# TTerm Connect Embedded Guide

## What is TTerm Connect?

TTerm Connect is Turbosoft's web-based terminal emulator. Built on common web technologies such as HTML5, Cascading Style Sheets (CSS) and Javascript, TTerm Connect requires no desktop software installation or management and is able to provide terminal emulation services regardless of platform.

TTerm Connect consists of a server component which provides the actual emulation and host connectivity and a lightweight browser-based client which provides the end user interface. Administration and management are also provided through a web based interface.

The server component is available for both Windows® and Linux® based servers, specifically Red Hat Enterprise Linux 7/8 and CentOS 7/8. Web server software explicitly supported includes Apache (Windows and Linux) and IIS (Windows).

The TTerm Connect client is simply any modern HTML5 capable browser with Javascript enabled. The following browsers are explicitly supported.

- Microsoft Edge®
- Mozilla Firefox®
- Apple Safari®
- Google Chrome®

## About this Document

The document applies to TTerm Connect version ***2.1.2.xxxx*** and above.

### Intended Audience

This document describes the process of embedding TTerm Connect within an existing webpage or crafting a custom webpage for host connectivity utilizing TTerm Connect. As such, it assumes an understanding of the technologies involved in web development including HTML5, CSS and Javascript. An understanding of the concepts of terminal emulation is highly desirable.

This document also assumes that TTerm Connect has already been successfully installed on an accessible server. If this is not the case, please refer to the **TTerm Connect Installation Guide** and complete the processes described therein before proceeding.

Access to the Administration Portal, the management web interface for TTerm Connect, is required to configure TTerm Connect for host connectivity and apply appropriate permissions to session profiles.

### Technical Support and Assistance

More information on working with TTerm Connect embedded can be found in the **TTerm Connect Javascript API** document included with your installation. For further assistance please visit ***www.ttwin.com/support*** or contact Turbosoft Support directly. Contact details are listed on the last page of this document.

# Embedding TTerm Connect

There are two web-based elements to TTerm Connect, an *Administration Portal*, where configuration, monitoring and management of host connectivity and end user activity take place as well as a *Desktop Client,* where end users can login and connect to host systems.

The Desktop Client is supplied with the intention of replacing a traditional desktop terminal emulator application. It is an optional component and may be replaced by embedding TTerm Connect in an existing web page (intranet, public facing). This document describes the process of embedding TTerm Connect within an existing webpage or crafting a custom webpage for TTerm Connect.

## Included Example Pages

Included in your TTerm Connect installation are several HTML templates which demonstrate various implementations of embedded TTerm Connect, from examples with a bare minimum of HTML, CSS and Javascript to more advanced pages. These are intended to be used as a guide to assist in incorporating TTerm Connect into your own webpages.

These templates can be found in the `/examples` directory of the web client installation. That is, if TTerm Connect is configured to offer a desktop client on ***https://myserver.com***, the example pages can be found at ***https://myserver.com/examples.***

Please note, the `/examples` directory should be REMOVED before taking an installed TTerm Connect website live. While none of the example pages allow connection to a host system without some minor configuration through the Administration Portal they should not be left online in a production environment.

The first included template is a **fixed size** implementation, a simple page displaying TTerm Connect in a small, fixed size.

The second template is a **full screen** example, expanding TTerm Connect to fill the entirety of the browser window.

The third template displays **multiple host connections on a single page**, illustrating a total of twelve sessions operating independently.

The final template represents an **advanced example**, embedding TTerm Connect in something resembling a real-world webpage. The page adds interactive elements for the end user such as a form which allows the user to select a host, port, communications type and emulation prior to attempting a host connection, along with additional user interface elements such as emulation key buttons and a disconnect button.

The directory web page at ***https://myserver.com/examples/index.html*** will list all these included examples files as clickable links. However, before a host connection is possible each will require some minimal configuration through the Administration Portal. This is detailed on the following pages.

# Fixed Size Example *fixed_size.html*

## What is it?

A minimal webpage containing a single embedded TTerm Connect session designed to automatically connect to a host on page load and display at a fixed height and width of 800 by 600 pixels.

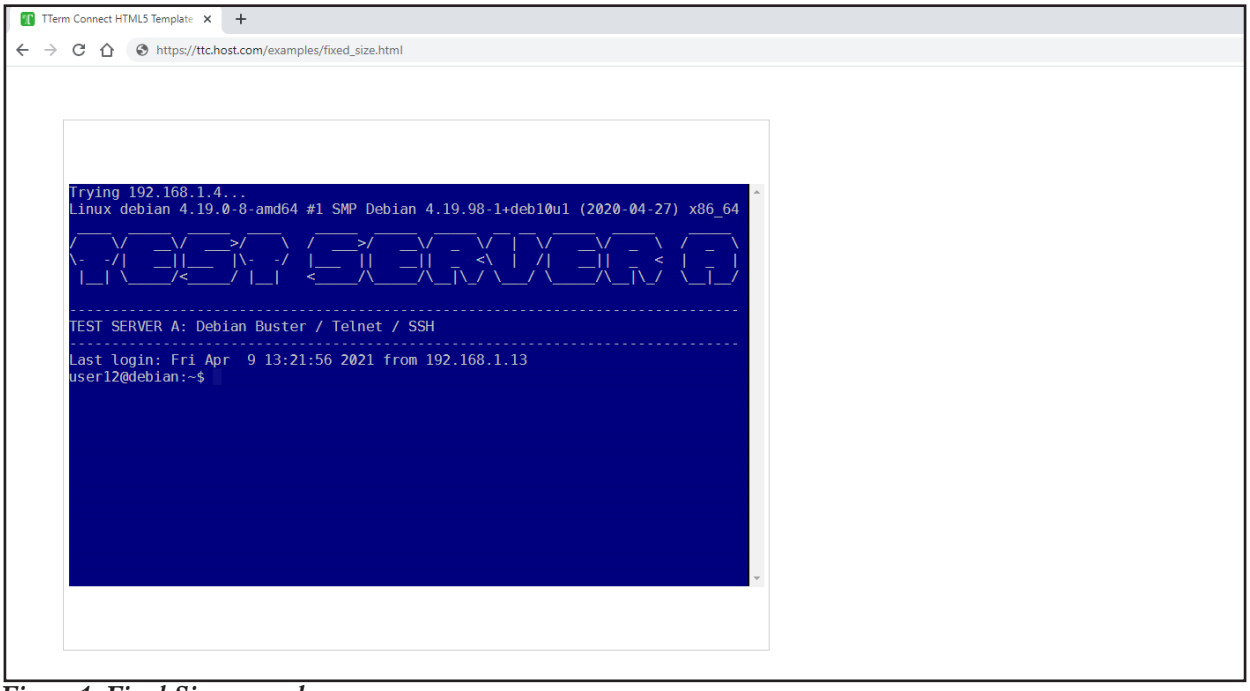


*Figure 1. Fixed Size example page.*

## Server Configuration

To allow this and all other example pages other than the Advanced example (***page 18***) to connect to a host system the following settings must first be configured through the Administration Portal:

- ***TTerm Connect must have* Guest Access *enabled.***

Guest access enables any user to access a TTerm Connect session, eliminating the need to own a TTerm Connect account. Note this DOES NOT mean the user has guest access to your host as that layer of access control remains the domain of your host system.

To set up guest access login to the Administration Portal and navigate to the **Server Configuration** page. **Guest Access** settings are defined under the **Server Components > Connection Manager** section. Set the **Allow Guest Access** option to **on** and specify a **Guest User Group** which all guests will be assigned to.

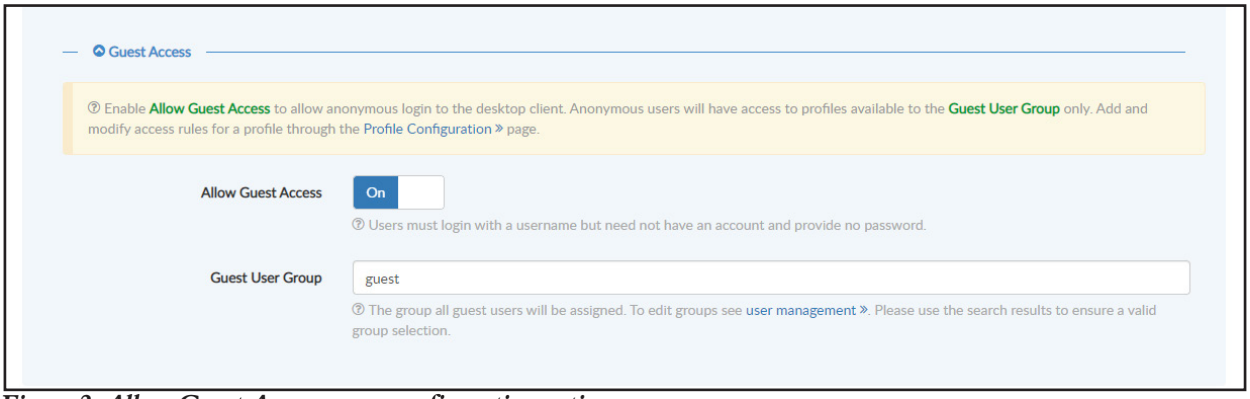


*Figure 2. Allow Guest Access server configuration option.*

On a new TTerm Connect installation, the **Guest User Group** selection drop down will contain the built-in user groups **default** and **guest**, the latter being a group specifically created for this purpose. In practice, any group may be used.

Refer to the ***TTerm Connect Quickstart Guide*** or in-program help for further information on creating additional groups or other aspects of server configuration.

► ***A profile must be created for use with the example page.***

TTerm Connect defines a **profile** as a set of configuration options that is used to spawn a session to a host.

To create a new profile in the Administration Portal navigate to the **Profiles Configuration** page. A **Profiles** table lists all profiles that currently exist on the server. Click the **New** button at the bottom of this table. A window will display requesting some basic parameters for your new profile such as a **Name**. Enter ***'example'*** and click **Create Session Profile**.

Note that several of the other templates look for a profile named ***'example'*** and it is strongly recommended this name be used. An alternate name is acceptable, but the HTML source of various template pages will need to be modified to reference this alternate profile name.

In the **Profile Summary** that subsequently appears edit the **Remote Host**, **Emulation** and **Communications** options to match your host. If you require SSH communications it is recommended, for the purposes of this example page, that you pre-enter a **Username** and **Password** in the **Connection > User Details** section to facilitate automated login.

Considering setting the **General > Session Settings > Start Settings > Auto Connect On Startup** setting to **ON**. This sets the profile to connect immediately and results in a host connection being established to the nominated host system when the web page loads and without any additional interaction from the end user. Alternately set to **OFF** and use initiate a connection using the Javascript API. See page 8 for more information.

► ***Grant the guest group access to your new profile.***

Access control rules grant and limit access to a particular profile based on user and group authorization.

By default, the administrator user account which created the profile, the owner, receives full access rights to the profile. It is also essential to give the **Guest User Group** as previously defined, access to your new profile.

To do this, edit the profile and select the **Access** tab. A table will list the existing access rules for user and group entities for this particular profile. Select the **Add New Rule** button and then select the **Guest Group** button to auto-fill the name of the guest user group as previously defined.

Click **Add Rule** to finalize the new rule and note the new addition to the list of access rules.

Save the changes to the profile.

▸ ***Ensure TTerm Connect is ready to serve end users.***

Finally, navigate to the dashboard of the Administration Portal and confirm that the server is in an active state and ready to serve host connections. At least one Server and one Emulation Service must be active/running.

With these configurations complete the fixed_size.html page will now connect to your nominated host system. Simply click the page link on the examples directory (`/examples/index.html`) to load the page and initiate a host connection.

## Source Code and Detail

To gain a further understanding of how the example page works, let us examine the source code. For brevity, the extensive comments included in the actual HTML file have been removed below.

```
<!doctype html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>TTerm Connect HTML5 Template - fixed size version</title>

    <link href="css/ttermconnect.min.css" rel="stylesheet">
    <script src="../js/ttc.config.js"></script>
    <script src="js/ttermconnect_base.min.js"></script>
    <script src="js/ttermconnect_portable.min.js"></script>

    <style>
        #container {
            width: 800px;
            height: 600px;
            margin: 60px;
            background-color: #FFF;
            border: 1px solid #CCC;
        }
    </style>

    <script>
        const TTermSession = new TTermConnect.Session();
        TTermSession.profile = 'example';

        const isGuestLogin = true;
        const username = "guest@email.com";
        const password = "";
        const task = "new";
        const server = TTermConnect.getBaseServer();

         /***
         * initial function called on page load event.
         ***/
         window.onload = (event) => {
            TTermSession.Service(server, username, password, task, isGuestLogin);
            TTermSession.Initialize(onReady);
         };

        /***
         * Session ready event
         * @param {object} session object
         * @param {bool} success
         * @param {string} msg
         *
         * the msg string format is [error: |warning: ][errorcode: ][string]
         * Refer to the TTerm Connect Javascript API
         * document for a full list of error codes.
         ***/
        const onReady = (session, success, msg) => {
            if (success === false) {
```

```
                alert(msg); //error, exit
                return;
            } else if (success === true && (typeof msg === “string” && msg !== “”)) {
                alert(msg); //message
            }

            session.setAlignment(“CENTER”, “CENTER”, true);
            setSessionEvents(); //attach session event handlers
        };

        /***
        * assign session event handlers
        ***/
        const setSessionEvents = () => {
            TTermSession.TTerm.onCommConnect = onCommConnect;
        };

        /***
         * onCommConnect event handler
         * @param {integer} ConnectState, ccsDisconnected: 0, ccsConnected: 1, ccsConnecting: 2
         ***/
        const onCommConnect = ({ ConnectState }) => {
            switch (ConnectState) {
                case TTerm.connectStates.ccsDisconnected:
                    TTermSession.TTerm.display(false);
                    TTermSession.TTerm.setBackgroundImage(“idle”);
                    break;

                case TTerm.connectStates.ccsConnecting:
                    TTermSession.TTerm.setBackgroundImage(“loading”);
                    break;

                case TTerm.connectStates.ccsConnected:
                    TTermSession.TTerm.display(true);
                    TTermSession.TTerm.setBackgroundImage(“idle”);
                    TTermSession.TTerm.resizeFont();
                    TTermSession.TTerm.focus();
                    break;
            }
        };
    </script>
</head>

<body>
    <div id=”container”>
        <div class=”ttermConnect”>
            <div id=”ttw_termwrapper1”></div>
        </div>
    </div>
</body>

</html>
```

The basic flow of the page is as follows:

❶ A `Session` object is instantiated. This object references a **profile** we have previously created and named '*example*' through the **Administrator Portal**. The profile defines the host specifics such as the remote host address, communications protocol and terminal emulation.

❷ Once the page is fully loaded the native Javascript `window:load` event is used to call `TTermSesssion.Service` to initiate login to an **Emulation Service** as a guest user.

❸ The session is then initialized, with the results passed to a callback function. If successful, the page defines several session behaviors and cosmetic settings and sets the terminal objects alignment within its parent container.

❹ The session is the processed as defined by settings in the '*example*' profile. If **Auto Connect On Startup** was set the session will attempt to connect to the remote host system immediately. The `onCommConnect` event handler actions any change in connection state.

Simply put, the end user should load the page and be greeted with a blinking cursor and host system prompt.

In TTerm Connect version 2.1.0 and above it is no longer necessary to explicitly enable the `onCommConnect` event through `putOnCommConnect()`. All changes in connection state are sent by the server to the client by default and it is only necessary to set a handler function to the session.`onCommConnect` property.

## When to use Auto Connect

TTerm Connect profiles can be configured to automatically attempt a connection to the host system when started. This setting can be particularly useful when the profile is also preconfigured with host, port and login credentials as it enables automatic connection and login to a host without any further input from the end user.

This setting can be found in the Administrator Portal profile editor under **General > Session Settings > Start Settings > Auto Connect on Startup** and is set to ON by default.

If your embedded requirements involve reading and manipulating the profile settings through the Javascript API before connection or adding additional settings (such as registering `watchScreenStrings`) then it essential that **Auto Connect on Startup** is be disabled. In such cases a connection should be initiated programmatically using the `apiCommConnect` method.

All Javascript API methods, excluding those belonging to the **Local** group, involve server communication and are asynchronous. Refer to the **TTerm Connect Javascript API** document for further information.

## Troubleshooting

If an initial connection is not successful one of the following error messages may be displayed:

- '**error: 70**' Profile not found. Confirm that the profile property set in Javascript matches a valid profile name (check spelling, for example) and that access rules are correct.
- '**error: 380**' This error indicates that the current user or session lacks the appropriate permissions to perform a particular operation. Certain TTerm Connect API methods such as `apiPropGetValue` and `apiPropSetValue` allow the manipulation of profile settings at runtime in the client. For security reasons these are disabled by default in the profile configuration. To resolve this error edit the individual profile in the Administrator Portal and confirm settings under **General > API Settings** are enabled if required
- '**error: 401**' This indicates an unauthorized request and suggests that guest access is not enabled on the server. Guest usernames may be rejected if they do not meet the minimum length settings or the requirement that guest usernames are emails as set in the TTerm Connection configuration. Check the Administrator Portal's Server Configuration page for guest username settings.
- '**error: 950**' or '**error: 960**' This indicates a failure to connect to an emulation service. Possible explanations for this are an incorrect server address when calling `TTermSession.Service` or emulation services have been stopped. Check the Administrator Portal and confirm the Server and Services are online.
- '**error: 970**' This indicates a failure to connect to the Connection Manager component of TTerm Connect. Check for issues with network connectivity and ensure the server is running. In a multi-server environment configured for load-balancing this message will only be seen if none of the servers in the pool responds.
- '**error: 980**' This indicates the HTML elements TTerm Connect needs to attach a terminal to are not present in the webpage. Check the section on "Required HTML" on page 12.
- '**error: 988**' This indicates a required resource, for example a font file, could not be loaded. This is indicative of a connectivity issue with the (Apache/IIS) webserver hosting the TTerm Connect webpage rather than a TTerm Connect server or a missing file on the webserver.
- '**error: 990**' or '**error: 992**' or '**error: 993**' This indicates the TTerm Connect server failed to respond to a request within an allotted time span. Possibly explanations include a network outage or server-side issue.

  These errors differ only in their origin. Error 990 will be seen when attempting to authenticate with TTerm Connect and establishing an initial server connection. Errors 992 and 993 are displayed when launching a host session. Note that these **does not** refer to a communications error between the webclient and the remote host, instead it refers to a communications error between the webclient and the TTerm Connect server.

Additionally, when using SSH check any host username and password details in the profile configuration are correct. Check also that the profile is correctly configured according to the host system requirements.

If a host connection is established but ends unexpectedly a user may see one of the following messages:

- '**error: 972**' This indicates the connection to the Connection Manager component of TTerm Connect has been unexpectedly lost. Check for issues with network connectivity and ensure the server is running.
- '**notice: 991**' This message is displayed when an administrator remotely closes (kills) a session through the Administrator Portal.
- '**error: 1006'** This message indicates the websocket for an individual session has unexpectedly closed. Refer to the Advanced Example (page 18) source code for an example of a custom handler for this error.

For detailed error code information refer to **Appendix D** of the **TTerm Connect Javascript API** document.

## Handling Session Events

Let us examine the sections of the template file dealing with the events the Session object triggers and how they are handled. Locate the following sections of Javascript which are executed as part of the `window.onload` event.

Here, the `setSessionEvents` function is called when the `TTermSession` object ready event is fired. Its task is to assign handlers to session events.

```
const setSessionEvents = () => {
     TTermSession.TTerm.onCommConnect = onCommConnect;
};
```

The onCommConnect event will fire in response to a change in the session connection state. `TTermSession.TTerm.onCommConnect = onCommConnect;` assigns the `onCommConnect` function as a handler to this event.

```
/***
* onCommConnect event handler
* @param {integer} connectState, ccsDisconnected: 0, ccsConnected: 1, ccsConnecting: 2
***/
const onCommConnect = ({ ConnectState }) => {

    switch (ConnectState) {
        case TTerm.connectStates.ccsDisconnected:
             TTermSession.TTerm.display(false);
             TTermSession.TTerm.setBackgroundImage("idle");
             break;

        case TTerm.connectStates.ccsConnecting:
             TTermSession.TTerm.setBackgroundImage("loading");

        case TTerm.connectStates.ccsConnected:
             TTermSession.TTerm.display(true);
             TTermSession.TTerm.setBackgroundImage("idle");
             TTermSession.TTerm.resizeFont();
             TTermSession.TTerm.focus();
             break;
    }
};
```

The event object contains an integer `ConnectState` property which is compared to values in the `TTerm.connectStates` object to determine the state.

**On disconnect:** The terminal display is hidden from view. The 'idle' image, which displays in its place, is set to the default TTerm Connect logo.

**On connecting:** The background image is set to the built-in default image, an animated, spinning icon.

**On connection:** The terminal display is made visible, and the idle image (now hidden) is reset to the default TTerm Connect logo. The terminal font is resized to best-fit, and focus is set on the terminal page element, enabling keyboard events to be captured by the terminal and the user to begin interacting with the host system.

The methods used are as follows:

### *Session*.TTerm.setBackgroundImage (imagePath)

This method sets the background image for the TTerm Connect session object. It provides a simple option to provide user feedback and to brand the session on your webpage and takes a single parameter. Images are automatically centered on the terminal display. Custom image sizes are recommended to be a minimum of 320 x 240 pixels.

`imagePath`, *string*, a relative path to an image file or one of several references to built-in images. Built in options are as follows:

'`idle`' shows a TTerm Connect logo.

'`loading`' shows an animated spinning icon.

### *Session*.TTerm.display (show)

This method shows or hides the terminal element.

`show`, *boolean,* if true shows the terminal display. Note that a visible terminal display will overlay the loading image set through **setBackgroundImage**, effectively hiding it.

### *Session*.TTerm.resizeFont ()

This method updates the terminal font size to ensure the terminal object fills the entirety of its parent container.

Note that in other example pages, such as the full screen variant, **resizeFont()** is attached to the native `window.onresize` event. This enables the terminal object to update to fill the browser display when the user manipulates the browser window itself.

### *Session*.TTerm.focus()

`focus` places the cursor/input focus on the terminal element, allowing the user to begin typing and interacting with the connected session. This method takes no parameters.

## Debugging

In the callback for the session initialization shown below note that the `TTerm.Logging` variable is a boolean set to `true`. This setting instructs the core TTerm Connect libraries to output various operational information to your browser's developer/debugger console.

This may prove useful in debugging your own custom templates during the development process; however it is advised to remove this line before publishing your webpage. `TTerm.Logging` is set to `false` unless overridden.

```
const OnReady = (session, success, msg) => {
	if (success === false) {
		alert(msg); //error, exit
		return;
	} else if (success === true && (typeof msg === "string" && msg !== "")) {
		alert(msg); //message
	}

	TTerm.Logging = true;
```

## Required Libraries

The linked files in the `<head>` tag contain essential libraries for working with TTerm Connect. When creating your own webpage ensure the following files are available and note relative paths as per `/examples`.

- `/js/ttermconnect_portable.min.js` - contains required TTerm Connect libraries in a single Javascript file. This is a special library that is not included in the Desktop Client and is designed specifically for embedded TTerm Connect.
- `/js/ttermconnect_base.min.js` - common core TTerm Connect libraries.
- `../js/ttc.config.js` - a generated file which contains information about the server pool. Should the pool configuration change this file will be automatically regenerated by TTerm Connect. In the template this file is referenced in the location where it is auto generated. If an alternate location is used in a custom web-page, then the generated file will need to be manually updated should pool configuration change.
- `/css/ttermconnect.min.css` - contains TTerm Connect style sheets.
- `/fonts/TurbosoftSansMono.ttf` - special terminal font, referenced by `ttermconnect.min.css`.

Files with a `.min` extension are minified, meaning they have been compressed to reduce download times in a production environment. It is not recommended that you alter the TTerm Connect source libraries.

## Required HTML

TTerm Connect requires some minimal mark-up to attach a terminal object to your webpage. Regardless of your implementation it is essential that the following appear somewhere in the `<body>` of your HTML document.

```
<div class="ttermConnect">
	<div id="ttw_termwrapper1"></div>
</div>
```

# Full Screen Example *full_screen.html*

## What is it?

A minimal webpage containing a single embedded TTerm Connect session aligned to the center of the page and filling as much of the available space as possible. Designed to automatically connect to a host on page load.

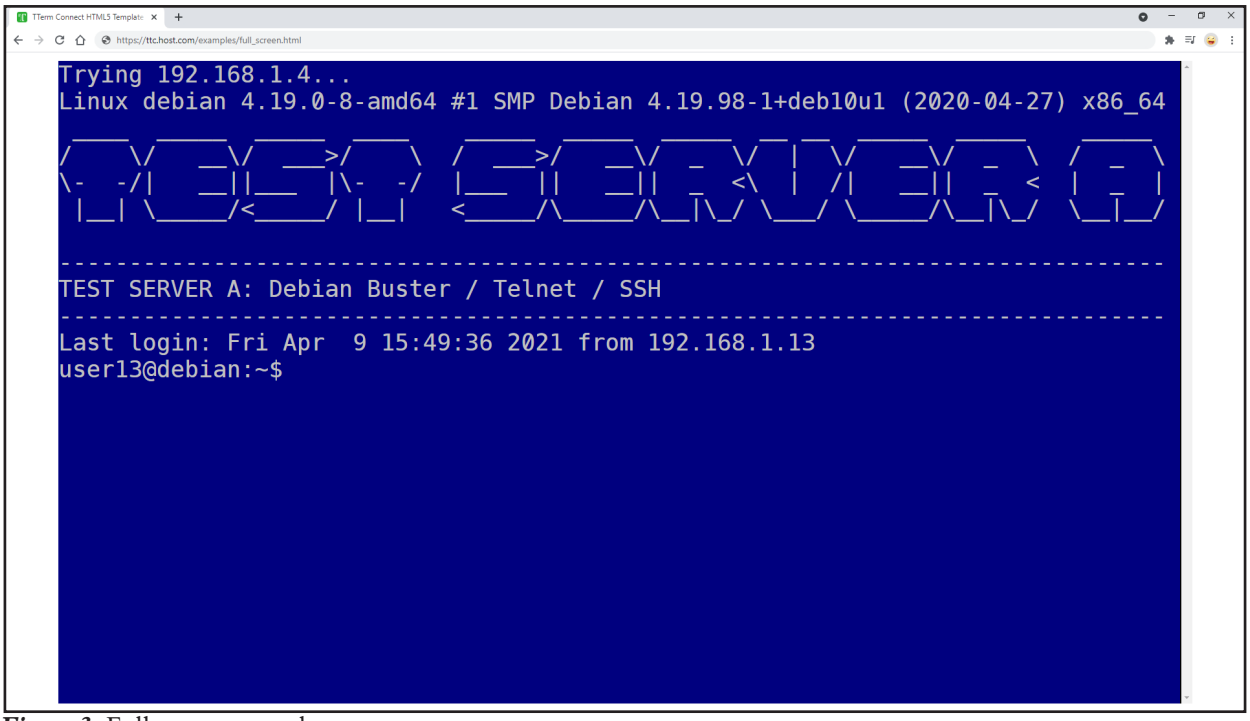


***Figure 3.*** Fullscreen example page.

## Server Configuration

The full screen example uses the same profile and server configurations as the fixed sized template. If these have been implemented the page will connect to a host system without further alterations.

Refer to the source code comments for further details.

# Tiled Example *multi_tiled.html*

## What is it?

This template demonstrates multiple independent terminals on a single webpage. Unlike the fixed size and full screen examples this example connects to twelve sessions simultaneously and makes uses of a '***session manager***' to achieve this. A '***session manager***' coordinates requests from the individual host session objects in the page to the TTerm Connect server components and handles the creation and closing of a host session through a dedicated websocket.

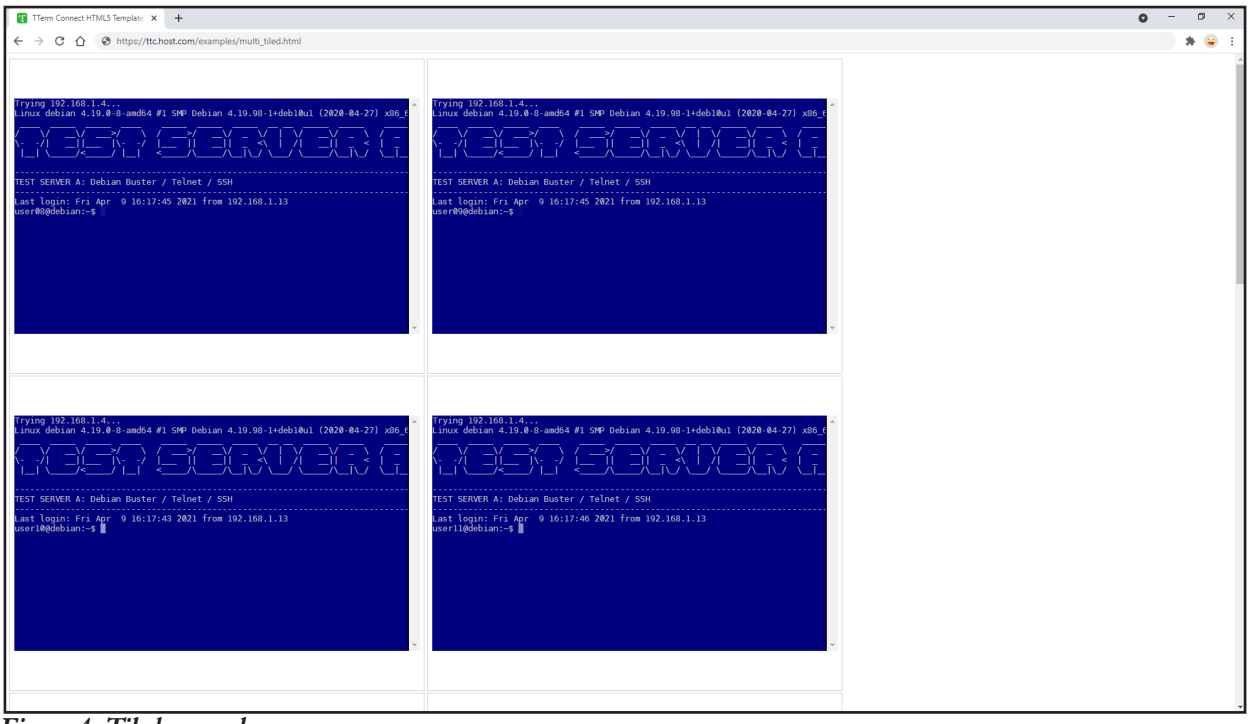


*Figure 4. Tiled example.*

## Server Configuration

The tiled template uses the same profile and server configurations as previous example pages. If these have been implemented as described this page will connect to a host system without further alterations.

Refer to the source code comments for further details describing how to implement this type of TTerm Connect embedded solution.

# Reconnect Example *reconnect.html*

## What is it?

This template demonstrates user authentication and the resuming of a preexisting user session.

It requires user login with a TTerm Connect user account and once authenticated, will reconnect to a previously open session belonging to that user. If no previous session is available, it will instead open a new session.

Sessions that were once open on a webpage but are no longer are considered ***detached***. A ***detached*** session is retained by the TTerm Connect server for a period of time determined by the ***Session Persistence*** settings found under ***Server Configuration*** in the Admin Portal.

Retention times are configurable and can be anywhere between five minutes and seven days, after which a detached session is automatically deleted. You will need to configure the ***Session Persistence*** settings before running this template. Note that, whilst it is possible for the server to retain multiple detached sessions for any one user, this sample template has been designed to reconnect with the first viable session it finds. A more realistic implementation might present the user with a list of sessions to choose from.

Like the previous tiled example this template makes use of a ***session manager***. A ***session manager*** coordinates requests to the TTerm Connect server components through a dedicated websocket.

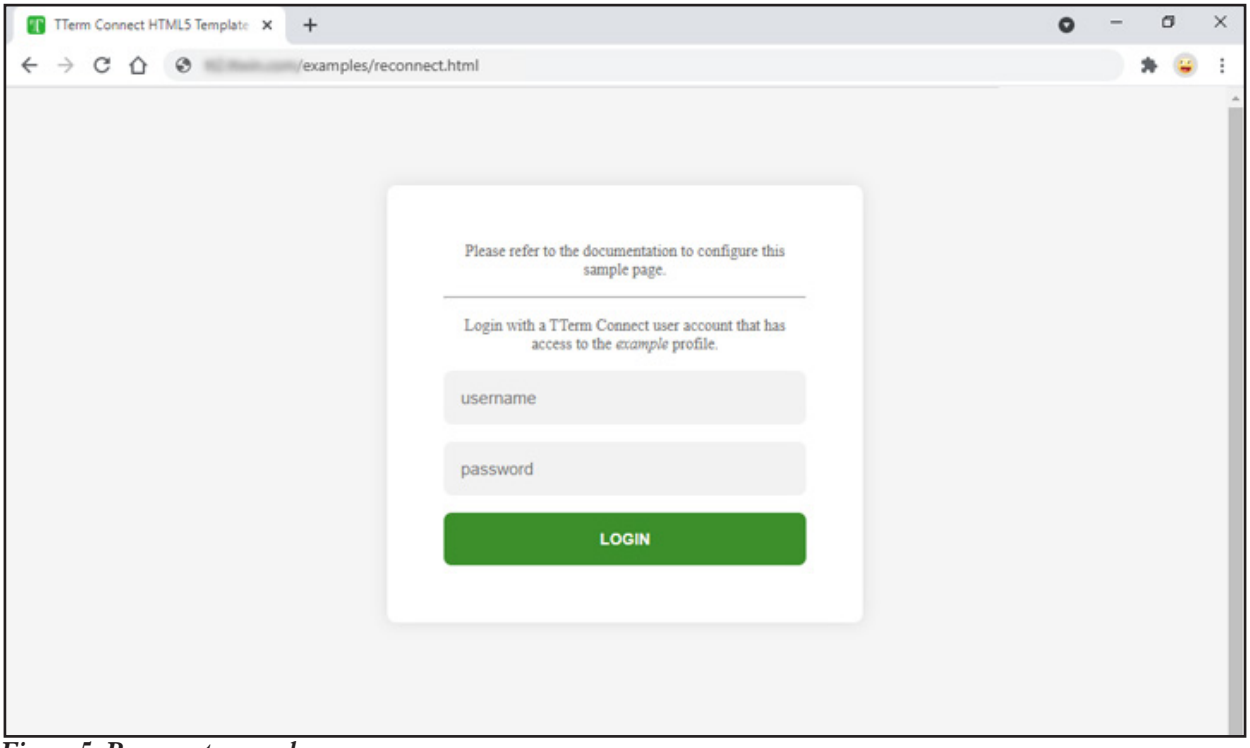


***Figure 5. Reconnect example.***

## Server Configuration

The template is hardcoded to use a profile named '***example***'. This profile must exist, must auto-connect to the host and must have access rules for any user account that attempts to login through the template. Note also that this template is designed ONLY to work with local user accounts.

Refer to the source code comments for further details describing how to implement this type of TTerm Connect embedded solution.

## WatchScreenStrings Example *watchstring.html*

### What is it?

This template displays a single terminal at a fixed size and demonstrates the watchScreenString APIs.

This collection of API methods is used to monitor the terminal display for nominated text. When this text is detected, an event is fired which can be used to run custom Javascript, creating opportunities for enhanced user feedback and customization based on the content of your host application screen.

At the bottom of this webpage is a text box where updates are displayed as the sample script runs through the process of registering strings to watch and then detecting them.

The watchScreenString portion of the script has an arbitrary delay of three seconds after successful host connection. It is configured to search the entire terminal display for a string. The search is case insensitive and non-repeating, meaning that after a single match the search will be terminated. The default search string is '*test*'. This and the other parameters of the search can be modified by examining the template's Javascript.

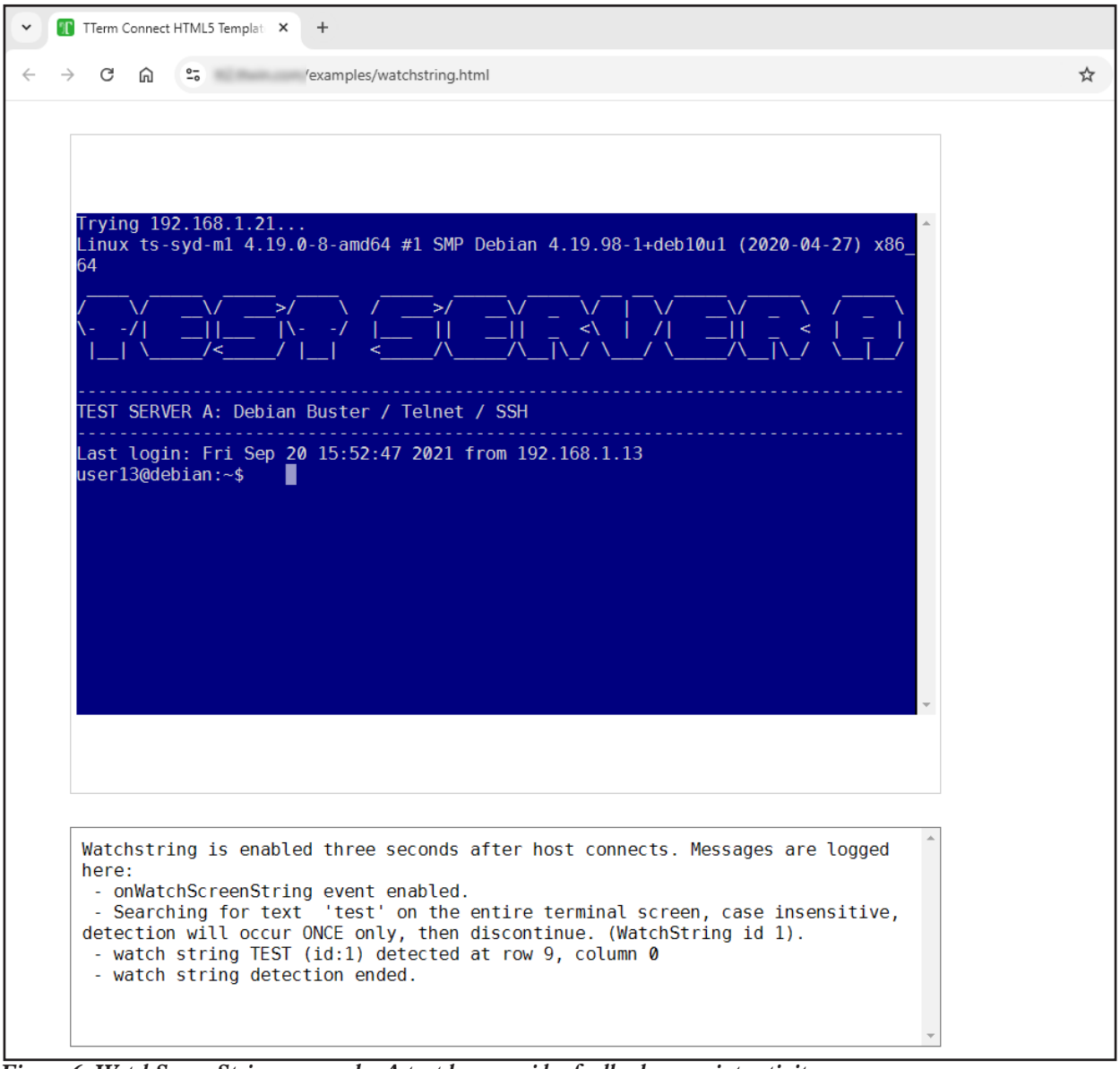


*Figure 6. WatchScreenStrings example. A text box provides feedback on script activity.*

## Server Configuration

The template is hardcoded to use a profile named '***example***'. This profile must exist, must auto-connect to the host and must be accessible to guest user accounts. The server must also have been configured to enable guest user access.

The template uses the same profile and server configurations as several previous example pages (***fixed, fullscreen, tiled***). If these have been implemented as described this page will connect to a host system without further alterations.

Refer to the source code comments for further details describing how to implement this type of TTerm Connect embedded solution.

## Advanced Example *advanced.html*

### What is it?

This template displays a single terminal within a web page which resembles a real-world intranet or public web-site. It demonstrates several more advanced TTerm Connect features for embedding terminal emulation in a web page. These include:

- Allowing the end user to select an emulation (limited to VT100, IBM 3270 and IBM 5250 in this example), communications protocol (limited to SSH or Telnet in this example), host and port prior to initiating a host connection. In previous examples these were fixed through the profile configuration but here they are altered at runtime via the Javascript API.
- Several page elements which send emulation keystrokes to the host system using TTerm Connect macros and Javascript. When using a VT100 emulation these are arrow keys, when using an IBM 5250 or 3270 emulation these are PF1 through PF4 keys.
- Connect and disconnect button elements.
- Printing the terminal screen or the user selection to a PDF file.
- Custom branding of the terminals disconnected (idle) screen and connecting (loading) screens. Refer to "*How can I replace the terminal images with my own image or logo?*" on page 24 for more details.
- Overriding TTerm Connect's built in login modal with a custom version (when connecting to an SSH host.).

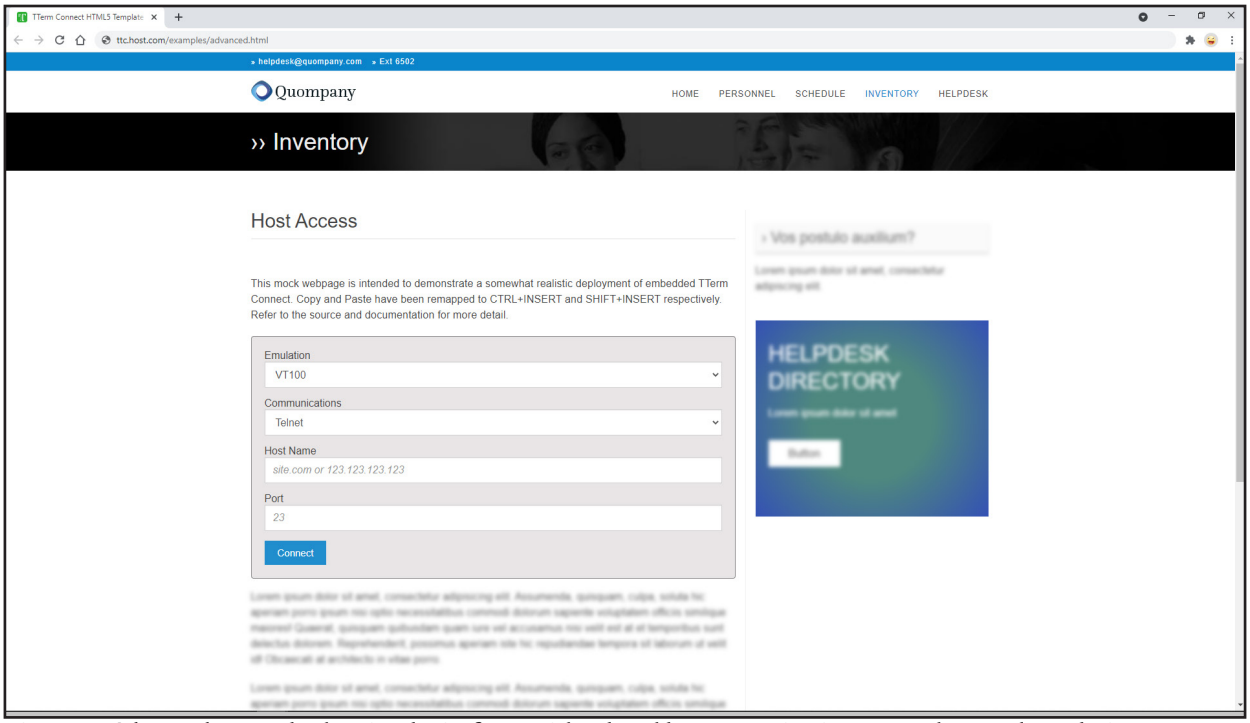


*Figure 7. Advanced example showing login form with selectable communications, port, host and emulation.*

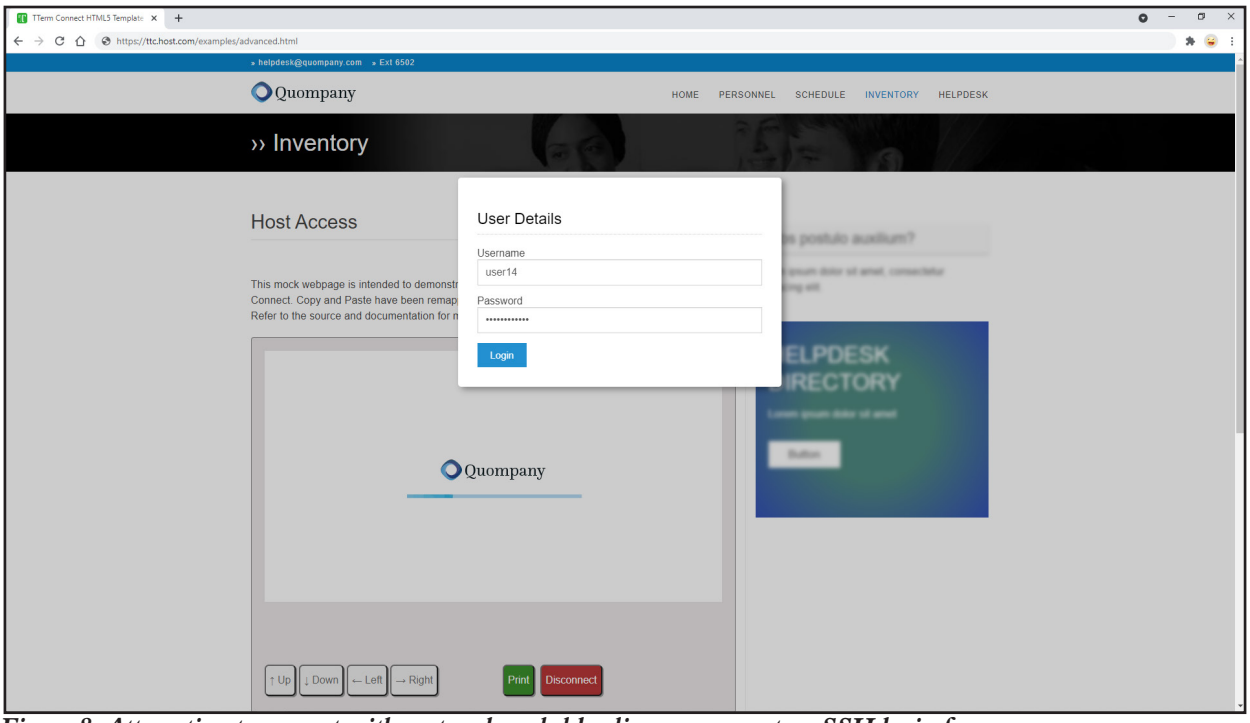


*Figure 8. Attempting to connect with custom branded loading screen, custom SSH login form.*

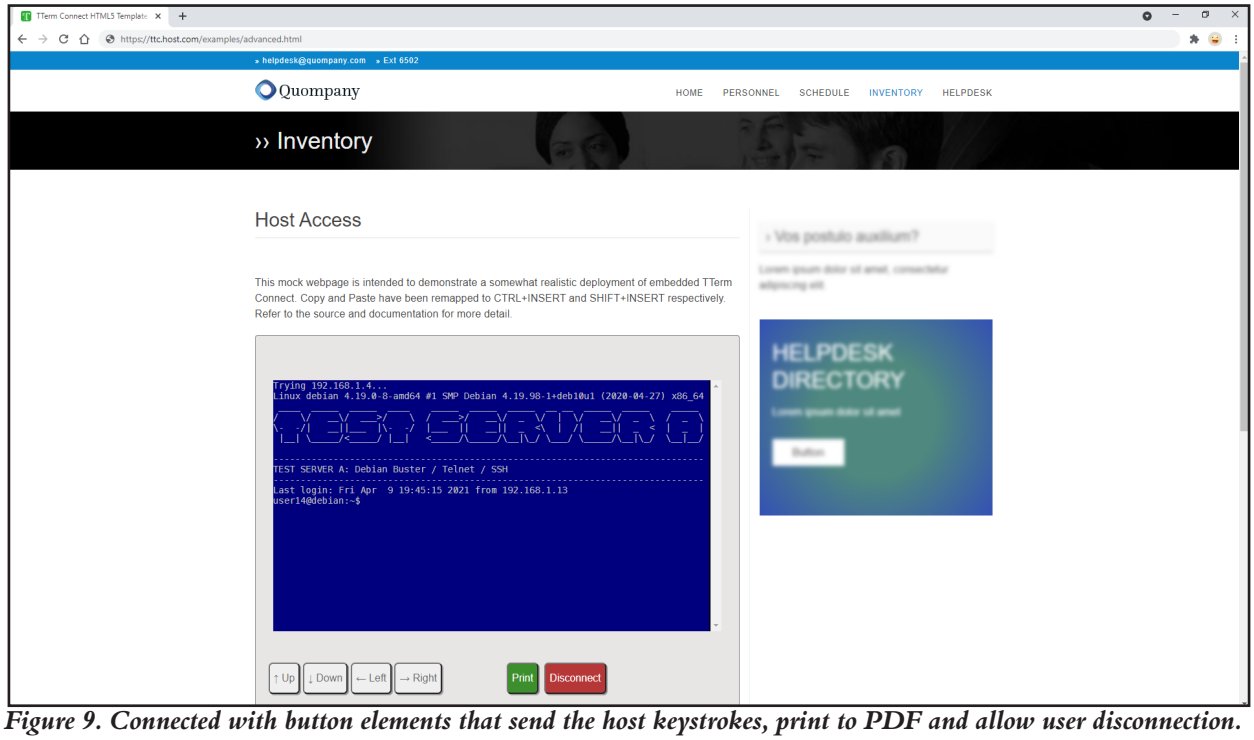


*Figure 9. Connected with button elements that send the host keystrokes, print to PDF and allow user disconnection.*

## Server Configuration

Follow the configuration instructions for the fixed size template (***page 3***) excluding those instructions for profile creation. The Advanced example page requires a different set of profile options.

▸ ***Create a profile for use with the Advanced example page.***

To create a new profile in the Administration Portal navigate to the **Profiles Configuration** page. A **Profiles** table lists all profiles that currently exist on the server. Click the **New** button at the bottom of this table. A window will display requesting some basic parameters for your new profile such as a **Name**. Enter ***'advanced-example'*** and click **Create Session Profile**. An alternate name is acceptable, but the html file content will need to be edited to match this profile name.

Leave **Emulation** and **Communications** settings as default as these will determined by the end user through the webpage form. Similarly, leave the **Remote Host** settings blank.

Under **General > Session Settings** ensure 98that the **Auto Connect On Startup** option is set to **OFF**. This is essential as the end user is required to fill in host details prior to initiating a connection through the webpage form submission.

The Advanced example page contains an HTML element which is used as a disconnect button. TTerm Connect has a configuration option which is designed to enforce clean host disconnections by requiring an end user to first logout from the host. For the disconnection button on the Advanced example page to function this MUST NOT be enabled. Under **General > Session Settings**, ensure **Allow Manual Disconnect** is set to **ON**.

▸ ***Enable the API required to change settings for a profile***

Lastly, the Advanced example pages manipulates several of the profiles configuration settings to allow connection to different hosts with different characteristics, for example swapping between telnet and SSH, between VT100 and TN3270. It does this through the Javascript API method `apiPropSetValue`. By default this method is disabled and, if called in a disabled state, will respond with an error code.

It must be enabled for each profile where `apiPropSetValue` is required through the Administration Portal **Profiles Configuration** page .

To do this, under **General > API Settings**, set the **Enable apiPropSetValue** item to **ON**.

The Advanced example page uses `apiPropSetValue` to manipulate the settings of a single profile. An alternate approach would be to use multiple profiles, one for each host type required and access them as needed through the API. Refer to the **TTerm Connect Javascript API** document for more detail.

► ***Grant the guest group access to your new profile.***

Access control rules grant and limit access to a particular profile based on user and group authorization.

By default, the administrator user account which created the profile, the owner, receives full access rights to the profile. We will also need to give the **Guest User Group** as previously defined, access to your new profile.

To do this, while editing the profile and select the **Access** tab. A table will list all the user and group entities which currently have access to this profile. Click or tap the **Add New Rule** button and click or tap the **Guest Group** button to auto-fill the name of the guest user group you previously set.

Click **Add Rule** to finalize the new rule and note the new addition to the list of access rules.

Save the profile.

The Advanced template page will now be capable of connecting to a host. Visit the ***/examples/index.html*** page and select it from the list of examples to view the page.

Examine the template source code comments for further detail on implementing this type of TTerm Connect embedded solution.

# Deploying

If you have been experimenting with the supplied example templates or building a project from within the `/examples` directory, recall that this directory should be removed in a production environment.

The recommended location for deploying your embedded TTerm Connect HTML file is the `/custom` directory of your TTerm Connect installation. This directory is created automatically at install time and is unique in that when TTerm Connect is upgraded its contents will not be altered.

All locations other than `/custom` will be overwritten when updating, meaning your files will be lost if deployed in the `/examples` directory.

As noted in "**Required Libraries**" on page 12, there are several Javascript and CSS files that an embedded TTerm Connect webpage must reference in order to function. These files exist in the `/examples/js` and `examples/css` directories but are also included in the root directory of your TTerm Connect installation.

Copy your embedded html file to the `/custom` directory and update the references in your HTML as follows:

- `/js/ttermconnect_portable.min.js` ➞ `../js/ttermconnect_portable.min.js`
- `/js/ttermconnect_base.min.js` ➞ `../js/ttermconnect_base.min.js`
- `../js/ttc.config.js` ➞ no changes are required
- `/css/ttermconnect.min.css` ➞ `../css/ttermconnect.min.css`

Using the `/custom` directory gives your end users a final URL which will look similar to ***https://myserver.com/custom/yourpage.html***

## TTerm Connect Upgrade Policy

From time-to-time Turbosoft will release updates to TTerm Connect. This may include updates to the core libraries that your custom webpages reference.

We take every care to maintain compatibility for minor version updates (for example *version 2.0-2.xx*) but may introduce new features and API incompatibilities with major version releases (*versions 2.xx and 3.xx*).

In such circumstances we will provide documentation outlining changes and migration paths to ensure your embedded pages can be updated and continue to function.

## Caching

Both the Desktop Client and embedded TTerm Connect webpages are served through third party webserver software such as Apache or IIS. In this instance it is important that cache settings are configured appropriately through your webserver to ensure that when TTerm Connect is upgraded your users always receive updated content without any extra actions required on their part.

For further information refer to your webserver documentation or consult the **TTerm Connect Installation Guide** document for configuration examples.

# Other Functionality

## How can I trigger a connection via a button or other user interface element?

Once a profile is configured with the appropriate host, communications and emulation settings call the following to attempt a connection:

```
TTermSession.TTerm.apiCommConnect();
```

Be aware that, as shown in the included examples, a profile can be configured to auto-connect on initializing. If the intention is to have an end user initiate a connection via a button click or similar, ensure that the **General > Session Settings > Start Settings > Auto Connect On Startup** setting is set to **OFF** in the profile configuration.

A disconnect can be initiated via:

```
TTermSession.TTerm.apiCommDisconnect();
```

Once again there are profile configuration settings which are designed to enforce clean disconnections (i.e., a user host logout rather than a dropped connection). Ensure that **General > Session Settings > Disconnect Settings > Allow Manual Disconnect** is set to **ON** in the profile configuration, otherwise the `apiCommDisconnect` function will not be permitted to disconnect from the host.

## How can I supply SSH usernames and passwords before connecting?

To programmatically supply login details the following properties should be set before calling `apiCommConnect`.

```
TTermSession.TTerm.apiPropSetValue("com.Password", string);
TTermSession.TTerm.apiPropSetValue("com.UserName", string);
```

Similarly, an SSH port may be specified via:

```
TTermSession.TTerm.apiPropSetValue("com.Port", integer);
```

For a full range of other profile configuration properties which can be set via the `apiPropSetValue` method, refer to the ***TTerm Connect Javascript API*** document.

## How can I replace the terminal images with my own image or logo?

There are two standard images which appear in place of terminal content in TTerm Connect webpage elements. They are the '***idle***' image and the '***loading***' image.

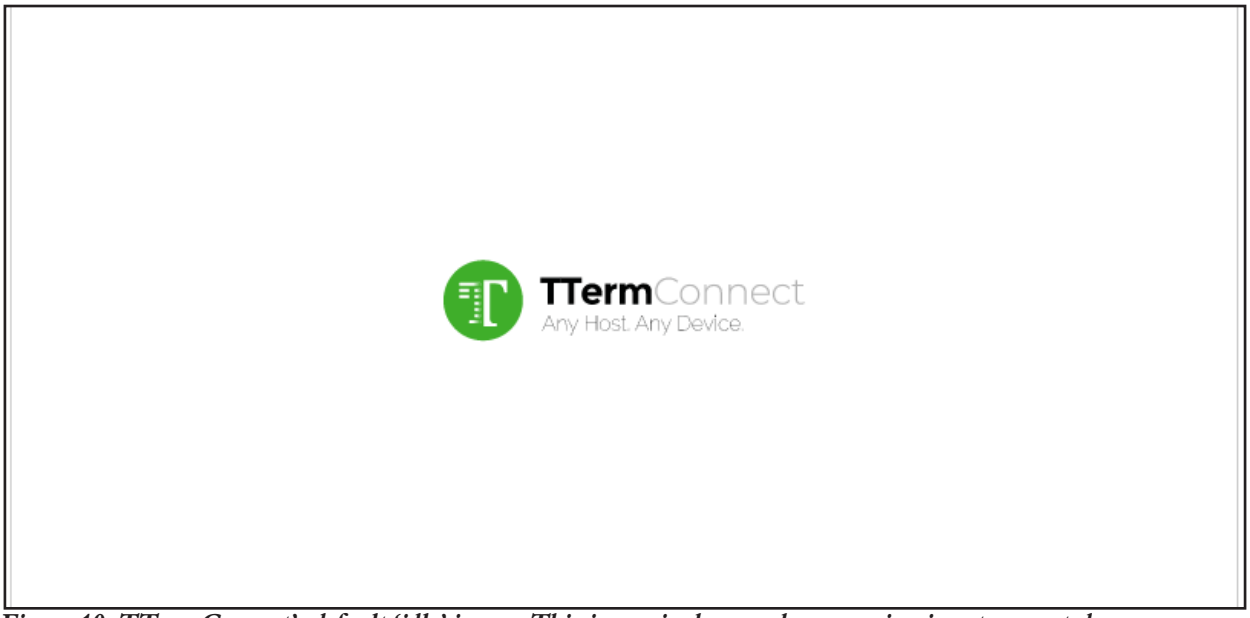


***Figure 10. TTerm Connect's default 'idle' image. This image is shown when a session is not connected.***

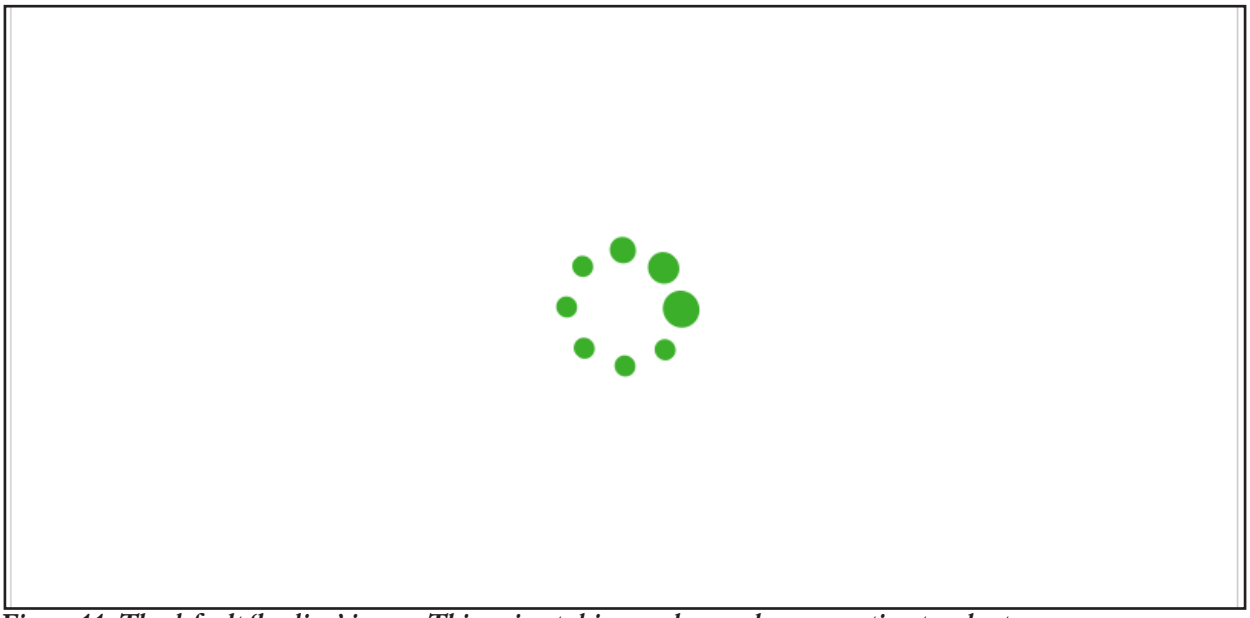

***Figure 11. The default 'loading' image. This animated image shows when connecting to a host.***

The `setBackgroundImage` method allows an embedded TTerm Connect session to select or replace these images on demand. Pass the strings `"idle"` or `"loading"` to utilize the built-in default images or pass a string with a relative path to a custom image.

```
TTermSession.TTerm.setBackgroundImage("idle");
TTermSession.TTerm.setBackgroundImage("loading");
TTermSession.TTerm.setBackgroundImage("../images/my_logo.png");
```

The `setBackgroundImage` method may be combined with the `onCommConnect` event to set images in response to changes in the connection state of the host session. The following is a truncated extract from the **advanced example** (***page 18***).

```
/***
 * onCommConnect event handler
 * @param {object} e, event object
 ***/
 const onCommConnect = ({ ConnectState }) => {
     switch (ConnectState) {
        case TTerm.connectStates.ccsDisconnected:
          TTermSession.TTerm.setBackgroundImage("idle");
          break;

        case TTerm.connectStates.ccsConnecting:
          TTermSession.TTerm.setBackgroundImage("loading");
          break;
     }
};
```

To suggested method to completely replace these images is to redefine the '***idle***' and '***loading***' images. This can be done after instantiating the session object and setting the properties below with relative paths to your custom images. An example of this functionality is provided in the **advanced example** (***page 18***).

```
TTermSession.backgroundImage = "../images/logo.png"; //set the default idle image
TTermSession.loadingImage  = ".../images/loading.gif"; //set the default loading image
```

An example of a custom loading image can be seen in ***Figure 7***.

## How can I print the terminal display?

API methods in TTerm Connect allow the printing of terminal content, either a user's selected text or the current display screen in its entirety.

TTerm Connect printing is an asynchronous operation and output is not sent to an attached printer, instead a PDF file is made available to the end user's device. A print request must first be sent through the `apiPrintScreen` method and, when the server is ready, an `onFileReady` event is then fired.

The Advanced Example (page 18) contains a button element which, when clicked, creates a PDF file of the current terminal display and downloads it to the user's device. View the page source for a working implementation.

First, let us examine the print request:

```
let printMode = (TTermSession.TTerm.hasActiveSelection()) ? TTerm.printMode.selected : TTerm.print-
Mode.fullscreen;
let paperOrientation = TTermSession.TTerm.getPaperOrientation();
let paperSize = TTermSession.TTerm.getPaperSize();

TTermSession.TTerm.apiPrintScreen(printMode, paperSize, paperOrientation, ({ data }) => {
    if (data.result !== 0) {
        //non-zero return value indicates an error.
    }
});
```

`apiPrintScreen` takes four parameters.

`printMode` {*integer*}, sets printing to the entire terminal display or only the user selection. Use the constants `TTerm.printMode.selected` and `TTerm.printMode.fullscreen` to set. When set to `TTerm.printMode.selected` attempting to print where no terminal content is selected returns an error (550).

`paperSize` {*string*}, TTerm Connect supports the following paper sizes:

- Letter
- Legal
- A3
- A4
- A5
- B4
- B5
- Executive
- US4x6
- US4x8
- US5x7
- COMM10

Each profile configuration contains a paper size setting which can be retrieved with `session.TTerm.getPaperSize()`. Alternately a value may be supplied directly by passing the paper sizes listed above as a string.

`paperOrientation` {*string*}, determines landscape or portrait output. Each profile configuration contains a paper orientation setting which can be retrieved using `session.TTerm.getPaperOrientation()`. Alternately a value may be supplied directly by passing the strings **`Landscape`** or **`Portrait`**.

`callback` {*function*} Returned data is supplied in the standard TTerm Connect format where a non-zero value is an error. Refer to the ***TTerm Connect Javascript API*** document for more detail.

The file itself is supplied asynchronously through an `onFileAvailable` event. To be captured in the client this event must be enabled and a handler assigned:

```
TTermSession.TTerm.putEnableOnFileAvailable(true);
TTermSession.TTerm.onFileAvailable = onFileAvailable;
```

In addition to printing the `onFileAvailable` event is used for several other purposes. The `Type` property will contain the string **printer:pdf** when the available file is print output. `onFileAvailable` returns:

`Type` {*string*} identifies the event type.

`Description` {*string*} This property is unused.

`Filename` {*string*} A supplied unique filename in the format `tterm_job_[unique number].pdf`.

`Data` {*string*} base64 encoded file content.

Examine the `onFileAvailable` event handler below:

```
const onFileAvailable = ({ Data, Description, Filename, Type }) => {
    switch (Type) {
        case “printer:pdf”:
            TTermSession.TTerm.downloadPDF(Data, Filename, ({ data }) => {
                if (data.result !== 0) {
                    //non-zero return value indicates an error.
                }
            });
            break;
        default:
            //discard
            break;
    }
};
```

On receiving the event the implementation can either parse the base64 data independently or utilize `downloadPDF`, a built-in method which takes a base64 encoded string and a file name string. This provides an opportunity to replace the generated `Filename` string with something more meaningful to the end user.

`downloadPDF` will trigger a download within the client browser.

## Printing on iPads

Printing to an iPad differs slightly from all other platforms. Where other devices will directly download the generated PDF, due to file system limitations for iOS, TTerm Connect will open the PDF in a new browser tab. From there the end user can print or open the PDF in a third-party app. End users will need to allow pop-ups on an iPad to view print requests.

The `Filename` property is not applied to PDFs generated on an iPad.

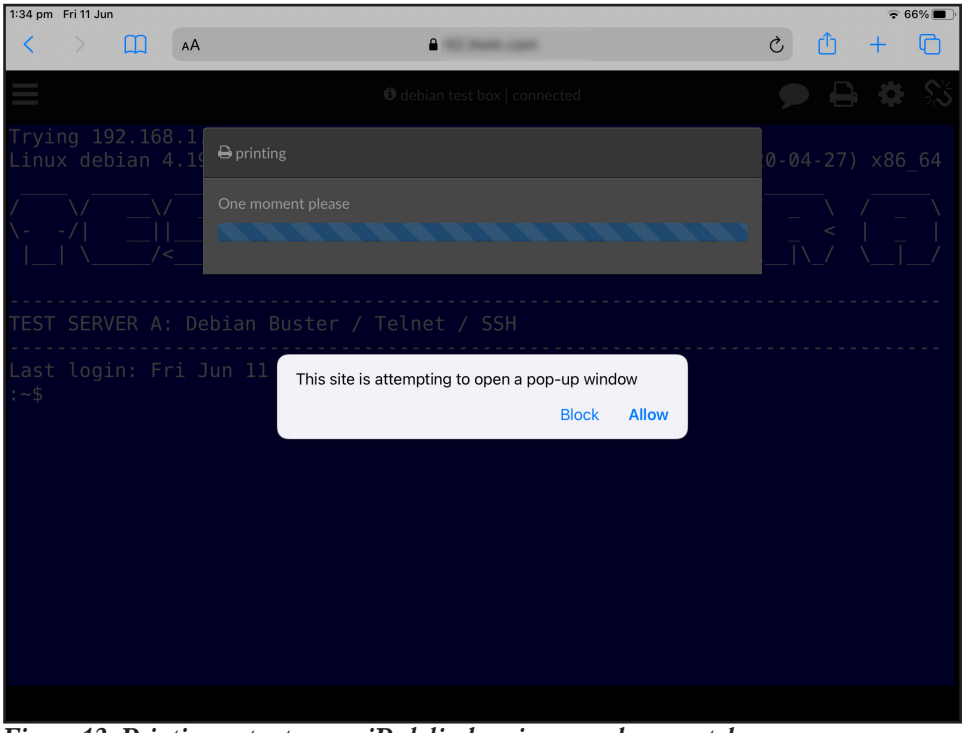


*Figure 12. Printing output on an iPad displays in a new browser tab.*

# Support and Further Information

## In-Program Help

TTerm Connect includes detailed in program help which may be accessed by clicking or tapping the ❓ help icon at the top right of screen in the Administration Portal and at top left of page in the Desktop Client. Help is searchable and throughout the application context sensitive links will open relevant help pages.

## TTerm Connect Javascript API

Your installation will include a copy of the **TTerm Connect Javascript API** document. This document outlines the various methods and properties available to you when developing an embedded implementation of TTerm Connect.

You can find this file linked from the examples title page located at ***https://myserver.com/examples/index.html***

## Contacting Turbosoft Support

For issues relating directly to TTerm Connect which are not covered by this document or the in program help content, consider checking the online resources available at ***www.ttwin.com/support*** or contacting Turbosoft support staff directly. To lodge a support ticket with Turbosoft support please visit ***www.ttwin.com/support-form*** and have the following details ready:

- Your **license details** - these can be found on the **About** page of the Administration Portal. Please also include any maintenance contract number or service agreement you may have with Turbosoft.
- **TTerm Connect version number** - TTerm Connect consists of a number of components and each will have its own, slightly different version number. These components are:
  - **The Administration Portal** - The web-based administration tool.
  - **The Desktop Client** - The optional web-based end user client.
  - **The Server** - The collection of server-based components which provide the core of TTerm Connects emulation and management processes.

  Please send the server license version and either the Desktop Client version OR the Administration Portal version number, depending on the source of the issue.

  Version details can be found on the **About** page of the Administration Portal and the settings section of the Desktop client. These details can also be found on the respective login screens.
- **Details of the client (end user) environment**, including operating system and browser, for example, *Windows 10, Chrome version 80.0.3978.102*

This information is essential to enable Turbosoft to resolve your issue in a timely manner.

## Contact Details


**Website:**

*www.ttwin.com*

**Email:**

*sales@ttwin.com*
*support@ttwin.com*

**Telephone:**

**Australia Toll Free:** 1800 628 646
**New Zealand Toll Free:** 0508 443 467
**North America Toll Free:** 1 800 519 9501

**International:** +61 2 9191 9270