough free disk space, the download of the files will fail, although the files will remain stored on the server. After you have freed up some disk space, you can retrieve the files individually using the Server Console.

4. Communication failure between client and server during download.

You can download the output files manually. You need to copy and paste or FTP all the files in the `[seedname]_Files/Documents` folder (located within the server-side job directory) into the client-side job results folder.

Tip: It is important to identify the reason for the failure of a remote scripted job before taking any action. In most cases, the error message is reported in the job report file `[seedname].pl.out` or in the `MatStudioLog.htm` file located in the server-side project directory `[seedname]_Files`.

Running MaterialsScript in standalone mode

The most convenient way of running scripts is via the Materials Studio interface, which performs all the preparatory tasks required to run a script job. However, in some circumstances, it may be necessary to run scripts in standalone mode with a set of input files prepared elsewhere. For example, you may wish to run a calculation on a server that is not running a gateway, i.e., a computer that does not communicate with your Materials Studio client (perhaps because a firewall prevents automatic file transfer and job launch).

For more general information on using standalone mode, see the [Running jobs in standalone mode](#) help topic.

For more information on MaterialsScript running in standalone mode in Materials Studio 7.0 and earlier, see the [Running MaterialsScript in legacy standalone mode](#) help topic.

For more information on running multiple MaterialsScript jobs in standalone mode, see the [Running multiple standalone MaterialsScript scripts in the same directory](#) help topic.

Generate the input files

A scripting job requires a least one file that contains the MaterialsScript for the job. Although the Materials Visualizer provides an integrated script editing environment through the Script Viewer, you can use any text editor to create your Materials Studio scripts, such as WordPad on Windows or vi on Linux. Typically a script job will require additional information such as structure details, for example as an .xsd file. These extra files must also be transferred to the server, where they may be placed in the same folder as the script.

Tip: When the script is run, Materials Studio automatically recognizes these extra files as documents and makes them available in the Documents collection.

Transfer the input files to the server

If you generated the input files manually using a text editor on the server machine, then no file transfer is required. However, if you generated the files on your PC using Materials Studio, you must transfer them to the server before you can start the calculation.

If you are unable to access the hard drive on the server, you should use the File Transfer tool to transfer files from the client to the server.

Execute the job

To assist you in running a scripting job in standalone mode, a batch/shell file called RunMatScript is supplied. It can be found in the folder etc/Scripting/bin in the main Materials Studio directory. RunMatScript scripts are used to start scripting jobs in standalone mode. RunMatScript.sh is provided for Linux servers, while RunMatScript.bat is provided for Windows servers.

Usage:

```
RunMatScript.sh [-h] [-np number of cores] [-q queue name] [-project]
scriptname [ -- script arguments ... ] (Linux)
```

or

```
RunMatScript [-h] [-np number of cores] [-q queue name] [-project] scriptname
[ -- script arguments ... ] (Windows)
```

Argument	Description
-h	Displays the help text.
-np	Specifies the number of cores on which to run the job. When this option is not specified a single core is used.
<i>number of cores</i>	The number of cores to use.
-q	Submits the job to the specified queue.
<i>queue name</i>	The name of the queue on which to run the job.
-project	Request that the script be run in legacy standalone mode
<i>scriptname</i>	The name of the Perl script to be run.
<i>script arguments</i>	Any arguments to be processed by materials script itself.

To use script arguments with a script, the script must be altered so that it can [process arguments](#). For example, a script called DMol3SPE might be altered so that it can take a character string argument holding the name of the structure file to be processed:

```
#!/perl
use strict;
use Getopt::Long;
use MaterialsScript qw(:all);
my %Args;
GetOptions(\%Args, "Structure=s");
my $doc = $Documents{"$Args{Structure}.xsd"};
Modules->DMol3->Energy->Run($doc) ;
```

In this example the script argument is "Structure" and if the input document is methane.xsd then the script should be executed in the following manner:

```
RunMatScript DMol3SPE -- --Structure methane
```

Note: The first "--" is important as it indicates that all subsequent arguments are to be processed by Materials script. All subsequent "--" indicate an argument name, for example "Structure" here, followed by the argument value, for example "methane" here.

Download the output files from the server

When the scripting job is complete, you must transfer the output files back to your PC for analysis in Materials Studio. See Running jobs in standalone mode for further information.

To transfer the output files back to your PC

1. Transfer the output files to the client PC using either copy and paste or the File Transfer tool.

Tip: If you have transferred files into a Materials Studio project folder, but you cannot see them in the Project Explorer, try using the *Refresh* button  to update the Project Explorer.

Open the output files in Materials Studio

Provided that you have transferred the correct files from the server to your PC and stored them in a folder in a Materials Studio project, you should be able to make use of any appropriate analysis options. Specifically, you can open the 3D Atomistic document containing the starting structure and update it, create a trajectory document and animate it, or visualize volumetric properties.

Running multiple standalone MaterialsScript scripts in the same directory

The simplest way to run multiple standalone MaterialsScript script jobs is to run each from a separate directory, so that they have independent input files and output files.

Limited support is also provided for running multiple scripts in the same directory. You can run standalone scripts one after the other, or concurrently, in the same directory.

Summary of limitations

- Only use of the Forcite module is supported.
- Scripts run in this mode should have unique names, in order for all outputs to be retained.
- When scripting concurrently, undefined results will occur if the same document file (for example, .xsd) is modified by one script and used by another.

Note: Most MaterialsScript functions on a document modify that document.

- These scripting modes are not supported for queued jobs (that is, use of "-q" in the RunMatScript command line).
- These scripting modes are supported for parallel execution (that is, use of "-np" in the RunMatScript command line), except for concurrent execution on clusters, which is not supported.

Running scripts in standalone mode in the current directory

When running standalone scripts, the script and its input documents are in the same directory tree, within the current directory. The input files are made available to the MaterialsScript script automatically as documents in the Documents collection.

While the script runs, more output documents may be created and added to the Documents collection. In addition, some of the input documents may be changed by the script. The script may even delete some of the documents. By default, these changes are then saved to the same directory location when the script finishes, overwriting the existing files.

Running multiple scripts sequentially or concurrently in standalone mode

You can run multiple scripts in the same directory, with or without waiting for earlier scripts to finish.

Sequential running

An example of the batch or shell commands for sequential scripting is as follows:

Windows

```
RunMatScript MyScript1
RunMatScript MyScript2
RunMatScript MyScript3
```

Concurrent running

For concurrent script execution, it is recommended that the scripts operate independently. In particular, the scripts should not attempt to modify the same document files, but instead operate on different document files, or files which are in different sub-folders. In addition, it is recommended that the scripts do not rely upon the existence of files or folders added, removed or renamed by the other scripts.

Examples of the batch or shell commands for concurrent scripting are as follows:

Linux

```
RunMatScript.sh MyScript1&
RunMatScript.sh MyScript2&
RunMatScript.sh MyScript3&
```

Windows

```
@start cmd /c "RunMatScript MyScript1"
@start cmd /c "RunMatScript MyScript2"
```

```
@start cmd /c "RunMatScript MyScript3"
```

The above sequence of commands automates the launch and running of `MyScript1.pl`, `MyScript2.pl`, and `MyScript3.pl` in the current directory, at the same time. Another way to achieve this is simply to invoke the normal `RunMatScript` command from separate shell windows in the same current directory, without waiting for completion of the others.

IMPORTANT! Concurrently running a script file of the same name in the same directory is not supported.

Note: Most `MaterialsScript` functions on a document modify that document. For this reason, concurrent scripting activities on the same document file may often give unpredictable results, and are not recommended.

Note: The additional `RunMatScript` command options for parallel execution and script arguments are also supported, except that queued execution is not supported. Parallel execution of concurrent scripts on multiple-machine clusters is also not supported.

Running MaterialsScript in legacy standalone mode

If you have `MaterialsScript` from `Materials Studio 7.0` and earlier that ran in standalone mode, you have two options:

- You can run your scripts in the legacy standalone mode
- You can upgrade your scripts to run in the new standalone mode

The main difference in legacy standalone mode is how the input and output files for the script are organized. Running an old script unchanged in the new standalone mode may give different results or encounter an error.

Running in legacy standalone mode

To run the legacy script unchanged, add the `-project` argument on the command line.

In this example the script to be executed in legacy standalone mode is named `MyScript.pl`:

```
RunMatScript -project MyScript
```

Upgrading legacy scripts

Legacy standalone scripts typically import input files from the folder in which the script is being run, for example:

```
my $doc = Documents->Import("./Input.xsd");
```

If this script is run in the new standalone mode, it will generate an error. The intended action is ambiguous, because `Input.xsd` in the current directory is now already recognized as a project document:

```
Import failed for target file
```

```
C:\<full path>\Input.xsd
```

```
The project already contains this document (function/property "Import") at -e line 51
```

For standalone scripting in `Materials Studio 7.0` and earlier, each `Import` took a copy as a new independent document in the `Documents` collection so that the copied file had a different location from the original. If that copy was modified later in the script, the original would remain unchanged.

To upgrade the script, consider whether taking a copy is necessary for your case. If not, simply change the import line to read:

```
my $doc = $Documents{"Input.xsd"};
```

If you do need separate copies of the original, then either locate your input files elsewhere on the server and reference that location in the Import, or change the import line to read:

```
my $doc = Documents->New("Input.xsd");  
$doc->CopyFrom($Documents{"Input.xsd"});
```

This will create a new document Input (2).xsd which is an independent copy of the original.

Dialogs in scripting

The Script Viewer allows you to write, edit, and debug scripts in the Materials Visualizer. Once written, scripted jobs can be set up and run on remote servers via the Materials Studio interface.

The following topics and their subtopics describe the dialogs used to interact with scripts and to run remote scripted jobs in the Materials Visualizer.

Find and Replace dialog

The Find and Replace dialog allows you to search for and substitute occurrences of a text string in the active script document. Matches are highlighted in the Script Viewer.

Find what: Specify the text string to be searched for.

Find Next: Finds the next instance of the specified text string in the active script document. When the search reaches the start or end of the script, it will restart from the bottom or top of the document, as appropriate.

Replace with: Specify the text string that will replace instances of the specified search string in the active script document.

Replace: Replaces the selected text in the currently active script document with the text string specified in the *Replace with* field.

Replace All: Replaces all instances of the specified search string in the currently active script document with the text string specified in the *Replace with* field.

Match case: When checked, indicates that the specified text string will be matched only if the capitalization of every letter in the string in the currently active script document is the same as that specified in the *Find what* field.

Find whole words only: When checked, indicates that the specified text string will be matched only if the string in the currently active script document represents a whole word bounded by white space or punctuation.

Direction: Specify whether to search the currently active script document upward from the cursor position toward the start of the script or downward from the cursor to the end of the script by selecting the appropriate option.

Help: Displays the Help topic in a browser.

Access methods

Menu	<i>Edit Find and Replace</i>
Shortcut menu	<i>Find and Replace</i>

Script Viewer Options dialog

The Script Viewer Options dialog allows you to control the way in which script documents are rendered in the Script Viewer and how they are printed.

The Script Viewer Options dialog contains the following tabs:

- **General:** Allows you to set a number of Script Viewer rendering options, including the font, indentation and syntax coloring schemes, and display of white space and line numbers.
- **Printing:** Allows you to specify the way a script document will appear when printed.
- **Colors:** Allows you to specify the colors to be used for syntax coloring, error highlighting, and other elements of the Script Viewer rendering scheme.

OK: Updates the settings with any changes and closes the dialog.

Cancel: Closes the dialog without updating any settings.

Apply: Updates the settings with any changes and does not close the dialog.

Help: Displays the Help topic for the current tab.

Access methods

Menu	<i>View / Viewer Options...</i>
Shortcut menu	<i>Viewer Options...</i>

General tab

The *General* tab allows you to set a number of Script Viewer rendering options, including the font, indentation and syntax coloring schemes, and display of white space and line numbers.

Viewer Font...: Provides access to the Font dialog, which allows you to select the typeface, style, and size of the font used to display script documents in the Script Viewer.

Indent new lines: When checked, indicates that the Script Viewer will add the appropriate amount of white space at the beginning of a new line inserted into a script document so that it is indented by the same amount as the previous line.

Show whitespace: When checked, indicates that the Script Viewer will show spaces in script documents as dots and tabs and carriage returns as guillemets.

Show line numbers: When checked, indicates that the Script Viewer will display line numbers in the margin to the left of the script.

Insert closing }]]: When checked, indicates that the Script Viewer will automatically insert a closing brace, parenthesis or bracket when you type an opening one, and place the cursor between them.

Use syntax coloring: When checked, indicates that the Script Viewer will change the color of text in script documents based on the function of that text (for example comments, keywords, operators, and so on).

Highlight matching Find text as it is typed: When checked, indicates that the Script Viewer will automatically highlight text in script documents that matches the text string in the *Find what* field on the [Find and Replace dialog](#).

Underline errors: When checked, indicates that the Script Viewer will highlight syntax errors in script documents by underlining them with a wavy line.

Tab size: Specify the width in terms of characters of a tab in script documents displayed in the Script Viewer.

Tip: Some script documents assume tabs of a set width. Use the *Tab size* slider control to adjust the tab width so that indentations appear correctly. Check the *Show whitespace* checkbox to show the tabs in a script document.

After changing the settings on the Script Viewer Options dialog, click the *OK* button to apply the changes and close the dialog or click the *Apply* button to make the changes and leave the dialog open so that

further alterations to the settings can be made, if required. Click the *Cancel* button to close the Script Viewer Options dialog and discard any changes to the Script Viewer display settings.

Access methods

Menu	<i>View / Viewer Options / General</i>
Shortcut menu	<i>Viewer Options / General</i>

Printing tab

The *Printing* tab allows you to specify the way a script document will appear when printed.

Printer Font...: Provides access to the Font dialog, which allows you to select the typeface, style, and size of the font to be used when printing script documents.

Use syntax coloring: Select from the dropdown list whether to color text based on its function (for example, comments, keywords, operators, etc.) when printing script documents. Available options are:

- [Always](#)
- [Never](#)
- [If shown in the Viewer](#) (default)

Print line numbers: Select from the dropdown list whether to include line numbers when printing script documents. Available options are:

- [Always](#)
- [Never](#)
- [If shown in the Viewer](#) (default)

Print whitespace: Select from the dropdown list whether to show spaces as dots and tabs and carriage returns as guillemets when printing script documents. Available options are:

- [Always](#)
- [Never](#)
- [If shown in the Viewer](#) (default)

After changing the settings on the Script Viewer Options dialog, click the *OK* button to apply the changes and close the dialog or click the *Apply* button to make the changes and leave the dialog open so that further alterations to the settings can be made, if required. Click the *Cancel* button to close the Script Viewer Options dialog and discard any changes to the Script Viewer display settings.

Access methods

Menu	<i>View / Viewer Options / Printing</i>
Shortcut menu	<i>Viewer Options / Printing</i>

Colors tab

The *Colors* tab allows you to specify the colors to be used for syntax coloring, error highlighting, and other elements of the Script Viewer rendering scheme.

Show this syntax category: Select from the dropdown list the type of script document feature for which you wish to change the text display color. Available options are:

- Keywords
- Variable names
- Function names
- Numbers
- Strings
- Comments
- Punctuation
- Operators
- Matching {}[]()
- Mismatching {}[]()
- Whitespace
- Plain text
- Everything else

in this color: Select a color to be used for the selected syntax category in the Script Viewer. Clicking on the color control provides access to the Microsoft Common Color Control for color selection. The spin controls allow the brightness to be adjusted.

Bold: When checked, indicates that text belonging to the selected syntax category will be displayed in bold type in the Script Viewer.

Show line numbers in this color: Select a color to be used for line numbers in the Script Viewer. Clicking on the color control provides access to the Microsoft Common Color Control for color selection. The spin controls allow the brightness to be adjusted.

on this background: Select a background color to be used for the window margin in the Script Viewer. Clicking on the color control provides access to the Microsoft Common Color Control for color selection. The spin controls allow the brightness to be adjusted.

Highlight text in this color: Select a background color to be used to highlight text that matches the text string in the *Find what* field on the [Find and Replace dialog](#). Clicking on the color control provides access to the Microsoft Common Color Control for color selection. The spin controls allow the brightness to be adjusted.

Underline errors in this color: Select a color to be used to underline syntax errors in the Script Viewer. Clicking on the color control provides access to the Microsoft Common Color Control for color selection. The spin controls allow the brightness to be adjusted.

After changing the settings on the Script Viewer Options dialog, click the *OK* button to apply the changes and close the dialog or click the *Apply* button to make the changes and leave the dialog open so that further alterations to the settings can be made, if required. Click the *Cancel* button to close the Script Viewer Options dialog and discard any changes to the Script Viewer display settings.

Access methods

Menu	<i>View / Viewer Options / Colors</i>
Shortcut menu	<i>Viewer Options / Colors</i>

Script Job Control dialog

Scripted Materials Studio calculations can be run in the background on a server via the gateway. The Script Job Control dialog allows you to select a server for a remote scripted calculation and to control some aspects of how the calculation will be performed.

Note: The options specified on the *Job Control* tab only apply to new jobs. They do not affect jobs that are already running.

Gateway location: Select a server for the scripted calculation from the list of available server machines. You can add servers to the list using the Server Console.

Queue: Specify the queue to which the job will be submitted. Select the desired queue from the dropdown list, which displays the available queues on the chosen gateway. See *Working with queues* for additional details.

Job description: Specify the name to be used to identify the job.

A default job description is automatically assigned. An alternative description can be chosen by unchecking the *Automatic* checkbox and entering the new name in the *Job description* text box.

Automatic: When checked, indicates that a job description will be selected automatically. Default = [checked](#).


Run in parallel on: Indicates that the job will be run on the selected gateway using the specified number of computer cores. The text to the right of the *Run in parallel on* control indicates the maximum number of cores available on the selected gateway.

Include subfolders: When checked, indicates that files in subfolders of the current folder in the Project Explorer will also be transferred to the server with the script. Scripts run from User menu commands offer individual control and ignore this option. Default = [unchecked](#).

More...: Provides access to the [Script Job Control Options dialog](#), which allows you to set additional options associated with monitoring and controlling the results of a remote scripted job.

Help: Displays the Help topic in a browser.

Access methods

Menu	<i>Tools Scripting Script Job...</i>
	<i>User Script Job...</i>
Toolbar	 <i>Script Job...</i>

Script Job Control Options dialog

The Script Job Control Options dialog allows you to set the options associated with monitoring and controlling the results of a remote scripted Materials Studio calculation.

Update structure: When checked, indicates that intermediate results will be used to update the displayed structure as the job progresses. Default = [unchecked](#).

Update graphs: When checked, indicates that intermediate results will be used to update the displayed graphs as the job progresses. Default = [checked](#).

Update textual results: When checked, indicates that intermediate results will be used to update textual results files as the job progresses. Default = [checked](#).

Tip: Intermediate updates are useful shortly after initiating a job to assess if it is progressing as expected.

Update every: Specify the time interval, in seconds, between requests for intermediate updates.

Note: The rate at which new results appear is limited by the time it takes for the server to calculate new results. This may be significantly longer than the chosen update interval.

Retain server files: When checked, indicates that the folder on the server containing the job files will be retained after the job is complete. Default = [unchecked](#).

If this checkbox is left unchecked, the job files on the server will be deleted. Regardless of whether it is checked or unchecked, copies of the results files will *always* be retrieved from the server, placed in the associated project on the local machine, and displayed in the Project Explorer.


Automatically view output: When checked, indicates that the output files from the job will be opened automatically when the calculation is complete. The files that are opened depend on the calculation that is being carried out. Default = [checked](#).

Notify on job completion: When checked, indicates that a dialog will be displayed when the job is complete. Default = [checked](#).

Tip: If you run several short jobs in one session, you may find it useful to stop the automatic display of job completion notices and results files.

Help: Displays the Help topic in a browser.

Access methods

Menu	<i>Tools / Scripting / Script Job... / More...</i>
	<i>User / Script Job... / More...</i>
Toolbar	 / <i>Script Job... / More...</i>

Script Library dialog

The Script Library dialog allows you to build up a library of scripts and other documents that are shared between projects. You can then create custom menu items that execute those scripts on the active document.

The Script Library dialog contains the following tabs:

- [Library](#): Allows you to define locations that contain scripts and other documents that are shared between projects. Allows you to import scripts and other documents into the library, from the project.
- [User Menu](#): Allows you to define custom menu items that execute scripts from the library.

Help: Displays the Help topic for the current tab.

Access methods

Menu	<i>Tools / Scripting / Library...</i>
	<i>User / Library...</i>

Library tab

The *Library* tab allows you to define locations where documents can be shared between projects and between users. Each location can be defined by more than one path, for example a Windows and a UNIX path, so that the documents can be accessed on a server as well as the client.

You can manage the contents of the library folders by importing documents from the project, and by deleting and renaming documents.

[File list]: Lists all the defined library folders and their subfolders and files. If a library folder has no valid path, the library will be displayed in gray.

File

Add...: Opens the [Add From Project](#) dialog so you can copy a document from the project to the selected folder.

Import: Imports a copy of the specified library document into the project.

Tip: If you can't add a file because these buttons are disabled, it could be because there is no location to add it to or the location has no path. Click *Location / Add* to add a location, then enter a path in the *Location paths* list.

Folder

Add: Adds a new subfolder to a location. Subfolders help you manage large numbers of documents.

Location

Add: Adds a new location to the list. Initially it will have no valid path, so no documents can be imported.

Location paths

Name: Specifies the name of the selected location.

[Path list]: Lists the paths to the location, in priority order from first to last. All paths should point to the same place. For example, the same folder shared between a Windows client and a UNIX server will have a different path in each operating system. The active path is the first in the list that is found to exist.

<click to add path>: Adds a new path to the list.



Add path: Adds a new path later in the list.



Delete path: Deletes the selected path from the list.

Move path to earliest: Moves the selected path to the earliest in the list.

Move path earlier: Moves the selected path earlier in the list.

Move path later: Moves the selected path later in the list.

Move path to latest: Moves the selected path to the latest in the list.

Access methods

Menu	<i>Tools / Scripting / Library... / Library</i>
	<i>User / Library... / Library</i>

Add From Project dialog

The Add From Project dialog allows you to copy a document from the project to the library.

OK: Adds the selected document to the library and closes the dialog.

Cancel: Closes the dialog without adding the document.

Help: Displays the Help topic in a browser.

Access methods

Menu	Tools Scripting Library... Library Add...
	User Library... Library Add...

User Menu tab

The *User Menu* tab allows you to add custom menu items that execute scripts on the active document. These menu items will appear on the User menu.

[Menu item list]: Lists the currently defined commands as they will appear in the User menu.

Insert

Command: Adds a menu item.

Group: Adds a group of menu items.

Archive

Import...: Opens the [Import Menu](#) dialog to merge menu items from a file into the current menu structure.

Export...: Opens the [Export Menu](#) dialog to write the selected command or group of commands to a file.

Tips:

- When sharing custom menu items with other users you must ensure that all the scripts and associated files are stored in a uniquely named location on a shared network path. The name and path of this location will be copied to the other users along with the custom menu items. If you wish to change the path of the files, export the custom menu items again and send the file to your users.
- Do **not** store files that you need to share in your *My Favorites* folder; this will change the path to the *My Favorites* folder on your users' machines, breaking their custom menu items. You can create a new location on the [Library](#) tab.
- To prevent others from changing scripts that you have shared with them, you can restrict the permissions on the script files.

Command

Title: Specifies the name of the menu item.

Description: Describes the command. This description appears in the status bar when the mouse is on the menu item.

Script: Specifies the library script document to run.



: Opens the [Choose Document](#) dialog so you can choose a library document.

View: Opens a read-only view on the specified library document. This does not import the document into the project.

Run on: Specifies whether the action is to run on the client, or be submitted to a server.

Requires: Specifies the document type required by the script. The command will only be available on the *User* menu when a document of the specified type is active. Available options are:

- Any document
- 3D Atomistic document
- 3D Atomistic Trajectory document
- 3D Atomistic Collection document
- Chart document
- Forcefield document
- Study Table document
- Text document

Arguments: Opens the [Define Arguments](#) dialog to define default input argument values for the script.

Uses: Specifies which documents and folders will be available to the script when running on the server. This has no effect on scripts that run on the client. Available options are:

- **Active document only** - only the active document in the project will be transmitted to the server along with the script.
- **Current folder only** - only the active document and other documents in the same folder are transmitted to the server. Subfolders are not transmitted.
- **Current folder and subfolders** - the active document and other documents in the same folder and subfolders are transmitted to the server.

Note: Library modifications can result in a previously valid script choice no longer existing in the library. In this case the script name will be highlighted in red and the command will be disabled on the User menu. To view the full library location URL, hover the cursor over the script name so that a tooltip appears.

Access methods

Menu	<i>Tools Scripting Library... User Menu</i>
	<i>User Library... User Menu</i>

Choose Document dialog

The Choose Document dialog allows you to pick the script to be executed, from the documents in the library.

OK: Confirms the selected document and closes the dialog.

Cancel: Closes the dialog without setting the document.

Help: Displays the Help topic in a browser.

Access methods

Menu	<i>Tools Scripting Library... User Menu Browse...</i>
	<i>User Library... User Menu Browse...</i>

Import Menu dialog

You can use the Import Menu dialog to merge commands from a file into your User menu.

File name: Specifies the name of the file you want to import. If you select a file in the file list area, this filename is automatically entered for you. You can also type in the name of a file. Select the file formats that will be displayed in the file list area. Available options are:

- User Menu Files (*.xml)
- All Files (*.*)

Open: Closes the dialog and imports the selected file

Cancel: Closes the dialog without importing the file.

Access methods

Menu	<i>Tools Scripting Library... User Menu Import...</i>
	<i>User Library... User Menu Import...</i>

Export Menu dialog

You can use the *Export* dialog to save the current User menu to a file. This can be used to share commands with others.

File name: Specifies the new name for the file you are saving.

Save as type: Contains all the file formats available for the file.

Save: Closes the dialog and saves the menu to the specified location.

Cancel: Closes the dialog without saving the file.

Access methods

Menu	<i>Tools Scripting Library... User Menu Export...</i>
	<i>User Library... User Menu Export...</i>

Define Arguments dialog

You can use the Define Arguments dialog to define arguments for your script. When the script is run from the User menu, a dialog will request values for the arguments, then the script will run.

Script: States the script that these arguments will apply to.

[Argument list]: Lists the names of the arguments currently specified for this command.

<click to add argument>: Adds a new argument to the list.



Add argument: Adds a new argument later in the list.



Delete argument: Deletes the selected argument from the list.

Move argument to start: Moves the selected argument to the start of the list.

Move argument up: Moves the selected argument up the list.

Move argument down: Moves the selected argument down the list.

Move argument to end: Moves the selected argument to the end of the list.

[Argument Name]

This section allows you to change the settings for the selected argument.

Name: Specifies the name of the argument. This will be the name used in the script.

Data type: Specifies the data type of the argument. This affects how the argument value is requested when the script is run.

Default value: Specifies the default value for the argument.

Help: Displays the Help topic in a browser.

Access methods

Menu	<i>Tools Scripting Library... User Menu Arguments</i>
	<i>User Library... User Menu Arguments</i>

Specify Arguments dialog

This dialog appears when running a script that requires arguments. It allows you to change the argument values from the defaults specified by the script author. The title used for this dialog is the name of the command chosen from the User menu.

The fields on this dialog depend on the script being run and are defined in the [Define Arguments dialog](#).

If you need clarification on the meanings of the arguments, you should either study the script or contact the script author.

Access methods

<i>User [User command]</i>
