

Toad Actions

A User's Guide

Mark.Lerch@Quest.com

September 2008

Introduction.....	1
Overview.....	2
Terminology.....	2
Data Files and Folders	3
Security	3
Creating Actions	3
Creating an Action in the AppDesigner window	4
Creating an Action via the Snapshot window.....	4
Creating an Action through using various Toad windows.....	4
Creating Apps	5
Opening Apps and Actions	5
Executing Apps and Actions.....	6
Logs and Messaging	7
Action Logs.....	7
Action Messages	8
Action Recall	8
Creating Action Toolbars.....	8
Using Window Snapshots	9
The Scheduling Subsystem.....	11
Action Reference	11
Actions Requiring Connections	12
The Select Variable Window	12
The Action Reference List.....	13
Import/Export Actions – Three to Get Data In & Out	13
DB Misc Actions – Database Actions Requiring a Connection	13
Utility Actions – Mirroring Toad's Utilities Menu	14
File Management Actions – For Disk Files & Folders	15
Control Actions – the Programmer's Action Group	16
Control Structures	16
Iterators	18

Introduction

If you are a Toad user you have already been using Actions, possibly without even realizing it. Actions are a technology which was introduced in version 9.1 as a way to encapsulate atomic chunks of functionality, giving formal definition to them so they can be reused, scheduled, streamed to disk and used in conjunction with other Actions to build entire mini Action applications, or “Apps.”

If you have ever used the Save Grid functionality you have used an Export Dataset Action. If you have ever performed a Database Health Check you have used a DB Health Check Action. And these, along with over a dozen other Toad windows, are available for reuse once you have used them. Simply look in the “Action Recall” App (File > Open Action) to see what’s available. But we’re getting ahead of ourselves, because reusing work you have done is just the beginning of what’s possible.

Overview

Before going into the details of how to work with Actions it is worth taking a moment to define our terms and talk a bit about implementation details.

Terminology

- **Action** – an Action is a specific instance of an Action Type. It is a named, atomic chunk of Toad functionality which was given formal definition for the purpose of re-execution, scheduling and re-using in Toad windows. Toad creates internal Actions to do some of its work. You can also create Actions to automate and simplify your day to day work.
- **App** – an App is a group of Actions, typically designed to work together as a mini-application, or “Toad App.” Every Action must be in an App, though an App can be used as simply a collection of unrelated Actions.
- **Action Type** – a template for an Action. For example there is an “Email Action” type and an “Export Dataset” type. If you create an Email Action you have created an instance of that type. Action Types can be used in certain places such as the Open Action dialog to help filter the list of Actions to just those Types you are interested in. The complete list of types can be seen on the Action Palette of the AppDesigner.
- **Action Recall** –The name of the internally managed App containing Actions which Toad automatically creates when you execute certain windows – “actionized” windows.
- **Actionized Window** – a Toad window which uses an Action to do its work. The “Save Grid” window is an example. If a window has been “Actionized,” it can have an Action created from it by simply clicking a button. It can also have an Action loaded into it.
- **Snapshot** – A snapshot is simply a term used for an Action which was created for the sole purpose of saving a window’s GUI state for reuse later. This was previously known as saving and loading ‘Settings.’ In the past Toad used a variety of ways to save and load a window’s state – INI files and other data files; now it uses Actions in all cases.

Defunct terms include “Vault” (now Action Recall), “Action Set” (now App), and “Action Palette” (now AppDesigner).

Data Files and Folders

There are several disk files and folders which store Action data. These all exist in your Toad User Files folder, in the Application Data area of the file system.

- **ToadActions.dat** – This is the principal data file which contains all the Apps and Actions in the system. If you are a heavy user of Toad’s Action technology it would be prudent to make an occasional backup of this file
- **ToadActions.log** – A log entry designating success or failure, as well as execution notes, is made for every execution of an Action. Because the log is much larger than the datafile it is kept in this separate file
- **ToadActions.err** – When Toad starts it reads in ToadActions.dat. In the remote chance that an error occurs, the data file is backed up to ToadActions.err and a new one is created. It then attempts to read in every App individually from the AppBackups folder. It is possible that the R&D team can diagnose the issue given an error file so we have kept this file for possible support cases
- **\AppBackups** – This folder contains a backup file in plain text format for each App in the system. Whenever the AppDesigner or Toad closes it iterates through each App in the system and writes it out to a disk file in this folder with the name “AppN,” where N is a sequence number. If Toad encounters a problem when it streams in ToadActions.dat it will attempt to manually load each App individually from the AppBackups folder

Security

Some Actions necessarily store the connection data as part of their properties. Whether or not the password is stored depends upon the main Toad setting for saving passwords.

- **If “Save Passwords” is turned on** Actions will store the password to their connection in the datafile using the same encryption Toad uses. Because the encryption key is tied to the machine, if the Action or datafile is put on another computer the password will not work and a prompt for password will occur when the Action is executed.
- **If “Save Passwords” is turned off**, passwords are not stored for Actions. Existing Actions in memory will have their passwords cleared if “Save Passwords” had previously been on.

Creating Actions

Actions can be created in several ways:

- Via the AppDesigner window
- Via the Snapshot window
- Indirectly through the use of various Toad windows

Creating an Action in the AppDesigner window

The AppDesigner, available off the Utilities menu, is the normal way one creates Actions. When you first open the Designer you will notice on the left side that Toad creates a default App to begin with, named “App1.” All Actions reside in Apps because they can interact with one another. Apps provide a way to create “Toad Action Applications,” which are collections of Actions designed to work together.

The tabbed area on the top of the right side is the *Action Palette*. It always holds the complete collection of *Action Types* which can be created. The first three tabs – Import/Export, DB Misc and Utilities – contain Action Types which are functional even as stand-alone Actions. The last two tabs – File Management and Control – contain Action Types which most will typically find useful only in conjunction with other Actions within an App.

To create an Action simply click the one you want in the Action Palette then click in the Action list area below the Palette. You can also double-click the Action. The Action will be created then auto-named for speed and convenience. You can now rename it by pressing F2 or right-clicking on it and choosing “Rename.” To work with it, double-click it or choose right-click “Properties.” It is through the Properties dialogs of Actions that all pertinent information is entered. If you click “Run” on the Properties window the Action will execute immediately. Otherwise click “Apply” to post your changes.

Creating an Action via the Snapshot window

Actions can be created based on the contents of Actionizable windows. See the bulleted list further below for the current group of these windows. Click the camera icon on the status bar and specify the Action name and App. See [“Using Window Snapshots”](#) for more details.

Creating an Action through using various Toad windows

Whenever you use certain Toad windows it is Actions which are doing the work behind the scenes. Since that is the case, we decided we might as well preserve those Actions in case you wish to perform the same operations in the future. As with all Actions, these auto-created Actions can be re-executed, opened and modified, scheduled or used as GUI templates to load into their companion windows. The current list of main Toad windows which create user-serviceable Actions include:

- Import Table Data
- Export DDL
- Save Grid
- Object Search
- DB Health Check
- HTML Schema Documentation
- Compare Schemas

In some cases Actions are not used for a window, even though they are available in AppDesigner to do similar things. Archive and FTP are examples of this.

We expect the list of Toad windows which have been “Actionized” to grow. As always we encourage your feedback to help guide us.

Creating Apps

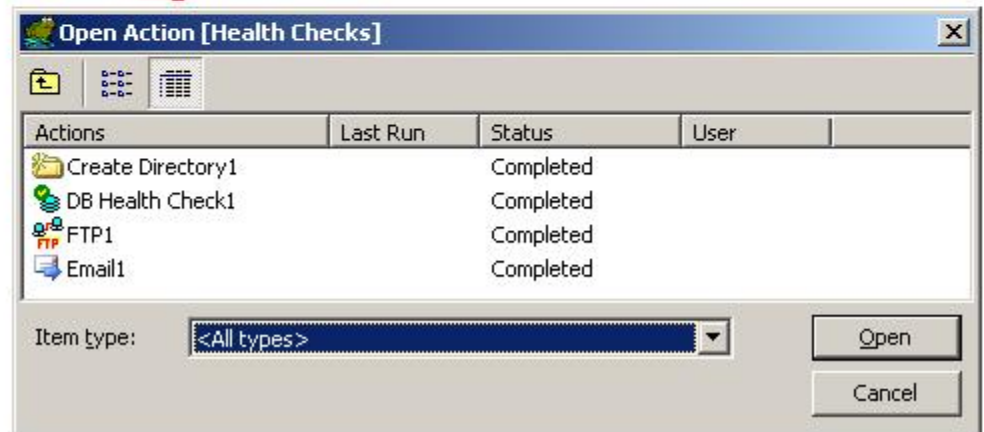
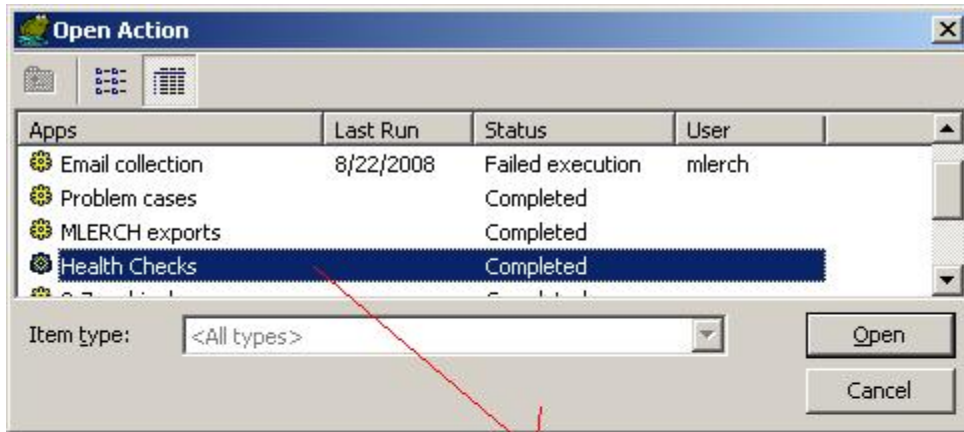
An App is a group of Actions designed to work as one executable unit. They are created and managed in the AppDesigner.

Apps are organized into user defined folders for better organization. The folder hierarchy is shown in the left side tree view. To create a new App, select the folder on the left where you want to place the new App then click the “Create new app” toolbar button. A new App with a default name will be created and selected. You now place the Actions you want in this App on the right side list view. Control the execution order by dragging and dropping them.

When the App is executed it will execute each Action in the list one at a time. If an Action fails, execution will stop.

Opening Apps and Actions

An App or Action can be opened through the File > Open Action dialog. If an App is opened, the QuickApp dialog (shown below) is opened. If an Action is opened its Properties dialog is opened.



Drill down into an App to display its Actions. If there are many Actions they can be filtered by type

Of course they can also be opened in the AppDesigner.

Executing Apps and Actions

There are a variety of ways Apps and Actions can be executed:

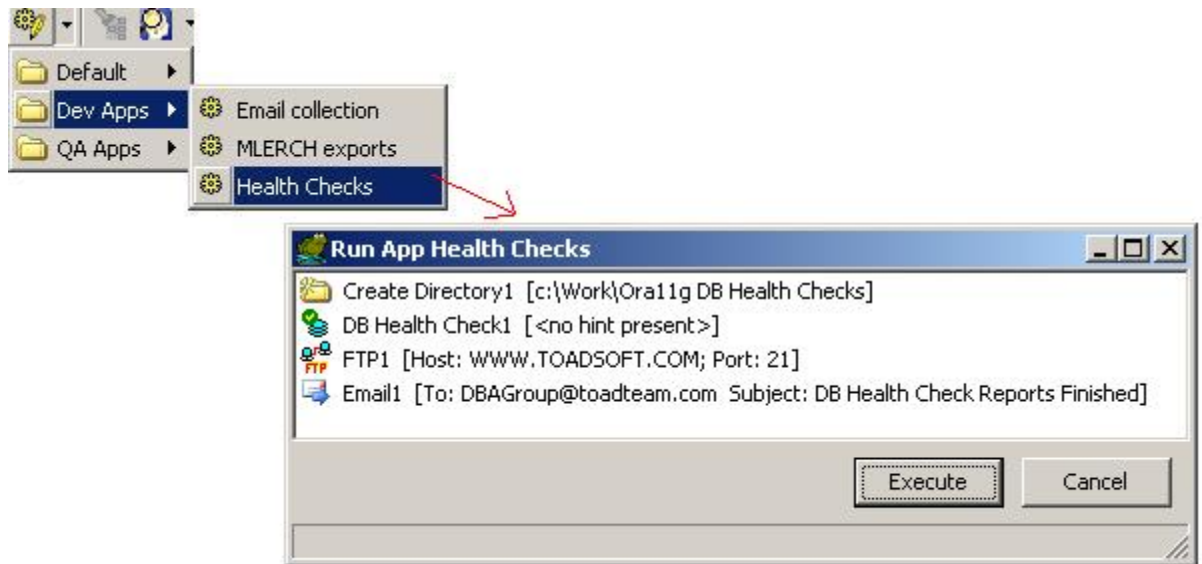
- Command-line, by passing the Action or App name to Toad.exe via the -A parameter. If an Action name is unique in your entire datafile it can be passed by itself, but we recommend you always qualify Actions by specifying the App in which they reside, separated by an arrow, e.g., "An App->An Action."

Examples:

```
Toad.exe -A "TNSPing Oracle 11 Server"
Toad.exe -A "An App to build Reports"
Toad.exe -A "Email Collection->Schedule Notification"
Toad.exe -A "Exports->Sales Report" "Send Status Email"
```

Any number of Apps or Actions can be listed on the command line. Surround them with double-quotes and separate each with a space.

- File > Open Action on the main menu
- Selecting the App from the Quick App toolbar button or the Utilities > AppDesigner menu flyout, where they are organized by folder. The following shows the QuickApp execution of an App which creates a disk folder then runs a DB Health Check on a database, outputting the results to files in the new folder. It then FTP's the result files up to a server and sends out an email to the team to alert them:



- By adding them as toolbar buttons (see “[Creating Action Toolbars](#)”):



- Through the AppDesigner

Logs and Messaging

Action Logs

Whenever an Action is executed, extensive information is stored in the Action Log. This information includes the Run number, Run Date, Run Time, the User who executed the Action, the Status of the Action (Success, Start Failure, Run Failure, Running) and Detail Messages the Action logs as it is performing the execution (for example, creating a connection, preparing a Query for execution and so on).

The Log Data for a single execution of an Action or App can be viewed in the “Single Run” tab of the App Designer. Subsequent executions will overwrite this data. To view the Run Data for all runs, select the All Runs tab. Detail messages within individual Actions can be viewed by double-clicking a Run Data line item.

The Execution Log tab on the left side of the App Designer displays the entire Action Log file contents. As you scroll through each Run Data line item, the individual line items for that entry are displayed in the memo beneath it.

Action Messages

An Action behaves somewhat differently in regards to user messages depending on whether it is executed through the Action Properties window, through the App Designer or through the command-line. For example, when the Run button is clicked in the Action Properties window, any validation errors (missing fields, e.g.) will be displayed in a message dialog box. When the same Action is executed via command-line, since message dialog boxes cannot be used the same message will be written to the log.

Action Recall

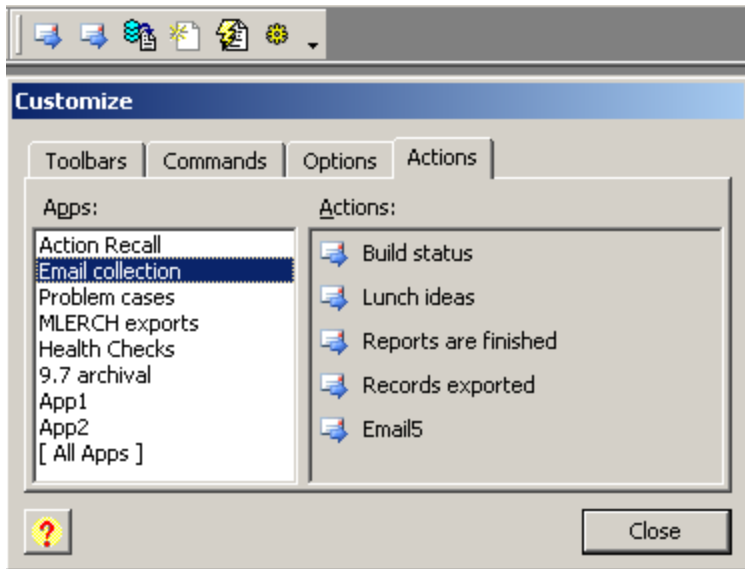
We have already seen in the Introduction how some Toad windows, after executing, place the internal Action they create into a special App named “Action Recall.” The number of Actions per Action Type which are saved can be controlled through an option found in Toad Options, General tab, “Save N Toad Actions per action type.” Once this limit is reached, newer Actions will push the oldest Actions off the stack. The age of an Action is defined by when it was last executed, not when it was created. Therefore Action Recall Actions of interest which you continue to execute will remain in the data file.

The special “Action Recall” App may not be renamed or deleted. You also cannot directly place Actions into it via the AppDesigner. Otherwise it is like any other App. Actions can be deleted from it, copied for use elsewhere or executed on the command line:

```
Toad.exe -A "Action Recall->Export Dataset14"
```

Creating Action Toolbars

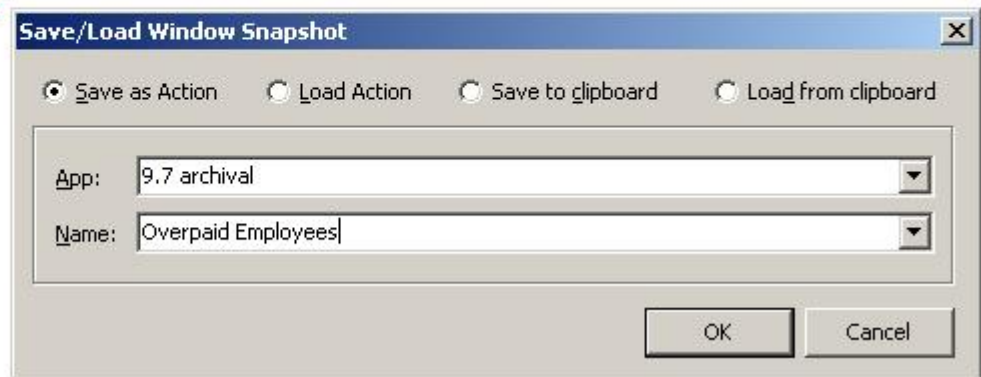
The Toolbar and Menu Customization dialog has an additional tab – Actions. This tab lists all of your current Apps on the left with the Actions they contain on the right. These items can be dragged & dropped onto existing or new toolbars to create Action Toolbars. Apps can also be dragged onto toolbars by choosing them after selecting the special “[All Apps]” item on the left.



Creating toolbars of Actions

Using Window Snapshots

On the status bar of each Actionized window is a camera icon. This opens the Save/Load Window Snapshot window:



The Snapshot Window has four functions:

- **Save as Action** – Select an App (or enter a new App name) and an Action Name. This will use the current window settings to create a new Action you can execute later or use as a “Save Settings” type of operation.
- **Load Action** – Select an App and an Action. Only Actions of the same type as the window will be shown in the dropdown. Once selected, the exact properties of the Action will be loaded into the window. This works like a “Load Settings” operation.
- **Save to clipboard** – Selecting this operation will disable the App and Action name fields. Click OK to have a temporary Action created, populated with the

underlying screen values and copied to the clipboard as plain text. Here is a truncated example of what an Action looks like in plain text:

```
object TarSaveGrid
  Enabled = True
  ID = 0
  ParentID = -1
  UserName = 'Export Dataset1'
  . . . . .
726F647563745C392E320B4175746F436F6E6E656374080C53617665506
17373
776F726408084661766F7269746508064D6574686F64020000000000}
  DatabaseObjects =
{545046300C54617244424F626A6563747300054974656D730A00000000
0000}
end
```

This text can be pasted directly into the AppDesigner and an Action will be created. It can also be used for the “Load from clipboard” operation, for example, if someone emails you an Action.

- Load from clipboard – if an Action’s plain text definition is on the clipboard it can be pasted into a window which matches its type. For example an Export Dataset Action can be pasted into the “Save Grid” window. It can also be pasted into the AppDesigner and a new Action will be created. This provides a way for you to share Actions with others and use their Actions in turn.

The Scheduling Subsystem

Scheduling has been enhanced throughout Toad to take advantage of Action technology. Specifically, whenever you click the Schedule button which appears on many windows, such as the Create windows, QSR is no longer used as an external tool to manage the job. Instead, an “Execute Script” Action is created. This ensures that one product, not two, is used to execute scripts, guaranteeing the same results you would achieve had you executed the script directly in Toad. It also allows you to retrieve the scheduled job to view the script which will execute, and alter its execution properties.

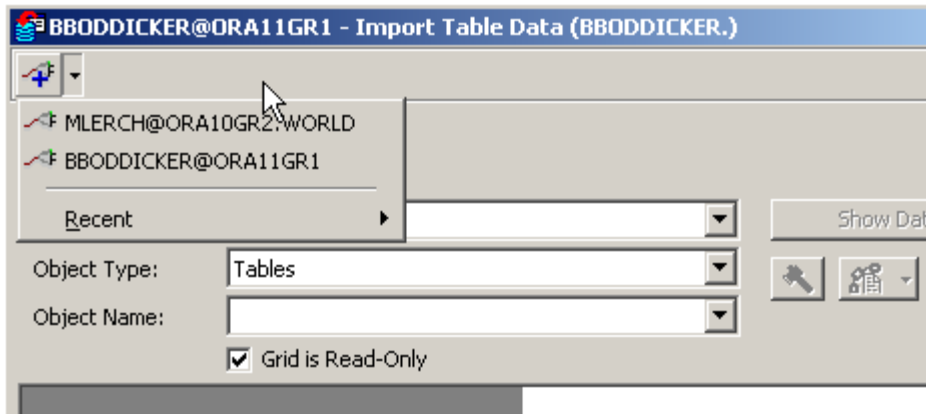
Scheduled Actions appear in the AppDesigner Scheduled Items tab, where you can view all scheduled actions or those for a given date. Right click on a scheduled item and you can change the schedule or change the Execute Script Action Properties.

Action Reference

This section will briefly review every Action available in Toad 9.7 as well as give an overview of characteristics common to many.

Actions Requiring Connections

Some Actions must be associated with a database connection to do their work. For example, the Actions on the Import/Export and DB Misc tabs of the Action Palette must all be tied to a connection. If you have an active connection in Toad, when you first open the Action Properties window for any of these Actions they will automatically become associated with your active connection. If you do not have an active connection, or you wish to change the association, you can do so by using the standard Connection button on the toolbar:



Select a Connection to assign it to an Action

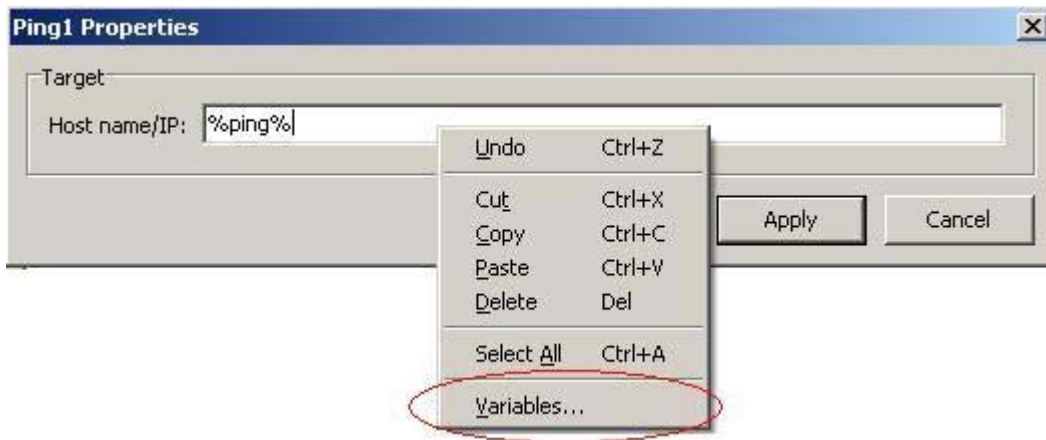
The Select Variable Window

New for 9.7 is the support of Variables in many places. By “Variables” we mean environment variables and user-defined Toad Variables (set through Options > Variables). Many of the Action Properties windows support Variable reference using the common convention of surrounding them with the percent symbol:

```
C:\Output Files\%ACTIVESESSIONUSER%_Export_%DATE%
```

How do you know which fields support a Variable? One way to know whether a plain edit box accepts a Variable is by right-clicking on it. Most edit fields which support Variables should have an additional “Variables” popup menu item which, when clicked, will open a list box showing all Variables. Select one and it will be inserted next to the current cursor position.

Only simple edit fields support the Variables popup window (and we may have forgotten to wire some up!) For non-edit fields, such as drop-downs, and those we’ve forgotten to wire up, you can still manually enter the Variable in many Action Property fields. We’ll be doing our best to support Variables everywhere possible in future Toad versions.



The Action Reference List

Import/Export Actions – Three to Get Data In & Out

- **Import Table Data** – Corresponds to the Database > Import > Import Table Data window
- **Export DDL** – Corresponds to Database > Export > Export DDL. When used as an Action there are additional buttons on the bottom. “Run” matches the green run arrow on the toolbar of the window as it appears when loaded from the main menu, and of course “Apply” commits any changes made to the Action’s properties.
- **Export Dataset** – Corresponds to a grid’s “Save As” window with the addition of an option to export objects. The Dataset tab allows the entry of a query or the selection of Materialized Views, Tables or Views.

DB Misc Actions – Database Actions Requiring a Connection

- **ANSI Join Syntax** – Converts a group of DML files to ANSI syntax. A number of users have been using Query Builder simply for the purpose of converting individual DML files to ANSI syntax. This Action facilitates doing this conversion for large groups of files. Specify the list of DML files and the output directory for the converted files.
- **Actionable Query** – Allows you to enter a SQL block or PL/SQL Anon block which returns a Boolean. If the query results in false the Action return code is false, otherwise true. This Action is useful in an If/Then block.
 - *If a SQL statement is entered* it will be prefixed with SELECT 1 FROM DUAL WHERE. For example, if you enter

```
(select ename from scott.emp where empno=7521)='WARD'
```

The query sent to Oracle by Toad is

```
SELECT 1 FROM DUAL WHERE (select ename from scott.emp where empno=7521)='WARD'
```

If a “1” is returned in the result set the Action returns True. You can then act on this with other subsequent Actions.

- *If a PL/SQL statement is entered* it will be executed as an Anonymous block and return True if the “RESULT” bind variable is set to 1, otherwise it will return False:

```
Declare
  v_sal NUMBER;
Begin
  select sal into v_sal from emp where ename='WARD';
  if v_sal > 10000 then
    :RESULT := 1;
  else
    :RESULT := 0;
  End If;
End;
```

- **Execute Script** – This Action allows for the execution of multiple script files or plain script text. Note that variables are supported: you can specify a %variable% as a filename or part of a file path and it will be substituted at run time. Variables are also supported in the Output directory/file field.
- **Object Search** – Corresponds to the main Toad Object Search window (Search > Object Search). The search parameters will be saved. When the Action executes the search will be performed. Future releases of Toad Actions may leverage the result set.
- **DB Health Check** – Corresponds to Database > Diagnose > DB Health Check. Note that the results will be written to the “Settings: Save results to file” field.
- **HTML Schema Documentation** – Corresponds to Database > Report > HTML Schema Doc Generator
- **Compare Schemas** – Corresponds to Database > Compare > Schemas

Utility Actions – Mirroring Toad’s Utilities Menu

- **Email** – Uses SMTP to send an email. Variables are supported in all fields.

- **Shell Execute** – Allows execution of an external application and is very similar to the “Toad Tools” window off the main toolbar. The macros shown on the properties window are supported in the Parameters field. In addition, variables are supported in all the fields, though for the 9.7 release the right-click menu on those fields does not show the “Variables...” menu item.
- **Archive** – Similar in nature to the Archive window, allows you to specify files to zip or unzip to a new or existing file. As of the 9.7 release variables are not supported in these fields.
- **FTP** – Similar to Toad’s FTP window but with a property designating Upload or Download of the specified files. Variables are supported in the Directory field.
- **Ping** – Performs a simple Ping against a Host or IP address. Variables not supported for 9.7.
- **TNS Ping** – Performs an Oracle TNS Ping against the selected database. Variables not supported for 9.7.
- **Service** – An Action to perform a Start, Stop or Toggle of a selected service.
- **Format Files** – Corresponds to the Format Files function of the Editor, allowing for the rapid formatting of many files at once. Variables supported everywhere.

File Management Actions – For Disk Files & Folders

Variables are supported in all these Actions.

- **Create Directory** – Creates a new Windows folder. If the folder exists it will not overwrite it.
- **Delete Directory** – Deletes a Windows folder. The operation will fail if the folder is not empty.
- **Move Directory** – Renames a Windows folder. The folder need not be empty.
- **Create File** – Creates a text file with specified content. Variables can be embedded in the content.
- **Delete File** – Deletes a file or files by the given name or wildcard specification.
- **Move File** – Moves a file or files by the given name or wildcard specification to the destination file name or folder.

- **Copy File** – Copies a file or files by the given name or wildcard specification to the destination file name or folder.
- **File Exists** – Tests for the existence of a file. The Action returns True if the file exists, False otherwise.

Control Actions – the Programmer's Action Group

The Actions in this group are intended to be used in conjunction with other Actions in an App; they have little meaning on their own.

- **Test Variable** – Tests a Variable for a value, returning True if the condition is met, False otherwise. This Action would typically be used with a control structure or to control the execution of subsequent Actions placed after it in an App. The conditions are <, <=, =, <>, >, >=. The Value field can accept Variables.
- **Set Variable** – Assigns a value to a variable.
- **Variable Prompt** – Displays an input window for a specified variable and assigns the value to the variable. This value remains throughout the current Toad session.
- **Message** – Offers a custom message box. The message content itself can contain variables. This Action is useful also from a simple debug perspective. The Action will return False if the “Cancel” button or equivalent is clicked.
- **Pause** – Pauses App execution for a specified interval of time. This is useful for operations which write out disk files or which otherwise can take substantial time to complete. Subsequent Actions may depend upon conditions for which they need to wait on the operating system.
- **Log Comment** – Writes a message into the Action log. Useful for debugging, and supports variables.

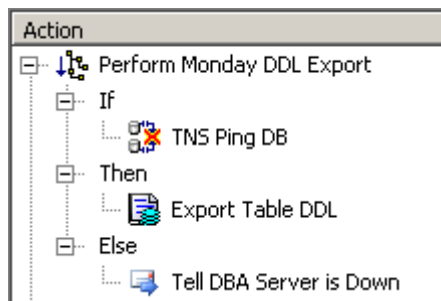
Control Structures

‘If/Then,’ ‘Repeat/Until’ and ‘While/Do’ are unique in their structure. They are the only Actions which have sub-nodes in the AppDesigner since they take collections of Actions as their language terms. When building these Actions, click on an Action in the Palette then click on the sub-node of the control Action to add an Action to it.

- **If..Then..Else** - Provides an “If Then Else” language structure for advanced Toad App development. Add the conditional Action(s) to the ‘If’ subnode, the Action(s) to execute upon true in the ‘Then’ subnode and the actions to execute upon false to the ‘Else’ subnode.

```
If [conditional Actions] Then [Actions if True] Else [Actions if False]
```

The proper way to read this Action is “If all of these actions return True, then execute all of these actions, otherwise execute these actions.” In other words, every Action in the “If” block will be evaluated. If they all evaluate to True, then all of the Actions under the “Then” block will execute. If they all evaluate to True, the return value of the main “If..Then” Action will be True. If any evaluate to False, the return value of the primary Action will be False. If any of the “If” Actions return False, all of the actions in the “Else” block will execute. Again, the return value of the primary “If..Then” Action will be True if all evaluate to True, False otherwise.



A simple example of If/Then/Else

- **Repeat..Until** - Provide a “Repeat” loop language structure for advanced Toad App development.

```
Repeat [statement Actions] Until [conditional Actions]
```

The Repeat Action executes its sequence of constituent Actions continually, testing the result of all the conditional Actions after each iteration. When all the conditional Actions execute successfully the Repeat Action terminates. The Action is always executed at least once because the conditional Actions are not executed and evaluated until after the first iteration. This differs from the While action, which executes its statement Actions only if the conditional Actions first all execute successfully.



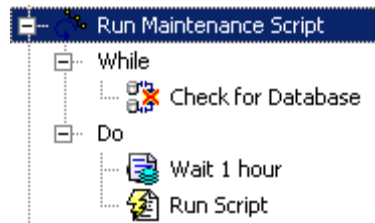
Download files from an FTP server until one of them contains a particular string we are looking for. The moment the string is found we end execution.

- **While..Do** - Provide a “While” loop language structure for advanced Toad App development.

While [conditional Actions] Do [statement Actions]

A While Action is similar to a Repeat Action, except that the conditional Actions are executed before the statement Actions are executed. Hence, if any of the conditional Actions are false, the statement Actions are never executed.

The While Action executes its constituent Actions repeatedly. As long as each of them succeeds, execution of the statement Actions continues.



Run a maintenance script every hour while the database remains up

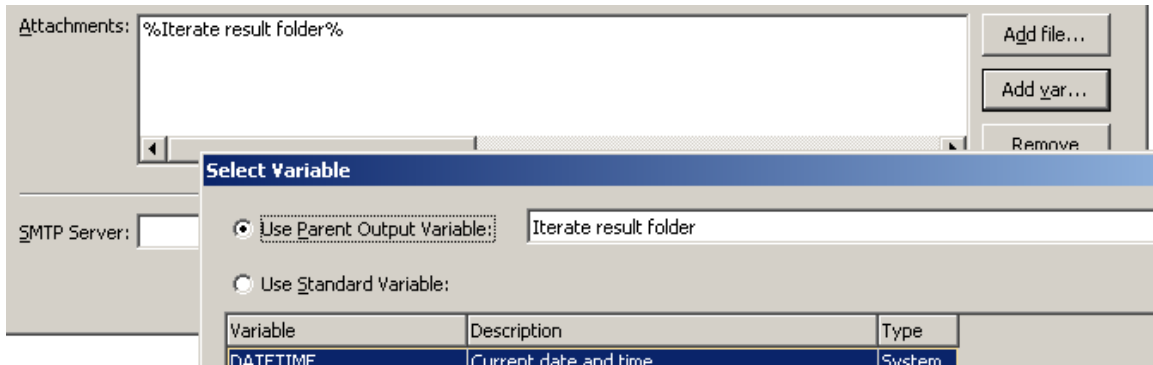
Iterators

An Iterator is a computer science term for an object which allows you to traverse through all the elements of a collection. There are Action Iterators to traverse through all the files in a folder, all the lines in a file and all the lines in a list of text. For each of these items, all the sub-Actions – nodes under the Iterator node in the tree – will be executed. For example, if you create a Folder Iterator Action, point it to a folder with 10 files, then place an Email Action under it, it will send 10 emails – one for each file in the folder. And each Iterator writes the current element in the collection into a variable with the same name as itself so all the child Actions can reference it.

Let’s look at a simple example to understand this clearly, since each Iterator works with the same behavior. The following shows a Folder Iterator Action which points to the “C:\Mark\Work\Toad\Source” folder. For each file in that folder it will execute the Email Action named “Send result file:”



The Email Properties window supports variables everywhere. The Attachment section has an “Add var..” button which opens the Variable Select window. When the Variable Select window is opened for an Action which has an Iterator as a Parent, it shows all Parent variables (which correspond to their Action Names) in a special drop-down:



Because the Email Action is a child of an Iterator Action named “Iterate result folder,” the Variable Select window lets us choose this special temporary variable. When the Email is executed as part of the Iterator Folder execution, the variable will be expanded to be the value of the current file. It will be added as an attachment to the email.

- **File Iterator** – Specify a file. The Iterator will invoke all child Actions for each line in the file. Each line in the file will be written to a variable by the same name as the File Iterator.
- **Folder Iterator** – Specify a folder, a folder specification and whether to recurse subdirectories or not. The Iterator will invoke all child Actions for each file in the folder. Each filename will be written to a variable by the same name as the Folder Iterator.
- **List Iterator** – Create a list of items, perhaps filenames, folders or simple text values. The Iterator will invoke all child Actions for each item in the list. The text of each item will be written to a variable by the same name as the List Iterator.