Multi-Target Automation with AWE Server and the MATLAB Scripting API

Hi Monika,

Short answer on architecture

AWE Server is a singleton on the host and maintains one *active* tuning connection at a time. There is no built-in mechanism for the Server to hold simultaneous connections to multiple Ethernet targets from a single MATLAB session. The supported pattern is: switch the Server's active connection → do work on that target → switch to the next.

Importantly, switching the Server's active connection does **not** stop the previously connected target — the BSP/AWE Core on each board runs independently of whether the Server is currently talking to it, so audio on the other boards keeps running uninterrupted. This is by design.

Connection management primitives in the MATLAB Automation API

The relevant commands are:

- `target_change_connection(CONNECTIONSTR)` — switches the transport. Accepts `'native'`, `'rs232'`, `'usb'`, `'ethernet'`, `'spi'`.
- `target_change_connection(CONNECTIONSTR, SYMTAB)` — same, with an optional second argument to *restore* a previously captured symbol table when reattaching to a target that was built via the Server (this lets you resume a session with full symbol resolution intact).
- `[SUCCESS, SYMTAB] = target_change_connection(CONNECTIONSTR)` — returns the current target's symbol table before switching, so you can save it for later restoration.
- `awe_server_command('connect,12001,15001,<TARGET_IP>')` — the lower-level command, used when rotating among several Ethernet targets at distinct IPs. (12001 is the target tuning port, 15001 is the host port used by AWE Server's MATLAB client; both are defaults.)
- `check_change_target()` — refreshes the cached target info in `AWE_INFO.targetInfo` after a switch. Call this after each switch so subsequent `build`/`load` operations see the correct target profile.
- `target_get_info` / `target_get_core_list` — query the currently connected target to verify the switch landed where you expected.
- `target_set_timeout(ms)` — increase from the 3000 ms default if your Ethernet link can be slow. Internally, AWE Server uses 30000 ms during the initial connect, which is a reasonable upper bound for less reliable links.
- `awe_server_command('exit')` and `awe_server_launch('open'/'close'/'kill'/'status')` for clean lifecycle management.

Canonical example for safe target switching

This is the documented round-trip from our engineering notes:

```
target_change_connection('usb');           % switch to USB target
SYS = build(SYS);                          % build a layout on it
```

```
test_start_audio;                          % start it running

% Now switch back to the PC for some offline processing.
% Save the USB target's symbol table on the way out.
[SUCCESS, SYMTAB] = target_change_connection('native');

% ... the USB target keeps running its audio; do PC-side work here ...

% Reattach to the USB target and restore symbols
target_change_connection('usb', SYMTAB);
```

The key guarantee here is in the comment: the USB target keeps running after you switch away. The same holds for Ethernet targets.

Recommended workflow for multiple Ethernet targets

Build a small wrapper that holds a list of target IPs and any cached symbol tables, and rotate through them:

```
targets = struct( ...
    'name', {'A','B','C'}, ...
    'ip',   {'192.168.1.10','192.168.1.11','192.168.1.12'}, ...
    'symtab', {[],[],[]});

target_set_timeout(10000);   % give Ethernet some headroom

for k = 1:numel(targets)
    awe_server_command(sprintf('connect,12001,15001,%s', targets(k).ip));
    check_change_target();   % refresh AWE_INFO.targetInfo for this target

    GSYS = load_awd('layout.awd');
    GSYS.SYS = build(GSYS.SYS);
    % ... tuning / measurement / regression checks ...

    % Capture symbol table before moving on, so we can resume cleanly
    [~, targets(k).symtab] = target_change_connection('ethernet');
end
```

A few practical notes:

- Always set the connection-specific port/IP on AWE Server *via the MATLAB API* (not the GUI) inside an automated run — changes made in the AWE Server GUI's Target → Change Connection dialog are persisted to `AWE_Server.ini` and can fight with scripted state.
- If your link is shared with other traffic, bump `target_set_timeout` rather than retrying — a higher timeout is more stable than retry loops, which can leave the Server in an awkward state.
- After every switch, call `target_get_info` (or `check_change_target`) before issuing build commands. The Server keeps a cached view of the target's block size, sample rate, core count, etc.; that cache is what subsequent build/wire operations consult.
- Wrap each iteration's switch + build in `try/catch` so one unreachable target doesn't abort the whole run. The Server will automatically fall back to the previous proxy if discovery fails, which is usually what

you want, but you'll want to log the failure.

Documentation

All of the documentation referenced below is published at https://dochub.dspconcepts.com:

- **AWE Server theory of operation** — Designer Reference Guide → Using the Server → Theory of Operation. Covers the Designer ↔ Server ↔ Target architecture and the AWS/AWB protocol flow.
- **Changing the target** — Designer Reference Guide → Using the Server → Changing the Target. Documents the GUI equivalent and the connection settings persisted to `AWE_Server.ini`.
- **MATLAB Scripting API** — MatlabAPI → Scripting API. The full Matlab Automation API reference; the "Continuous Integration / Regression Testing" subsection maps directly to your use case.
- **AWE_INFO global** — MatlabAPI → AWEINFO. Documents the global config struct, including the `displayControl` / `buildControl` knobs you'll likely want for automation runs.

Best, Rob