User-Defined Tools

To extend its functionality ToDoList lets you configure tools to perform action and operations that are outside the scope of ToDoList as a generalised task manager.

A User-Defined Tool (UDT) is a template which specifies a tool (application, script or batch file) together with additional command-line parameters and information on how to display the tool in ToDoList's user interface.

What can be done with a UDT?

- Reports can be generated, emailed, or faxed.
- Data can be passed to other project management systems.
- ERP systems can incorporate the data into schedules.
- Data can be sent via web service to remote systems.
- Task values can be copied from one attribute to another within ToDoList.
- Task values can be modified within ToDoList.

Creating a UDT

Note: For detailed information on configuring a UDT see (Menu Bar > Tools Menu > Preferences > User Defined Tools.

The goal is to create a command template that can be populated with data at run-time, and then executed from the DOS prompt. It would look something like this:

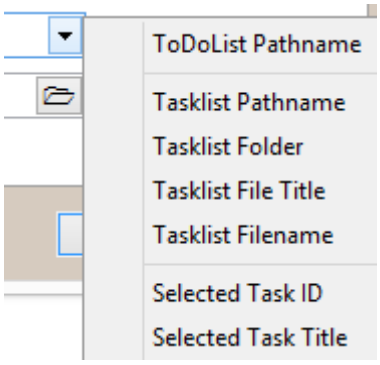
```
c:\path\to\program\name.exe -switch1 -switch2 "data1" -switch3 "data2"
```

The switches and data are whatever is required for the 'name.exe' program to work. These are the tool's 'Arguments'.

Arguments
A set switches and/or data that will be passed to the tool via its command-line. The down-arrow button attached to this field will display a list of “placeholder” variables for data that ToDoList will substitute when you run the tool. For example an argument might be set as follows:

```
/taskid=$(selTID)
```

The $(selTID) is inserted by ToDoList after clicking the down-arrow button to show a list of available



data. In that list is “Selected Task ID”. When clicked, the placeholder “$(selTID)” is inserted into the arguments field. So with the above argument set, at runtime, the value of the current selected task will be substituted, maybe #217, and the final command will look something like this.

```
c:\path\my_script.bat /taskid=217
```

Whatever happens in that command is outside the scope of ToDoList and this documentation. It's up to you to find or create scripts or programs which do things that you want, and then to get ToDoList to provide those programs with required data to provide the results you seek.

User Placeholders
Placeholders prefixed by 'user' will result in the user (you) being prompted to enter information when the UDT is executed. This is useful where the information is not known in advance or it frequently changes. User placeholders typically take 3 additional arguments:

- a unique variable name (eg. var_text1)
- a prompt string (eg. “Enter your username”) This should be in quotes it is not required.
- an optional default string to display (eg. “anonymous”) This should be in quotes it is not required.

Example of a User Placeholder:

- -clientname $(usertext, vt1, “Name of client”,“Unknown”)
  - The user selects the UDT from the menu or toolbar. A small prompt window is shown with the title set to the name of the UDT. A textbox is shown with the label “Name of client”, and a default value of “Unknown”.
  - The variable name is completely unimportant. It just needs to be unique of the names used in your UDTs. In this case vt1 is used for variable text, and it's assumed that other fields might get names like vt2, vt3, etc.
  - Note that the placeholder name and the variable name are Not in quotes, but the text data is.
  - The command line will get _-clientname Some name_ with whatever name was entered, or _-clientname Unknown_ if no name is entered.

There are no quotes around the user data for the command-line. That could cause a problem with the program that processes the data. The next example adds quotes:

- -clientname **“**$(usertext, vt1, “Name of client”,”Unknown“)**“**
  - Note the quote before the dollar sign and at the end. This results in the following:
  - -clientname **“**Unknown**“**

Tool Examples

The following (rough) examples illustrate possible uses of the tools system and one possible way of solving each challenge.

- Challenge 1 : Display the raw active tasklist in your browser
    1. Set the tool path to point to your default browser .exe file.
    2. Enter $(pathname) in the arguments field.
- Explanation: All this does is call your browser, passing it the full pathname to the active tasklist.

- Challenge 2 : Render the active tasklist to HTML using an XSL transform
    1. Create a batch file containing %1 %2 %3 -o %4 on line 1 and %4 on line 2, and set the tool path to point to this file.
    2. Use the following Argument. This is all one line! :

```
  "$(userfile, var_msxsl, "Path to Msxsl.exe")" "$(pathname)" "$(userfile, var_xslfile, "Path to Xsl file")" "$(folder) $(filetitle).html"
```

Note that there are two *userfile* placeholders, and that fields are surrounded by quotes as described above.

- Explanation:
    - The first argument prompts the user to browse to msxsl.exe, which is the rendering engine we will be using.
    - The second argument is just the full pathname of the active tasklist which we are going to render.
    - The third parameter prompts the user for the XSL file with which to carry out the transform.
    - The last argument is the output file (just the tasklist pathname with a .html extension).
    - The second line in the batch file acts to display the resulting html file in your default browser.

- Challenge 3 : Connect to Bugzilla via the External ID task field
    - *Work in Progress.* Details about his example can be found .... ??

- Explanation: This allows you to open Bugzilla with the task's External ID as bug number.