

Customization 1 - Intro

Site Structure and Basic Concepts



Presented by Roger Sprik for the 2019 PSUG Middle East Conference
March 2019

Introduction

Customization can be defined as anything that changes the stock interface in PowerSchool. There is more than one method to customizing PowerSchool's appearance and content. The methods include:

- Localization (Translation)
- Custom Fields and Database Extensions
- Custom Screens - easy-to-make screens from the user interface, no coding required
- Customizing pages - either customizing existing pages or creating new ones
- Page Fragments - custom snippets that enhance an existing page
- Plugins - complete customizations that can be easily installed on other servers

The level of expertise required can start low with Localization and Custom Screens and range from medium to high if you customize existing pages or create your own custom pages. The areas you will encounter and will need to learn more about include:

- **HTML** - "Hyper Text Markup Language" - the language of web page structure
 - `This text will be italicized
This will be on a new line`
- **PSHTML** - "PowerSchool HTML" - the tags unique to PowerSchool web pages
 - `~([Students]grade_level)` will display the student's grade level
- **CSS** - "Cascading Style Sheets" - the language of web page style: how it looks
 - `<div class="box-round">` - PS "rounded box"
 - `<td style="text-align: right;">` - make a table cell right-aligned
- **Javascript** - the language for making interactive web pages
- **jQuery** - a Javascript library to simplify the scripting of web pages
 - `$j("#grade_-3").hide();` - hide the -3 grade level element
- **SQL** - "Structured Query Language" - the language for querying databases
 - `SELECT lastfirst, grade_level FROM students WHERE enroll_status=0`
- **Angular** - (for the advanced) an even more modern Javascript library to create powerful web pages that behave more like complete applications
 - ```
<tr data-ng-repeat="notice in filteredNotices">
 <td>{{notice.lastfirst}}</td>
 <td>{{notice.req_hours}}</td>
</tr>
```
  - An Angular table row that will repeat the name and required hours for each row in the data.

The wonderful thing about customizations is that they can be shared and easily copied to other PowerSchool servers. The large and active PowerSchool community actively and freely shares hundreds of customizations. Individuals and Companies market powerful customizations that you can pay for. Consultants are available for a fee to help you tailor a customization for your school.

It may seem daunting at first, but plan your path to becoming a Power Customizer. Start simple, then give yourself incremental levels of difficulty. Take on realistic projects that challenge you just enough to learn the next step. The possibilities are really only constrained by time and your willingness to learn.

## The “Best thing” and the "Worst thing"

It's been said that the best thing about PowerSchool is that it is customizable... and that the worst thing about PowerSchool is that it is customizable.

Any change you make to PowerSchool will mean more than customizing the page just once. Making sure your changes remain compatible with each release will also take time. Because the "page fragment" method of customizing can dramatically minimize the impact of upgrades, you should consider learning how to use page fragments, but it will involve more advanced coding. You also need to consider the risk and reward of dramatically changing PowerSchool functionality. This should NOT discourage you from making changes, but should prompt you to be thorough in your testing, your documentation, adopting a system of managing your customizations that works for you, and considering the eventual legacy you will leave to your successor.

## Localization or "Language Translation Toolkit' (LTK)

If you just want to change a label that appears on a screen, PowerSchool's translation feature is probably the best tool. You may think of the "Localization" or "Language Translation Toolkit" (LTK) as a way to show content in languages other than English, but it is also a very good choice for English -> English "translations". For example, if on the stock Parents screen you want "Mother's Name" to read instead "Parent 1 Name", you should use LTK. You can also use LTK to "rename" the field, which can be useful for searches and export as well. You can even use LTK to change grade levels like "0" to "K" and "-1" to "TK".

### **LTK Documentation:**

- search the built-in help on the term "Localization" or "Translation"
- find more information on PowerSource at <https://support.powerschool.com/article/74801> (v9)
- The Administrator User Guide since version 10 combines a lot of other guides into one very large document. (2,794 pages!!!)
  - v11 <https://support.powerschool.com/article/79085> (Page 894)
  - v12 <https://support.powerschool.com/article/80644> (Page 856)

# Custom Fields

## Resources

<https://support.powerschool.com/article/7744>

How It Works - Creating Legacy Custom Fields and Screens

<https://support.powerschool.com/article/77671>

Database Extensions Visual Walkthrough for PowerSchool 10.x

<https://support.powerschool.com/article/83171>

Database Extensions Advanced User Guide

<https://www.psugcal.org/index.php?title=Customization>

Southern California User Group - Look for the "Database Extensions Handout"

For a very long time PowerSchool has offered the ability to easily create custom fields and use them in custom screens, reports, searches, exports and lists. These are now called "Legacy Custom Fields" because since version 7.9, there is a new method for custom data entry points called "Database Extensions". Database Extensions currently run side-by-side with the legacy way of doing custom data, but eventually will completely replace it. As of version 12.1 certain sets of custom fields must be migrated to Database Extensions, such as core fields and state reporting fields, but migration of user-created custom fields is not yet required.

We'll cover the basics to get started. Learn more in the Database Extensions class or in the resources above.

### Legacy Custom Fields:

- Users go to System - Custom Fields/Screens
- Only Student, Staff (teachers table), Course and Section fields were available to be created this way
- Only text fields could be created
- All data went into one giant custom field that had to be parsed
- If user-created custom field migration has been performed on the server, you cannot create legacy custom fields
- Limit of 999 total fields for a given table (core + custom)

### Database Extensions:

- Users go to System - Page and Data Management - Manage Database Extensions
- Users create REAL tables and fields of various data types.
- Migration options for legacy custom fields: all-at-once (not recommended) or one-at-a-time.
- Are either **one-to-one** (i.e. one student, one value - "Vehicle Permit ID") or **one-to-many** (i.e. one student, many values - "College Applications")

- Many, many more tables can be extended, users can create one-to-many tables and even independent tables.
- Organized into "groups" with a group name. The group name becomes part of the field name in searches, exports, reports and custom pages. U\_Students\_Extension is the default group name for extensions of the Students table. Tip: You don't have to accept the default group names. **Keep group names unique and brief.** (U\_ETX, U\_VDM, U\_COLLEGE, etc).
- No more 999 field limit per table.

Using Custom Fields		
Functional Area	Legacy Custom Fields	Database Extensions (one-to-one)
In searches, lists, exports and wherever it asks for a "fieldname"	Fieldname Example: DistrictID	ExtensionGroupName.Fieldname Example: U_Students_Extension.DistrictID
Reports that use DATs	~(Fieldname) Example: ~(DistrictID)	~(ExtensionGroupName.Field_Name) Example: ~(U_Students_Extension.DistrictID)
Custom Pages (HTML)	[PrimaryTable]Fieldname Example: [Students]DistrictID	[PrimaryTable.ExtensionsGroupName]FieldName Example: [Students.U_Students_Extension]DistrictID

### More HTML examples:

#### Legacy custom field

*In a form:*

```
<input type="checkbox" id="staffchild" name="[Students]staff_child" value="">
```

*Read-only:*

```
<td>~([Students]PG1_cellphone)</td>
```

#### Database extension in a group named "U\_VCS"

*In a form:*

```
<input type="checkbox" id="intl" name="[Students.U_VCS]international" value="">
```

*Read-only:*

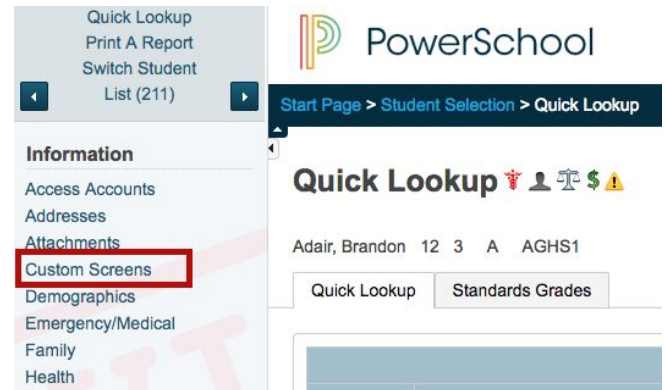
```
~([Sections.U_VCS]BTSN_Room)
```

# Custom Screens

The easiest way to create a custom screen in PowerSchool can be done right from the user interface. They require little to no coding expertise. These are the screens you see when you select a student and click on "Custom Screens" in the left navigation.

To create a custom screen:

- Go to System > Custom Fields / Screens > Custom Student Screens
- From there click "New", give it a name and submit.
- It will now appear in the list.
- Click "Edit Fields" to start adding content.
- Use the "Help" link for additional instructions.
- Tip: You can use html codes in the "Label" for line breaks, other formatting, links to other pages, etc.



## Custom Student Screen Fields: HS Parking Info

New			
Label	Field Name	Type	Sort Order
Permit #	permit	Entry field	1
Vehicle Information (one per line) Format: License - Vehicle info (incl color) Ex: 1ABC234 - 03 Ford Focus White	vehicle	Entry box	2

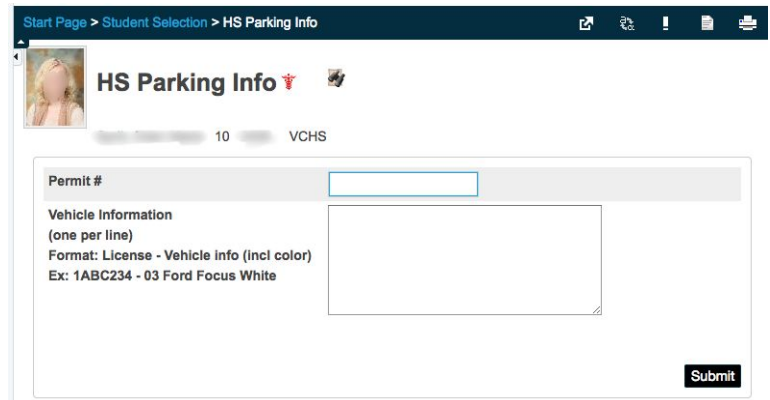
## Edit Custom Student Screen Field: HS Parking Info

Option	Value
Label	Vehicle Information (one per line) 
Field Name (Fields)	vehicle
Sort Order	2
Input Type	Entry box
Width (applies only to 'entry field' and 'entry box' types)	40 (width in characters)
Height (applies only to 'entry box' types)	8 (height in lines)

## Examples

Some examples of how you can use Custom Screens when you just need something quick and simple:

- Additional GPA information
- View/Edit your own custom fields
- Consolidate data from separate screens, such as data useful for the registrar/counselors



## Custom Pages

Custom pages are either custom versions of stock PowerSchool pages, or new custom pages that do not exist in the core product. Modifying or creating custom pages requires knowledge about HTML and PSHTML. It may also require varying levels of knowledge about CSS, Javascript, jQuery, SQL and/or Angular. Unfortunately, PowerSchool technical support cannot officially help you with your customizations. Fortunately, examples abound and the user group is always there to answer questions if you run stuck.

### Turning it on

Customization is not enabled by default. Go to Start Page > System > System Settings > Customizations and check the box. Turning off customizations temporarily can sometimes help you figure out if a problem you're experiencing is due to a customization

### Architecture

PowerSchool has a core set of web resources available in a folder on the server. It also has a separate location for custom web resources that parallels the core resources. When a user requests a page, the server checks the custom location first and will serve the custom version if it exists. If no custom version exists, then the core or "stock" resource will be rendered.

The core PowerSchool pages are on the server at:

```
<PS Install Folder>\application\components\
powerschool-core-<version#>\system\server\resources\web_root
```

Example:

```
C:\Program Files\PowerSchool\application\components\
powerschool-core-10.1.1.0.1223580\system\server\resources\web_root
```

Customizations do not edit the original core page or resource. Customization makes a new page, either a copy of the core page or a brand new custom page and saves it in the alternate custom location. If you need to remove a customization, you simply delete it. Since you never touched the core resource, the stock page is always there for the server to fall back on.

## Custom web\_root

For background information, the legacy customization method required direct access to the server and saving your custom pages to the custom web\_root folder. This legacy method is still supported and custom files saved directly to this folder still work.

The legacy custom pages are located at:

<PS install folder>/data/custom/web\_root

Example: C:/Program Files/PowerSchool/data/custom/web\_root

## Custom Page Management (CPM)

CPM was originally developed for "hosted" customers who did not have direct access to their server. However, there are other advantages to CPM, such as working from anywhere, templates, custom text keys and comparison tools. CPM is now widely used and recommended even for schools that host their own server. Another advantage is that CPM is stored in the database, so when you make a database backup, you are also backing up your customizations.

To begin using CPM you must enable the feature. If you have customizations that exist in your legacy custom web\_root, there is an option to migrate them. See PowerSource article 80649 (<https://support.powerschool.com/article/80649>) for the v12 guide.

To get to CPM before version 12.1

Start>PS Administrator>Custom Pages (*Separate portal requiring separate account*)

CPM for Version 12.1 and later

Start>System>Page and Data Management>Custom Page Management

(*Within admin portal, no separate account required.*)

### **Must setup permissions!!!**

- "What's New in PowerSchool 12.1" on PowerSource  
<https://support.powerschool.com/exchange/view.action?download.id=958>
- Presented on the PowerSchool Insider Episode 54

## The cache problem

For performance reasons, the PowerSchool server "caches" resources into memory, such as custom web\_root pages and wildcards. It refreshes this cache every 5 minutes. Most changes (especially in CPM) take effect immediately, but if you make a change to a custom page and don't see the changes rendered immediately, the you can either wait 5 minutes or manually refresh the cache. To manually refresh the cache simply turn Customizations off and back on again. Go to Start Page > System > System Settings > Customization and uncheck the "Customizations enabled" box and submit. Return to the page, check the box, and click submit again.



Another cache problem can be your local browser. Most browsers cache images, scripts, and css files. If you are doing an advanced customization to an image, script or css file, you may need to clear your browser's cache to see the changes.

## "I can't get rid of a customization"

With three possible locations for PowerSchool pages (Core, custom web\_root and CPM), things can get a little confusing. In order of priority, when a user requests a PowerSchool page, the server will decide which page to use in this order:

### Page Priority

**1. CPM (if it's enabled) 2. Custom web\_root 3. Core web\_root**

**Tip:** It's been known to happen that a customization is left in the custom web\_root folder and is forgotten. A more updated customization is made in CPM. Then at some point the customizer deletes the CPM customization and wonders why they are getting a strange, broken or older version of the page. It may be because the forgotten copy in custom web\_root is now being served. One way to test this and even work around it temporarily if you don't have direct access to the server is to go back into CPM and "customize" the page. Copy the original content into your custom copy without changing anything. If your problem goes away then you have an old copy of that page in custom web\_root and you can contact your hosting team or your server administrator to get it removed.

## Dissecting a PowerSchool Page

A web page is saved on the server, and then processed and "rendered" for the client. When a client requests a PowerSchool page, the server processes all the code, especially the PSHTML code, converts it to data, and delivers it to the client.

### Special Page extensions

- .htmlr Render the page in the browser but do not process special PowerSchool tags.
- .htmlt View the unrendered (server-side) page code in the browser window
- Right-click (Windows) or Control-click (Mac) on a PowerSchool page and choose "Inspect" or "View Page Source" to see the rendered (client-side) code.

## Customization Reference

For a good resource about tools, HTML, PSHTML, CSS, JavaScript / jQuery, SQL and more - a "customization reference" by Eric Schaitel is available as a plugin to PowerSchool. I recommend you install it from: <https://support.powerschool.com/exchange/view.action?download.id=772>

It's a plugin, so you if you haven't already, you must enable customizations and Custom Page Management. Once you install and enable the plugin from System > System Settings > Plugin Management Configuration, you'll find it from the District Office from the District Setup page.