## Chapter 10:

# Async Protocol

The Async Protocol (protocol ID 13) consists of atoms that control the communication flow between the host and client. Atoms of the Async Protocol also provide the following miscellaneous functions:

- Displaying an alert dialog box
- Sending a message to let the client tools know the member is offline
- Sending a message to let the client tools know the member is online
- Playing a sound if the member has a sound card

## Async Protocol Atoms

The Async protocol atoms are described in alphabetical order in the rest of this chapter.

# atom$async_alert

## 7 ($07)

**atom$async_alert** displays a Windows alert box with the specified text string as the alert message. The flag parameter specifies the style of alert box to display.

**Note**: **atom$async_alert_raw** is the atom for transferring data in form_edit. **atom$async_alert_raw** is not documented elsewhere because the description and syntax are the same as **atom$async_alert**.

### Syntax

```
atom$async_alert <alert_style> <alert_text>
```

`<alert_style>` Specifies a 1 byte alert-style flag. If the high bit is set on the style flag, the alert pops the top form before it comes up. Alert styles and their values are as follows

```
1. ALERT_TYPE_INFO

2. ALERT_TYPE_ERROR

3. ALERT_TYPE_POP_INFO

4. ALERT_TYPE_POP_ERROR

5. ALERT_TYPE_WARNING

6. ALERT_TYPE_POP_WARNING
```

`<alert_text>` Specifies an alert text.

### Return Value

Unchanged.

### Example

The following example sends an informational alert to the member confirming that the member has updated the language preference list:

```
atom$uni_start_stream
atom$async_invoke_language_pref
```

```
atom$uni_end_stream
atom$uni_start_stream
atom$uni_next_atom_typed <ascii>
```

**`atom$async_alert <info> "Your language list has been updated."`**

# atom$async_alert_add_date_time

## 33 ($21)

**atom$async_alert_add_date_time** adds the host date and time integer stamp for the host process.

### Syntax

```
atom$async_alert_add_date_time_<date time>
```

`<date time>` Specifies the host date time stamp integer value.

### Return Value

Unchanged.

### Example

The following example adds the host date time integer value to an alert message:

```
atom$uni_start_stream <00x>
atom$async_alert_start
atom$uni_start_stream
atom$async_alert_add_date_time
atom$async_alert_add_text <text>
atom$async_alert_end <00x>
atom$uni_end_stream
```

# atom$async_alert_add_text

## 32 ($20)

**atom$async_alert_add_text** adds text to the alert box.

### Syntax

```
atom$async_alert_add_text <text>
```

| | |
|---|---|
| `<text>` | Specifies the text string. |

### Return Value

Unchanged.

### Example

The following example adds text to an alert message:

```
atom$uni_start_stream <00x>
atom$async_alert_start
atom$uni_start_stream
atom$async_alert_add_date_time
atom$async_alert_add_text <text>
atom$async_alert_end <00x>
atom$uni_end_stream
```

# atom$async_alert_end

## 34 ($22)

**atom$async_alert_end** ends the alert box creation action.

### Syntax

```
atom$async_alert_end
```

### Return Value

Unchanged.

### Example

The following example ends the modifications of an alert message:

```
atom$uni_start_stream <00x>
atom$async_alert_start
atom$uni_start_stream
atom$async_alert_add_date_time
atom$async_alert_add_text <text>
atom$async_alert_end <00x>
atom$uni_end_stream
```

# atom$async_alert_start

## 31 ($1F)

**atom$async_alert_start** builds an async alert box of multiple lines.

### Syntax

```
atom$async_alert_start
```

### Return Value

Unchanged.

### Example

The following example starts the modification of an alert message:

```
atom$uni_start_stream <00x>
atom$async_alert_start
atom$uni_start_stream
atom$async_alert_add_date_time
atom$async_alert_add_text <text>
atom$async_alert_end <00x>
atom$uni_end_stream
```

# atom$async_allow_switch_screen_names

## 55 ($37)

**atom$async_allow_switch_screen_names** lets the member switch screen names without losing the connection to the host. Use this atom to determine whether the member has permission to switch screen names. The menu calls the atom when the member chooses the **Switch Screen Names** button and the atom then broadcasts a message to all tools. Any tool not ready to switch screen names returns `FALSE`.

### Syntax

```
atom$async_allow_switch_screen_names
```

### Return Value

Returns nonzero if it is allowed to enable switch screen names.

### Example

The following example allows the member to switch screen names without losing connection to the host:

```
atom$async_allow_switch_screen_names
atom$if_last_return_false_then 3 4
atom$uni_invoke_no_context 41-42465
atom$uni_sync_skip 3
atom$sm_send_token_arg 6Ex 74x
atom$uni_sync_skip 4
```

# atom$async_auto_launch

## 38 ($26)

**atom$async_auto_launch** continues the navigation process to automatically go to a web site when AOL is invoked from a hyperlink or external application. This atom is sent by the host as part of the logon sequence. It is the atom that is used most often to deliver sign-on tools on demand (TODs).

### Syntax

```
atom$async_auto_launch
```

### Return Value

Unchanged.

**Note: atom$async_auto_launch** is a host-only atom used for continuing the sign-on process. You cannot use this atom in normal forms development.

# atom$async_display_errors

## 41 ($29)

**atom$async_display_errors** displays the error messages stored on the client.

### Syntax

```
atom$async_display_errors
```

### Return Value

Unchanged.

### Example

The following example displays the error messages stored on the client:

```
atom$uni_start_stream
atom$man_set_context_relative 30
atom$async_display_errors
atom$man_end_context
atom$man_update_display
atom$uni_end_stream
```

# atom$async_error_box

## 6 ($06)

**atom$async_error_box** displays a Windows dialog box with an alert message.

**Note:** The functionality of this atom has been replaced by **atom$async_alert**.

### Syntax

```
atom$async_error_box <error number_string>
```

| | |
|---|---|
| `<error number_string>` | Specifies an error number string. |

### Return Value

Unchanged.

### Example

The following example displays a Windows error box with an alert message:

```
atom$<man_set_context_relative> 1
atom$man_get_attribute prot$mat atom$mat_bool_invisible
atom$if_last_return_false_then <15>
atom$async_error_box <"Your error here">
atom$uni_abort_stream
atom$uni_sync_skip 15
```

# atom$async_exec_app

## 25 ($19)

**atom$async_exec_app** executes the application whose text string you provide.

### Syntax

```
atom$async_exec_app <application file string>
```

| | |
|---|---|
| `<application file string>` | Specifies an application file to execute. |

### Return Value

Unchanged.

### Example

The following example executes an application that takes the member to the designated Intranet site:

```
atom$async_exec_app <"start.exe http://www2.ops.aol.com/
Tech-Docs/unr/index.htm">
```

# atom$async_exec_context_help

## 14 ($0E)

**atom$async_exec_context_help** invokes the AOL help file, **aol.hlp**, and takes you to the part of the file related to the query.

**Note**: **atom$async_exec_context_help_raw** is the atom for transferring data in form_edit. **atom$async_exec_context_help_raw** is not documented elsewhere because the description and syntax are the same as **atom$async_exec_context_help**.

### Syntax

```
atom$async_exec_context_help <help text>
```

| | |
|---|---|
| `<help text>` | Specifies a context. |

### Return Value

Unchanged.

### Example

The following example invokes the AOL help file, **aol.hlp**, and takes you to the part of the file related to mail:

```
atom$man_start_object <trigger, "Help">
atom$act_replace_select_action
atom$uni_start_stream
atom$async_exec_context_help "mail"
atom$uni_end_stream
atom$man_end_object
```

# atom$async_exec_help

## 13 ($0D)

**atom$async_exec_help** invokes the AOL help file, **aol.hlp**; this atom is statically defined to always open the **aol.hlp** file.

### Syntax

```
atom$async_exec_help
```

### Return Value

Unchanged.

### Example

The following example invokes the AOL help file, **aol.hlp**, and takes you to the part of the file related to the query:

```
atom$man_start_object <trigger, "Help">
atom$act_replace_select_action
atom$uni_start_stream
atom$async_exec_help
atom$uni_start_stream
atom$man_end_object
```

# atom$async_exec_help_file

## 16 ($10)

**atom$async_exec_help_file** invokes the AOL help file, **aol.hlp**, or another help file that you designate. You have the option of putting in a text string to go to a particular help file or open the **aol.hlp** file.

### Syntax

```
atom$async_exec_help_file <file name string>
```

`<file name string>` Specifies a particular help file.

### Return Value

Unchanged.

### Example

The following example invokes the AOL help file, **aol.hlp**, and takes you to the part of the file related to the query:

```
atom$man_start_object <trigger, "Help">
atom$act_replace_select_action
atom$uni_start_stream
atom$async_exec_help_file
atom$uni_start_stream
atom$man_end_object
```

# atom$async_exit

## 0 ($00)

**atom$async_exit** sends a `MSG_NOTIFY_EXIT_REQUEST` to all registered tools. If the result is `RESULT_OK`, an alert, asking the member whether or not to sign off or exit, is displayed.

### Syntax

```
atom$async_exit
```

### Return Value

Unchanged.

### Example

The following example starts the process of exiting a member from the application:

```
atom$start_object <trigger> "Exit"
atom$act_replace_select_action
uni_start_stream_wait_on
atom$async_exit
atom$uni_end_stream
atom$man_end_object
```

# atom$async_exit_aux

## 0 ($00)

**atom$async_exit_aux** causes the online service application to sign off (if online) or exit (if offline). This atom causes the sign-off process to bypass popup functionality.

### Syntax

```
atom$async_exit_aux
```

### Return Value

Unchanged.

### Example

The following example causes the online service application to sign off (if online) or exit (if offline):

```
atom$start_object <trigger> "Exit"
atom$act_replace_select_action
atom$uni_start_stream_wait_on
atom$async_exit_aux
atom$uni_end_stream
atom$man_end_object
```

# atom$async_force_off

## 17 ($11)

**atom$async_force_off** completely drops the line to the host.

### Syntax

```
atom$async_force_off
```

### Return Value

Unchanged.

### Example

The following example completely drops the line to the host and sends a message to the member that says "Thank you for using America Online":

```
atom$man_strat_object <trigger>
atom$act_replace_select_action
atom$uni_start_stream
atom$man_close_update
atom$async_force_off <"Thank you for using America Online.">
atom$uni_end_stream
atom$man_end_object
```

# atom$async_get_alert_result

## 24 ($18)

**atom$async_get_alert_result** returns the result of an alert box.

### Syntax

```
atom$async_get_alert_result
```

### Return Value

Turns to l ("el") value.

### Example

The following example returns the result of an alert box, which offers a choice between true or false:

```
atom$uni_start_stream
atom$async_get_alert_result
atom$if_last_return_false_then <11, 12>
atom$uni_sync_skip <11>
atom$man_end_context
atom$uni_sync_skip <12>
atom$act_get_db_value <14x, ffx, 00x, 04x>
atom$if_last_return_true_then <1, 2>
atom$uni_sync_skip <1>
atom$uni_end_stream
```

# atom$async_get_screen_name

## 29 ($1D)

**atom$async_get_screen_name** returns the screen name as the l ("el") value.

### Syntax

```
atom$async_get_screen_name <screen name>
```

`<screen name>` Specifies a screen name.

### Return Value

To whatever the current screen name.

### Example

The following example returns the screen name to whatever the current screen name happens to be:

```
atom$act_replace_select_action
atom$uni_start_stream
atom$async_get_screen_name
atom$uni_use_last_atom_string prot$async atom$async
<async_storename>
atom$man_end_context
atom$man_update_display
atom$uni_end_stream
```

# atom$async_invoke_language_pref

## 36 ($24)

**atom$async_invoke_language_pref** invokes an operating system call to select a language preference.

### Syntax

```
atom$async_invoke_language_pref <language preference string>
```

| | |
|---|---|
| `<language preference string>` | Specifies a language preference. |

### Return Value

Unchanged.

### Example

The following example invokes an operating system call to select a language preference:

```
atom$uni_start_stream
atom$async_invoke_language_pref
atom$uni_end_stream
```

# atom$async_invoke_timezone_pref

## 35 ($23)

**atom$async_invoke_timezone_pref** invokes an operating system call to select a time zone preference.

### Syntax

```
atom$async_invoke_timezone_pref <timezone preference string>
```

| | |
|---|---|
| `<timezone preference string>` | Specifies a time zone. |

### Return Value

Unchanged.

### Example

The following example invokes an operating system call to select a time zone preference:

```
atom$uni_start_stream
atom$async_invoke_timezone_pref
atom$uni_end_stream
```

# atom$async_is_client_32bit

## 40 ($28)

**atom$async_is_client_32bit** verifies that the client is a 32-bit microprocessor.

### Syntax

```
atom$async_is_client_32bit
```

### Return Value

True or false as l ("el") value.

### Example

The following example verifies that the client is a 32-bit microprocessor:

```
atom$man_set_context_relative <1>
atom$async_is_client_32bit
atom$if_last_return_false_then <11, 12>
atom$man_append_data "16 bit"
atom$uni_sync_skip <11>
atom$man_append_data "32 bit"
atom$uni_sync_skip <12>
atom$man_end_context
```

# atom$async_is_current_screenname

## 48 ($30)

**atom$async_is_current_screenname** verifies the current screen name.

### Syntax

```
atom$async_is_current_screenname
```

### Return Value

Unchanged.

### Example

The following example verifies the current screen name:

```
atom$man_set_context_relative <2>
atom$idb_set_context <20-0-16>
atom$idb_offset 1
atom$idb_get_string
atom$uni_save-result
atom$uni_use_last_atom_string prot$async
atom$async_is_current_screenname
atom$if_last_return_true_then <1 2>
atom$mat_bool_invisible no
atom$uni_sync_skip 1
atom$mat_bool_invisible yes
atom$uni_sync_skip 2
```

# atom$async_is_guest

## 42 ($2A)

**atom$async_is_guest** verifies whether the sign-on account is a guest account or not.

### Syntax

```
atom$async_is_guest
```

### Return Value

True or false.

### Example

The following example verifies whether the sign-on account is a guest account or not:

```
atom$async_is_guest
atom$if_last_return_true_then <1 2>
atom$man_set_context_relative <30>
atom$mat_bool_disabled <no>
atom$uni_sync_skip <1>
atom$ man_set_context_relative <30>
atom$mat_bool_disabled <yes>
atom$uni_sync_skip 2
atom$man_end_context
atom$man_update_display
```

# atom$async_is_known_subaccount

## 27 ($1B)

**atom$async_is_known_subaccount** verifies if the screen name is a subaccount of the master account.

### Syntax

```
atom$async_is_known_subaccount
```

### Return Value

True or false.

### Example

The following example verifies if the screen name is a subaccount of the master account:

```
atom$async_is_known_subaccount
atom$if_last_return_true_then <1 2>
atom$man_set_context_relative 30
atom$mat_bool_disabled <no>
atom$uni_sync_skip <1>
atom$man_set_context_relative 30
atom$mat_bool_disabled <yes>
atom$uni_sync_skip 2
atom$man_end_context
atom$man_update_display
```

# atom$async_launcher_name

## 39 ($27)

**atom$async_launcher_name** compares the name given to the name when the application was started.

### Syntax

```
atom$async_launcher_name
```

### Return Value

The value returned compares the name given to the name when the application was started.

### Example

The following example compares the ATT name given to the name when the application was started:

```
atom$async_launcher_name <ATT>
atom$asyncif_last_return_true_then 1 2
atom$man_start_object trigger
atom$mat_context_help "Return to ATT&T WorldNet (sm)
Service."
atom$mat_art_id 1-0-29290
atom$act_replace_select_action
atom$uni_start_stream
atom$async_exit
atom$uni_end_stream
atom$man_end_object
atom$uni_sync_skip 1
atom$man_start_object trigger
atom$mat_context_help "Today’s News:\nYour 24-hour new
source."
atom$mat_art_id 1-0-21882
atom$act_replace_select_action
atom$uni_start_stream_wait_on
atom$sm_sendf1 0-32-2573
atom$man_end_object
atom$uni_sync_skip 2
```

# atom$async_logout

## 49 ($31)

**atom$async_logout** invokes the sign-off process and exits the application. See atom$async_signoff on page 10-38 for a related atom that does not exit the application.

### Syntax

```
atom$async_logout
```

### Return Value

Unchanged.

### Example

The following example invokes the sign-off process and exits the application:

```
atom$uni_start_stream
atom$async_logout
atom$uni_end_stream
```

# atom$async_offline

## 5 ($05)

**atom$async_offline** sends a `MSG_NOTIFY_OFFLINE` and a `MSG_CCL_HANG_UP` to all registered tools. Use this atom on the Goodbye form.

### Syntax

```
atom$async_offline
```

### Return Value

Unchanged.

### Example

The following example is used on the Goodbye form to send messages to all registered tools that the client is going offline and communications links are ended:

```
atom$async_offline
```

# atom$async_online

## 4 ($04)

**atom$async_online** sends a `MSG_NOTIFY_ONLINE` to all registered tools. Use this atom on the Welcome screen form.

### Syntax

```
atom$async_online
```

### Return Value

Unchanged.

### Example

The following example is used on the Welcome screen to let all registered tools know that the client is online:

```
async_online
```

# atom$async_playsound

## 11 ($0B)

**atom$async_playsound** plays a digitized sound if the member has a sound card.

When this atom is used, the online service application searches the DOS path for the sound name.

In the Matterhorn client (GM June 11, 1999) and subsequent clients **atom$async_playsound** is upgraded to handle multiple sound requests.

**Note**: **atom$async_playsound_raw** transfers data in form_edit. **atom$async_playsound_raw** is not documented elsewhere because the description and syntax are the same as **atom$async_playsound**.

### Syntax

```
atom$async_playsound <sound_name or anthem>
```

| | |
|---|---|
| `<sound_name or anthem>` | Specifies the sound name string or anthem. |

### Return Value

Unchanged.

### Example

The following example plays the Goodbye sound message:

```
atom$async_playsound <"Goodbye">
```

# atom$async_playsound_dammit

## 11 ($0B)

**atom$async_playsound_dammit** forces a sound to play even if another one is currently playing.

You will encounter this atom in older clients. In the Matterhorn client (GM June 11, 1999) and subsequent clients there is no need to use **atom$async_playsound_dammit** because **atom$async_playsound** is upgraded for multiple sound requests.

### Syntax

```
atom$async_playsound_dammit <sound_name>
```

| | |
|---|---|
| `<sound_name>` | Specifies the sound name string. |

### Return Value

Unchanged.

### Example

The following example forces the You’ve Got Mail sound message to play immediately:

```
atom$async_playsound_dammit <"GotMail">
```

# atom$async_relogon

## 44 ($2C)

**atom$async_relogon** invokes the process of signing on again.

### Syntax

```
atom$async_relogon
```

### Return Value

Unchanged.

### Example

The following message starts the process of signing on again:

```
atom$async_relogon
```

# atom$async_relogon_init

## 43 ($2B)

**atom$async_relogon_init** starts the process of signing on again.

### Syntax

```
atom$async_relogon_init
```

### Return Value

Unchanged.

### Example

The following example starts the process of signing on again:

```
atom$uni_start_stream
atom$uni_invoke_no_context <41-31433>
atom$async_relogon_init
atom$buf_destroy_buffer
atom$buf_start_buffer 159
buf_set_token "UR"
buf_add_atom_data <0>
buf_close_buffer
atom$uni_end_stream
```

# atom$async_screen_name_changed

## 26 ($1A)

**atom$async_screen_name_changed** sets the internal client pointers to the currently selected screen name. This atom alerts all tools to the fact that the current screen name has changed.

### Syntax

```
atom$async_screen_name_changed
```

### Return Value

Unchanged.

### Example

The following example sets the internal client pointers to the currently selected screen name:

```
atom$async_screen_name_changed
```

# atom$async_set_screen_name

## 37 ($25)

**atom$async_set_screen_name** sets the screen name to whatever screen name is then in use.

**Note**: This atom does not work on a branded client because you cannot set the screen name to any name other than the one that logged on to initiate the current session.

### Syntax

```
atom$async_set_screen_name <screen_name>
```

| | |
|---|---|
| `<screen_name>` | Specifies the screen name. |

### Return Value

Unchanged.

### Example

The following example sets the screen name to the one that logged on to initiate the current session:

```
atom$uni_start_stream
atom$async_set_screen_name <"name">
atom$uni_end_stream
```

# atom$async_sign_on

## 30 ($1E)

**atom$async_sign_on** invokes the sign-on process.

### Syntax

```
atom$async_sign_on
```

### Return Value

Unchanged.

### Example

The following example invokes the sign-on process:

```
atom$uni_start_stream
atom$async_sign_on
atom$morg_finish_jump_bookmark
atom$uni_end_stream
```

# atom$async_signoff

## 47 ($2F)

**atom$async_signoff** explicitly invokes the sign-off process without exiting the application. The atom overrides any previous request to exit the application. See atom$async_logout on page 10-28 for a related atom that does exit the application.

### Syntax

```
atom$async_signoff
```

### Return Value

Unchanged.

### Example

The following example invokes the sign-off process without exiting the application:

```
atom$uni_start_stream
atom$async_signoff
atom$uni_end_stream
```

# atom$async_storename

## 45 ($2D)

**atom$async_storename** stores the new screen name.

### Syntax

```
atom$async_storename <name>
```

| | |
|---|---|
| `<name>` | Specifies the screen name string. |

### Return Value

True or false, depending on whether the password is stored.

### Example

The following example stores the new screen name:

```
atom$man_set_context_relative <1>
atom$de_start_extraction <0>
atom$de_get_data_pointer
atom$uni_use_last_atom_string prot$async
atom$async_storename
```

# atom$async_storepassword

## 46 (2E)

**atom$async_storepassword** stores the new password.

### Syntax

```
atom$async_storepassword <password>
```

`<password>` Specifies the password string.

### Return Value

True or false, depending on whether the new password is stored.

### Example

The following example stores the new password:

```
atom$man_set_context_relative <40>
atom$de_start_extraction <0>
atom$de_get_data_pointer
atom$uni_use_last_atom_string prot$async
atom$async_storepassword
```

# atom$async_system_usage

## 52 ($34)

**atom$async_system_usage** reports on the individual client’s system characteristics.

## Syntax

```
atom$async_system_usage <n>
```

| | |
|---|---|
| `n=0` | Windows version (in 32-bit, 95, or NT) |
| `n=1` | Percentage of memory free |
| `n=2` | Percentage of system resources free (16-bit only) |

## Return Value

Unchanged.

## Example

The following example reports on the individual client’s system version of Windows:

```
atom$man_set_context_relative <40>
atom$async_system_usage 0
atom$man_end_context
atom$man_update_display
```