Child Support Functions List

This document will explain how to use the standard and custom functions for Child Support (PRISM) scripting.

# Standard VBScript Functions

See the W3schools' website for a good description of standard VBScript functions. This website is located at <http://www.w3schools.com/vbscript/vbscript_ref_functions.asp>.

# Standard BlueZone Script Functions

These functions are provided by the developer of BlueZone, for use in the BlueZone Script Host. They will not work in other programs, such as Excel macros or AS400 scripting.

## CloseSession [Session Type], [Session Identifier]

Closes a BlueZone window.

## EMConnect [SessionShortNameStr]

Connects to a BlueZone session. For most scripts, this can simply be called like so:

EMConnect ""

Using the function like that will connect to whichever screen the user started the script on. So, for example, session "A" (or "S1") might be on the left screen, and session "B" (or "S2") might be on the right. If the user starts the script in session "B", EMConnect will instigate whatever code follows in that session, and likewise for session "A".

There may be instances where you want to specify a screen. For example, if you are moving information from PRISM to MAXIS. In that case, you can use the PRISM features (assuming your staff log into PRISM first) by using the function:

EMConnect "A"

And then when it's time to go to the second MAXIS window, use the function:

EMConnect "B"

It's important to remember: when using these functions in this way, to include some kind of "program check" logic, to make sure you are in PRISM, MAXIS, or MMIS, when programming the script.

No matter which of the three above ways you use the function, remember to use it. Without this, the script never connects to BlueZone, and cannot execute any code.

## EMFocus

This will make the BlueZone screen pop-up over all other screens on the computer. Useful if you have a script that is bringing up a document, and you want to force BlueZone to "stay on top" of the windows.

## EMGetCursor [Row Variable], [Column Variable]

This will scan BlueZone and determine what the row and column the cursor is on. This is useful if you need to use that variable for EMReadScreen or EMWriteScreen.

## EMPrintScreen

This will trigger a "print screen" event, defaulting to the system default printer. BlueZone does not appear to contain scripting functionality for "print multiple screens".

## EMReadScreen [Variable to Read to], [Length of Characters to Read], [Row to Read], [Column to Read]

This function will read part of the current BlueZone window, and insert the contents into a variable of your choosing. For example:

EMReadScreen income\_type, 2, 3, 8

In this case, it will read 2 characters, starting at row 3 and column 8. Whatever those two characters are, will become a text string in the variable income\_type.

This is an absolutely critical function to learn for BlueZone Scripts in Child Support. This function forms the basis of form autofill, calculating income amounts, or prepopulating case note templates.

## EMReceiveFile [File to Receive From FTP]

This is not currently being used in any DHS or Anoka County BlueZone Scripts at this time. Theoretically, it could be used for interagency file transfers, but additional research is required before we can look at this function.

## EMSearch [Variable or Text to Search For], [Row to Start Searching], [Column to Start Searching]

This function will search for a string somewhere in the BlueZone window. This is very useful for "finding" text somewhere on the screen, such as case numbers and footer months, especially if that text "moves around" from screen to screen (case numbers are a good example of this phenomenon in PRISM).

For example:

PRISM\_row = 1  
PRISM\_col = 1  
EMSearch "Case: ", PRISM\_row, PRISM\_col

This could be used to see if the phrase "Case: " is found anywhere on the screen. By predeclaring PRISM\_row and PRISM\_col to be "1", then it will be easier to find out if the text was "found" on the screen; if it wasn't, PRISM\_row (or PRISM\_col) will be "0" after this function. If it was "found", you'll have the row and column as variables in your script, and you can easily use that info with an EMReadScreen function to read text.

## EMSendFile[File to Send Via FTP]

Again, at this time, this function is not being used.

## EMSendKey[Keystrokes to Send to BlueZone]

This function can be used to send keystrokes to a BlueZone window. These keystrokes can be text, or key commands such as "enter" and "F1". For the most part, to send text to a BlueZone window, it is faster and more accurate to use the EMWriteScreen command. Also, for the most part, sending keystrokes has been superseded by the custom functions, which are detailed below. You may find that you need to use EMSendKey if you are requiring that the text you are writing to BlueZone has to "wrap around" multiple lines. But, in many of these cases, there are custom functions (detailed below) written to fulfill the same process in fewer steps.

## EMSetCursor [Row to Set Cursor], [Column to Set Cursor]

This will place the cursor on a particular row and column. This can be used in conjunction with EMSendKey to send text to a particular field, or to execute a specific command in a particular field.

## EMWaitCursor [TimeoutVal], [RowVal], [ColumnVal], [ExtraWait]

Not currently used.

## EMWaitForText [TextStr], [RowVal], [ColumnVal], [TimeOut]

Not currently used.

## EMWaitReady [Time to Wait for Refresh], [Additional Time to Wait]

This function is used frequently throughout the FUNCTIONS FILE to indicate that the system should wait a certain amount of time before proceeding to the next line. It differs from the Pause function, in that it checks the state of the BlueZone window to see if the screen has "refreshed", meaning new screen data has been received from the central server.

Because the time is in milliseconds, using anything other than the default "0", is not recommended, as the screen delay will likely exceed any parameter. By using "0", the script will just determine the appropriate time, based on a constant monitoring of the screen state.

Recommended usage:

EMWaitReady 0, 0

## EMWriteScreen [String or Variable to Write], [Row to Write], [Column to Write]

This function is absolutely critical. It will write any text or variable to the BlueZone window, which is necessary for filling out screens.

Example:

EMWriteScreen "Hello there", 4, 6

This would write the string "Hello there" at row 4 and column 6. You could use a variable as well, simply omit the quotes and drop the variable in that field.

## OpenSession [SessionTypeVal], [SessionIdentifierVal], [ConfigFileStr], [TimeoutVal], [WaitPaints]

Not being used at this time.

## Pause [Milliseconds to Pause]

This function will simply pause a script for a specified amount of time.

## Run [Command to Execute]

This function will execute a command via the command line. It can be used for advanced functionality such as opening up a document or hyperlink, or a program on your computer.

## StopScript

This function ends a script, without executing any of the code which takes place after. It is frequently used for cancel buttons.

# FUNCTIONS FILE custom functions

In addition to the custom functions written above, if you utilize the FUNCTIONS FILE, you will gain additional functionality, as explained below.

## attn

This function will issue the "attn" command, popularly known as "escape". This can be used to navigate between screens in MDHS.

## back\_to\_self

This function will navigate back to the main SELF menu of a MAXIS screen. You have to be in MAXIS for it to work.

## end\_excel\_and\_script

Assuming you've declared your Excel document with the name "objExcel", this function will close the Excel document, and end the script at the same time. Otherwise, just use the script\_end\_procedure custom function described below.

## enter\_PRISM\_case\_number([Case Number Variable])

This function will enter a PRISM case number in the appropriate field in PRISM, and remove the dash from entry. This uses an EMSendKey as opposed to an EMWriteScreen.

## find\_variable([String], [Variable], [Length of New Variable])

This function will find a variable by condensing the EMSearch function (described above) into a single line. For example:

call find\_variable("Case: ", case\_number, 13)

This would perform an EMSearch for the text "Case: ", and once found it would enter the 13 characters after that string into the case\_number variable.

## fix\_case([Variable to Fix], [Smallest Length to Skip])

This function will change the case of a phrase, from all uppercase to correct capitalization. It contains an additional variable to allow certain lengths to be "skipped" from this capitalization. So, as an example:

actual\_address = "742 EVERGREEN TER"  
call fix\_case(actual\_address, 1)

The above would create the output "742 Evergreen Ter". But in the following example:

call fix\_case(actual\_address, 3)

The above would skip the "TER" of the original string, so you'd end up with the output "742 Evergreen TER". This can be helpful when dealing with strings which are likely to contain acronyms, such as some addresses, bank titles, insurance companies, etc. In any case, you must use a variable to fix (and not a pure string, as the function can't modify a pure string, so would create an error).

## navigate\_to\_MAXIS\_screen([Function], [Screen])

MAXIS screens are organized into functions and screens, such as "REPT/ACTV", and "STAT/ADDR". This function will streamline navigation, which could be useful for Child Support scriptwriting if you find yourself writing a script which navigates to a MAXIS screen for scanning a case, looking up info, etc. You must be in MAXIS for this function to work correctly.

## navigate\_to\_PRISM\_screen([Screen])

Navigates to a PRISM screen. You must be in PRISM for this function to work correctly.

## PF[number]

This simple function will execute a PF keystroke of your choice. Examples of usage would be:

PF1

or:

PF12

This is not just one function, but a couple dozen functions, and is explained here in condensed form for the sake of saving space.

## PRISM\_case\_number\_finder([Variable for PRISM Case Number])

This function will find a PRISM case number on the screen, by condensing the functionality of EMSearch into a single, simple function. Here is the most common usage:

call PRISM\_case\_number\_finder(PRISM\_case\_number)

## PRISM\_case\_number\_validation([Case Number Variable to Validate], [Variable for Outcome])

This will validate if a PRISM case number entered by a worker (or found by a script) is a valid formatted case number. For example:

call PRISM\_case\_number\_validation(PRISM\_case\_number, case\_number\_valid)  
If case\_number\_valid = False then  
 MsgBox "This is not a valid case number"  
 StopScript  
End if

Let's say our PRISM\_case\_number variable was "13-01", it would write the case\_number\_valid variable as False, and stop the script. But, if it was "0000000013-01", it would write the case\_number\_valid variable as True, and continue working. You can use this to ensure that a worker has entered in a correct case number before moving on to new screens/dialogs.

## run\_another\_script([Script Path])

If you want one script to run another script, you can use this code. Just put the path of the script you're going to run in the parenthesis, within quotes.

## script\_end\_procedure([Closing Message])

While StopScript will close a script, it does not contain language to write to a usage statistics database, nor does it contain a section to add a closing pop-up. This will do both things by default. To include a message, use it as so:

call script\_end\_procedure("Success! All cases updated successfully.")

To skip the message (but still get the benefit of logging usage statistics), simply leave the message area blank, but include the set of quotes, like so:

call script\_end\_procedure("")

## script\_end\_procedure\_wsh([Closing Message])

This is identical to the script\_end\_procedure function, but includes different language for the few scripts that are written to run outside of BlueZone. At this time, it's only used on "ACTIONS – initialize scripts", which is an Anoka County exclusive script.

## step\_through\_handling

Details coming soon.

## transmit

Sends a transmit key to BlueZone.

## write\_editbox\_in\_PRISM\_case\_note([Bulleted text], [Editbox to Write to CAAD], [Spaces for Indenting])

This will write the contents of an editbox into the current line of a PRISM CAAD note. To use, you should already have the cursor placed on the correct line of CAAD (so you probably want to start your first line by using the EMSetCursor command, explained above). The function contains logic to determine if you're reaching the end of a line (and if so, it'll kick the text to the next line after indenting), or the end of a page (if so, it'll jump to the next page). This makes CAAD noting much simpler. Here's an example:

call write\_editbox\_in\_PRISM\_case\_note("NCP name", NCP\_first\_and\_last, 6)

This will create a bullet item called "NCP name:" (notice it puts the colon in for you), enter the NCP\_first\_and\_last editbox, and if it goes over a single line, it will indent the next line 6 characters.

## write\_new\_line\_in\_PRISM\_case\_note([String or Variable to Write to New Line])

Like the write\_editbox\_in\_PRISM\_case\_note function, this will write a single line into a CAAD note, and will check to see if you're reaching the end of a page. If you're reaching the end of a page, it'll go to the next page. This is helpful for times when you do not want to have a bulleted item, or you want to have a text item noted without a colon.