

PowerSchool Special Programs Configuration Guide

For Version 25.1

POWERSCHOOL

PowerSchool Special Programs Configuration Guide

© PowerSchool Group LLC

Printed in USA

All rights reserved.

These materials may not be reproduced in whole or part in any form without written permission from PowerSchool.

Trademarks

PowerSchool is a trademark of PowerSchool Group LLC. Adobe is a registered trademark of Adobe Systems Incorporated. Apple®, Macintosh® and Mac® are registered trademarks of Apple Computer, Inc. and Safari™ is a trademark of Apple Computer, Inc. Firefox is a trademark of the Mozilla foundation. Microsoft and Windows™ are trademarks or registered trademarks of Microsoft Corporation.

Table of Contents

Overview of PowerSchool Special Programs (PSSP)	1
What's New in Version 25.x	1
What's New in Version 24.11	2
What's New in Version 24.6	3
What's New in Version 23.11	4
What's New in Version 23.6	5
What's New in Version 22.11	5
What's New in Version 22.6	10
What's New in Version 21.11	13
What's New in Version 21.6	16
What's New in Version 20.11	20
What's New in Version 20.6	24
What's New in Version 19.11	27
What's New in Version 19.4	29
What's New in Version 18.1	34
What's New in Version 18.0.1	35
What's New in Version 18.0	36
What's New in Version 17.1	37
What's New in Version 17.0.1 (Service Pack #1)	39
What's New in Version 17.0	39
What's New in Version 16.1.2 (Service Pack #2)	40
What's New in Version 16.1.1 (Service Pack #1)	41
What's New in Version 16.1	41
What's New in Version 16.0	41
What's New in Version 16.0.1 (Service Pack #1)	43
What's New in Version 16.0	44
Organization of Data in PowerSchool Special Programs	47
Profile Repository	48
Document Repository	51
Communications Repository	53

Configuration Tasks	54
Creating and Assigning a Configuration Task	54
Configuration Task Workflow	54
Profile Data Schema Customization (Basics)	56
Getting Started (Tour of Profile Configuration)	56
Understanding the Data Dictionary	63
Adding a Profile Form Section	68
Creating and Modifying Keyword Tables	72
Adding a New Data Field	78
Adding Profile Constraints	87
Adding an “Automatic” Profile Form Section	96
Profile Data Schema Customization (Advanced)	99
Adding a Calculated Field	99
Profile Field Attributes	103
Creating a Child Profile Type	106
Configuring Database Update Scripts	115
Configuring Deactivation/Reactivation of Profiles	120
Configuring Profile Tags	120
Configuring General Ed Students for On-Demand Importing	121
Configuring Profile Access via Alternate Location Fields	123
Many-to-Many Students/Locations Security Model	125
Configuring Student Data Transfer Envelopes	127
Configuring End-Of-Year Rollover	129
Service Capture Configuration	130
Reporting Snapshot Approval Process	140
State/Regional Module Related Configuration	142
Implementing Custom SQL User Defined Functions (UDFs)	144
Assessment Repository Customization	150
Configuring Analysis Fields in the Assessment Repository	150
Configuring Screening Groups	155
Configuring Progress Monitoring Groups	158
Document Template Customization (Basics)	167
Getting Started (Tour of Document Templates)	167
Creating a Document Template	172
Adding a New Section to a Document Template	176
Adding Fields to a Template Section	180
Setting up Public Statements for a Long Text Field	191
Document Template Security Configuration	194

Configuring Template and Section Properties	200
Setting Up Required Document Fields	213
Configuring Section and Document Actions	215
Best Practices for Validating User-Entered Document Data	235
Active Documents	237
Special Document Fields	240
Snippets	241
Document Template Customization (Advanced)	243
Document Owner Security	243
Document Workflow	244
Data Gathering	247
IEP "Snapshot" Section	250
Repeating Sections/Rows and Child Templates	251
Implementing a Goals Section	264
Section Extension Child Templates	270
Storing Uploaded Files Associated with Profiles & Documents	272
Self-Service Documents	276
Document Template Translations	279
Customizing Document Events	286
Customizing the Documents List View with Summary Extracts	287
RtI Intervention Plan Document Templates	290
Configuring Bulk Operations on Documents	293
Master/Detail Documents	294
Language Localization	296
Configuring DocuSign Integration	297
Configuring Digital Signature	304
Other Configuration for Digital Signature	308
Using DocuSign Configuration with Digital Signature	310
Optimizing Document Speed and User Responsiveness	311
Printing Special Document Content for Student/Parent/Guardian	312
Configuring Forms	314
HTML	314
CSS	314
HTML Best Practices	315
Accessibility	316
Best Practices to Support Translations	317
Field Directives & Validation Capabilities by Data Type	318
Using JavaScript and the DataFormAPI	323

Conditional Form Logic: #JSIF Directive	334
Conditional Form Logic: #IF Directive(s)	336
Pop-up Help Definitions and Guides	341
Assessment Directives	343
Profile Grid Directives	345
Configuring Packages	349
Advanced Reporting	351
Advanced Report Overview	351
Basic Advanced Reporting Directives	354
Advanced Report Measures	356
Creating an Advanced Report	357
Command (#) Directives	366
Adding Charts to Advanced Reports	366
“Expando” Report Sections	367
Setting Drilldown Columns	370
Optimizing Advanced Report Performance	370
Configuration Management Tool	371
CMT Overview	371
Model Versions	373
Comparing Databases	374
Synchronizing Configuration	375
CMT Renaming Fields and Other Objects	380
CMT Data Transformations	381
CMT Conditional Logic and Other Special Cases	383
PowerSchool SIS Integration	384
Overview	384
Alerts	384
HTML Format Directives Reference	388

Overview of PowerSchool Special Programs (PSSP)

Chapter

1

PowerSchool Special Programs (hereafter shortened to PSSP) is a case management system designed to address the the following areas:

- Special Education: Referral, Assessments (academic, health, psychological, etc), eligibility determination, Individualized Education Program, services delivery tracking
- Other special programs like Section 504, Speech only IEPS or services plans, English Language Learner (ELL) plans, Gifted, RTI (with progress monitoring embedded in document), etc.

This guide covers the functions of PowerSchool Special Programs platform to build a “model” that has data schema, forms, document templates, document workflow and business rules needed to meet compliance and related requirements for a state, province, or even a particular customer that may be large enough to warrant its own unique model. Such models normally address one or more of the programs mentioned previously: Special Education, Section 504, Speech, ELL, etc.

This guide assumes the following prerequisites:

- You are already familiar with all end-user aspects of PowerSchool Special Programs related to the areas for which you are building or customizing a model. These aspects are covered in the user-facing online user guides accessible from the application help menu.
- You have skills with relational databases, exposure to query or expression languages (experience with SQL is helpful but not essential) and mark up languages: HTML and CSS. Some experience with JavaScript will be helpful for more advanced or complicated scenarios.
- You are authorized to log into PowerSchool Special Programs and make these fundamental changes.

What's New in Version 25.x

- General Form Configuration
 - [New in 25.1.0.0] Section 508/Accessibility: The “Accessibility” section on page 316 describes a way to use \$ syntax when associating label tags with fields. The \$ syntax will now fully validate the field name in the “for” attribute such that if the field name is not valid, it will be called out as a misconfiguration. To label the No checkbox of a

logical field that allows empty values, use the <label for="\$LogicalField!"> syntax (note the exclamation point).

- [New in 25.2.0.0] **Section 508/Accessibility:** To label the time field of a date/time field, use the <label for="\$DatetimeField#TIME"> syntax.
- **Standard Reports**
 - [New in 25.1.0.0] **State/Regional Controller:** New settings are available that allow a model to behave optimally in the context of the state/regional controller module. These new options are currently all related to “merged” list reports (list report results merged from multiple district tenants). See the “State/Regional Module Related Configuration” section on page 142 for more information.
 - [New in 25.1.0.0] **Snapshot Reporting:** As described in the “Reporting Snapshot Approval Process” on page 140, it is possible to configure a snapshot reporting and approve process based on a child profile type. Normally, only CONSULTANT/ADMIN users can populate the snapshot data and click the “Approve Reporting Snapshot” button. But it is also possible to authorize a staff security group to do this by making sure the security group has the following privileges: 1) the new “Administrate Snapshot Reporting” reports security privilege, and 2) the system-wide editing privilege for the child profile type used for the snapshot data.

What's New in Version 24.11

- **General Enhancements:**
 - [New in 24.11.0.0] **Model Text Search Enhancements:** The model text search box on the user home page of the CONSULTANT that allows a CONSULTANT to search the model based on text has several enhancements. First, the search now has checkbox-based filtering that allows filtering the result list to certain high-level areas: Profile Types, Update Scripts, Keyword Tables, Document Templates. Second, the search widget can now find individual keywords and keyword table fields. **Note:** if you want to search for a keyword that is two-characters long and find that the search does not trigger until you have typed three characters, you can type an equals (=) character at the beginning of the search text to do an exact search for the keyword of interest.
- **Document Configuration**
 - [New in 24.11.0.0] **Document Action Enhancement:** Document actions with a trigger type of “Modify Data When Document Opened” in the past did not trigger when documents are bulk printed. This type of document action now has a check box option labeled “Execute for Bulk Printing”. If the option is checked, the document action will trigger prior to rendering the document during bulk printing.

- **Profile Configuration**
 - **[New in 24.11.0.0] New Child Profile Type “Internal Data Only” property:** If this checkbox property is enabled for a child profile type, it hides the child profile type from a number of screens for all non-CONSULTANT users: profile flyout menu, new standard report “type of information” dropdown, Administration > Utilities menu, the list of profile types available under Administration > Configuration > Profile Types. Effectively, this property hides the child profile type from end users when it will be used only internally.
- **Standard Reports**
 - **[New in 24.11.1.0] List Report Row/Column Colorization:** Previously, you could colorize rows of a standard list report by introducing a column with a title of “@Color” and formula that computes the color based on data being reported. While “@Color” continues to be supported for legacy reasons, the system now supports “@RowColor” and “@ColorColumn” columns. A “@RowColor” column colorizes the entire row (same as “@Color”) whereas a “@ColorColumn” column colorizes only the column to its left. A report may contain more than one “@ColorColumn” columns to colorize various columns differently.

What's New in Version 24.6

- **Document Configuration**
 - **[New in 24.6.4.0]** A new data modification option labeled “Modify Globals Fields” is now available in document/section actions. It allows document/section actions to update fields in the Globals profile.
- **Standard Reports**
 - **[New in 24.6.3.0]** For standard list reports that are under configuration management, the sort levels setup screen now has a new option for the state/regional module labeled “Do Not Group by District Tenant When Merging District Tenants”. When this option is checked, merged-tenant reports will not group the rows by district tenant. Instead, the defined sort levels will be applied across district tenants.
 - **[New in 24.6.3.0]** For standard list reports that are under configuration management, the selection criteria setup screen now has new settings under the “Advanced Model Report Settings” label. One such setting is a field set expression that can be used to update certain profile data fields that the report relies on for accuracy. The field set expression is generally used when there is already an existing update script that updates fields nightly that the report relies on for accuracy, but using this property, the fields could be updated just as the report runs as a supplement to the update script.

- **Advanced Reports**

- [New in 24.6.4.0] The configuration screen for report-wide inclusion rules now includes new settings under the “Advanced Model Report Settings” label. The field set expression works the same way as it does for standard list reports (as described above under What’s New > Standard Reports).

What's New in Version 23.11

- **Model Search**

- [New in 23.11.3.0] On the properties page for many model elements, CONSULTANT users will now see a button below the properties that can be clicked to search for other model elements that reference the model element for which properties are being viewed. The search results view supports previewing key properties or drilling all the way into items. The main purpose is to allow the CONSULTANT to review and understand any dependencies from a single view.

- **Form Configuration**

- [New in 23.11.6.0] When a filter formula is applied to a profile field directive using the F modifier, the lookup results immediately appear without the user needing or even being allowed to conduct a general search. But if you would like the user to be able to “escape” the filter and conduct a general search, you can now add a hint in the form of a comment appended to the end of the formula, like this: F"formula;AllowSearch".

- **Profile Configuration**

- [New in 23.11.3.0] The Unicode property is now available in profile character fields. This includes the build-in name fields such FirstName, LastName and MiddleName; and in this case, the property is automatically propagated to the linked name fields in document templates where relevant.

- **Document Configuration**

- A new document template property option labeled ‘Compress Long Text Fields’ can be a helpful solution when the maximum number of top-level fields in a document template (about 1000) has been reached or is at risk of being reached soon. Note that a warning appears when adding a field and less than 100 additional fields (estimated) can be added until the maximum is reached. When the new option is enabled, any top-level long text fields are compressed into a single database column by the database engine in a way that is largely invisible to the application and to users, although there may be a slight decrease

in performance when accessing such fields. With this option enabled, any new or existing top-level long text fields collectively count as 1 towards the limit.

- [Advanced, New in 23.11.1.0] When adding a new child template, there is now a new property labeled “Child Template Type” that has two choices. The default “Standard (Repeating)” choice creates the normal type of child template used for repeating sections and repeating rows. The second “Section Extension (Non-Repeating)” choice creates a new type of child template designed as a solution to the limitation of approximately 1000 top-level fields in document templates. For more information, see the “Section Extension Child Templates” on page 270.

What's New in Version 23.6

- **Profile Configuration**
 - A new “CEDS Element ID” property has been introduced for profile fields. This optional property can be used to map profile fields to data elements defined in “Common Education Data Standards” (CEDS).
- **Document Configuration**
 - [New in 23.6.4.0] Document actions with the “Prevent Creating Document” trigger are executed against the profile to determine whether it is valid to create a document of that type. This type of document action has been enhanced to allow references to DocHistoryYear which will be set to the target history year of the new document (in order to check whether that history year is appropriate).
- **Advanced Report**
 - By default, the system infers default columns when drilling down into each report cell based on cell rules and other heuristics, and an end user can customize the inferred default columns. A new feature has been introduced giving the report configurator the option of configuring a specific set of columns in a particular order, which then replaces the inferred columns as the default columns. To do this, select a configuration task, switch to the Layout tab, and select Setup > Set Drilldown Columns. The end user can still customize the default columns whether they are inferred or preset as part of the report definition. Since the preset columns are part of the report definition, they are synced by the CMT.

What's New in Version 22.11

- **General Configuration Productivity**
 - Dropdowns that tend to have a lot of items in profile, document and standard reporting configuration dialog boxes have been enhanced with “typeahead” filtering to make items

easier to select. Examples include the data type dropdown when creating new profile/document fields, and the “type of information” dropdown when creating new standard reports. Simply click in any dialog box dropdown that support this and start typing to quickly find any items that contain the text you are typing.

- **Form Configuration (Profiles/Documents)**

- **[New in 22.11.3.0]** A common pattern is to show a line with a value from the globals profile using a directive like this:

```
<div>{@GlobalField}</div>
```

However, this can result in a blank line if *GlobalField* is empty since the directive, by default, renders an empty value as placeholder whitespace. The following format is now supported for convenience removal of the blank line:

```
<div>{@~GlobalField}</div>
```

This is now supported for all directives that show the value of a calculation. Just introduce a ~ character before the formula itself, which suppresses the white space.

- **[New in 22.11.1.0]** When displaying a calculated logical field on a form via a field directive, you now have more control over how it is presented to the end user. By default, such fields are presented a read-only checkbox. You can present it instead as dual checkboxes using {LogicalField:+"On"- "Off"} or as a simple text value using {LogicalField:D+"On"- "Off"}.
 - **Improved JavaScript API support for lookup/non-lookup fields:** Enhancements have been made to facilitate tracking changes to lookup/non-lookup combination fields via the DataFormAPI. In the directive below, the specified JavaScript function (configured in the relevant form’s JavaScript tab) will now be called when either the lookup or the non-lookup changes in value. From this function, you can use the new DataFormAPI.getFieldValue('LookupField') function to obtain the value of either the lookup or non-lookup based on which is selected. You can detect if the non-lookup is selected using the new DataFormAPI.getIsNonLookupSelected('LookupField') function. You can even use the new DataFormAPI.invokeNonLookup('LookupField') function to switch the field to the non lookup from your function (which has the same effect as the user clicking non-lookup).

```
{LookupField:O"NonLookupField"J"api:onChangeFunction"}
```

Important: The lookup/non-lookup combination is always referred to from the DataFormAPI using the name of the lookup field and not the non-lookup field.

- **Improved JavaScript API support for file fields:** Enhancements have been made to facilitate tracking changes to file fields via the DataFormAPI. A JavaScript notification function can now be specified for a file field as shown in the directive below. Additionally, DataFormAPI.getFieldValue('FileName') will now return the name of the file as a string, or the empty string if no file is selected.

```
{FileName:J"api:onFileFieldChanged"}
```

- **Improved JSIF support for file fields:** JSIF directives now have full support for referencing file fields. In some scenarios, this can be used to implement a workable solution for allowing the user to upload 1-N files at a particular point on a form, including in a child profile where other approaches are not available.

- **Profile Configuration**

- **[New in 22.11.4.0]** Field-directives for multi-value fields now automatically filter the values based on the InUse keyword table column. Previously, one needed to use the V modifier like this: {*multivaluefield*:V"InUse"}. Furthermore, the V modifier has been enhanced to support more than one column (including InUse as a column is acceptable but not necessary).
- **Bulk Operation Enhancement:** In previous versions, profile constraints can be configured to be available to staff users as bulk operations provided that such constraints meet the following criteria: 1) the "Is Evaluated For" option is set to "Bulk operation with (edit security bypass)", and 2) the constraint is associated with a security group for which to enable this capability. However, there previously was no way to allow the ADMIN/CONSULTANT to access such constraints. Now such constraints will be available to ADMIN/CONSULTANT users if the "Do not evaluate for ADMIN/CONSULTANT" option is not enabled. Upon upgrading to 22.11.0.0, the "Do not evaluate for ADMIN/CONSULTANT" option of existing bulk operation constraints will be preset to enabled for the purpose of preserving pre-existing behavior.
- **"Student at a Glance" Flyover Enhancement:** This functionality is now integrated into the student search box on the home page. Note that this flyover can be configured to present key summary data on students to users viewable from any place the student appears on end users' home page. The flyover can be differentiated by role, and can be configured to show the most critical status information that role users need to see, such as IEP due date, reevaluation due date, etc. A flyover is configured simply by adding a new profile form section with the appropriate content and setting the section property labeled "Purpose of section" to "Flyover".

- **Document Configuration**

- **[New in 22.11.3.0]** When renaming a document template section, the system shows an alert reminding you that the prior may need to be retained in the prior name(s)

field if the prior name had been synced with the Configuration Management Tool. Furthermore, the alert provides an easy one-click button to do it for you.

- [New in 22.11.3.0] Long text fields in documents now support a minimum and maximum length property (constraint). The system allows text that does not meet such constraints to be saved but does prevent the overall section from being marked as complete until these constraints are met.
- [New in 22.11.3.0] There is a new checkbox option in child template properties labeled “Replace Child Documents When Manually Copying from Other Document”. In certain configurations where child documents are prepopulated with document actions, manual copying can result in the appearance of seemingly duplicate child documents. This option can be used to prevent that by replacing existing child documents with those from the source document.
- [New in 22.11.3.0] Under certain conditions, child templates with “child profile to child document” data flow can be configured to “reflow” when a draft document is updated from the profile at a time later than its creation. At this time, this is not supported for child templates used in any repeating sections. A prerequisite to enabling “reflow” is to configure a profile reference field in the child template that refers to the source child profile type and has the “Child Document Title” and “Read Only” properties enabled. Note that only one field in the child template can have the “Child Document Title” property enabled. After configuring this field with the required properties, a “Reflow When Draft Document is Opened” checkbox option will appear in child template properties. When this is enabled, the initial data flow upon document creation will automatically populate the profile reference field with the “Read Only” and “Child Document Title” properties so that it refers to the source child profiles. When the draft document is later updated from the profile, child documents that refer to a source child profile will be updated (or deleted if the child profile has been deleted), and any new child profiles will be inserted with the profile reference field populated.
- [New in 22.11.1.0] If flowback is configured from a child template to a child profile type and you wish to limit which child documents flow to the child profile type, you can now add a logical data or calculated field named AllowFlowback to the child template. Only child documents for which AllowFlowback is true will flow back. Note that if child documents with AllowFlowback=true flow back during an initial flowback, but then flowback a second time with AllowFlowback having changed to false, child profiles from these child documents will be removed.
- By default, existing documents will copy their current Behavior Option value when revised, even if that value differs from the district or model’s value. An “Apply When Revising” sub-option has been introduced that (when selected) will apply the district’s Behavior Option value to the newly created revised document upon

revision instead of copying the original document's value.

(Behavior Option)

(Prevent Applying to Existing Documents)

(Apply When Revising)

- You can now specify a default value for document fields that have document-to-profile data flow but not profile-to-document data flow.

- **Formula Language**

- A new **PositionPattern**(*searchpattern, teststring*) function returns the character index (starting at one) of the first occurrence of *searchpattern* in *teststring*, or zero if not found. For details, see the online “Formula Reference” at the following URL:
<https://docs.powerschool.com/SEPUSR/formula-reference/character-functions>.
- New **TranslationCreatedOn**('languageid') and **TranslationFinalizedOn**('languageid') functions have been introduced into the formula language to allow access to dates that a particular document language translation was added to a document or finalized provided that such document language translations were added or finalized with version 22.11.0.0 or later. For details, see the online “Formula Reference” at the following URL:
<https://docs.powerschool.com/SEPUSR/formula-reference/special-document-values-and-functions>

- **Student Transfer Envelopes**

- When a model will be used in conjunction with a state/regional controller, it is now possible to configure service capture records (e.g., Service Records) such that they will be eligible for inclusion in student transfer envelopes. For full details on this and other aspects of configuration for student transfer envelopes, see the “Configuring Student Data Transfer Envelopes” section on page 127.

- **Advanced Reporting**

- **[New in 22.11.3.0]** The pie chart directive supports new attributes that can be used to improve the presentation of the resulting pie chart. For more details, see the “Adding Charts to Advanced Reports” section on page 366.

What's New in Version 22.6

- **Form Configuration (Profiles/Documents)**
 - [New in 22.6.4.0] The ^ field directive modifier makes a field disabled (grayed out) to the user but still writeable by javascript via the Form API. Previous this was only supported. Previously, this was supported for character, shorttext, and long text data fields. It now supports additional data types: integer, numeric, date, date/time, keyword.
 - [New in 22.6.4.0] Directives such as {#IF_ADMINCONSULTANT}, {#IF_EDITANDADMINCONSULTANT}. which previously allowed only ADMIN/CONSULTANT users to access administrative content and fields on a form, now also allows staff users with a new System Administration security privilege labeled “Access Admin Form Content” to access the content. Previously, this pattern required specifying security groups by name in the InAnyOfSecurityGroups function, which easily breaks when security groups are renamed.
- **Profiles**
 - [New in 22.6.1.0] When configuring profile constraints, macros are now supported in the “User Message” constraint property, allowing for dynamic user messages that include the values of fields. For student profile constraints, the following macros are supported by default: {ID}, {FirstName}, {LastName}, {MiddleName} and {FirstLastName}. You can make additional fields available for use in macros by setting the ‘Always Available for Macros’ profile field property for those fields. When configuring a profile constraint, the available macros are listed above the “User Message” field as links that can be clicked for easy insertion. Only macros that are listed will work.
 - [New in 22.6.1.0] A new “Send Message” profile constraint action has been introduced that will send a specified message when triggered. When configuring a “Send message” profile constraint, you specify a message subject and body (which both support the same macros as the user message) and can optionally mark the message as high priority. To prevent sending overly frequent messages, it is often helpful to pair this section action with another action that sets some type of flag that indicates that a notification is needed. The “Send Message” constraint supports an optional “field set” expression that can be configured to clear the flag after sending the message. Alternatively, the “Send Message” constraint can be directly linked to a preceding constraint via the “Preceding Constraint” property, which is visible when “Is evaluated when...” is set to “Section edited, the preceding constraint has been triggered”.

- [New in 22.6.1.0] A new “Insert Child Profile as Secondary Parent” profile constraint action has been introduced, which is very similar to the existing “Insert Child Profile” constraint, but with a key difference. “Insert Child Profile as Secondary Parent” constraint allows you to insert a new child profile for any child profile type for which the current top level profile type is the secondary parent. For example, you can configure a student profile constraint with “Insert Child Profile as Secondary Parent” to insert the student into the caseload for a specified staff user (which is impossible using “Insert Child Profile”). You can also configure such a constraint in a child profile of students to insert the student in the caseload for a specified staff user.
- **Documents**
 - [New in 22.6.4.0] **Enabling Student Transfer Envelopes to Transfer Final Live Documents:** Normally, documents transferred in a student transfer envelope as live documents are recreated in draft mode in the receiving database because it is typically not possible to repopulate all fields (particularly staff fields) leaving any number of sections incomplete (i.e., required fields may be empty). However, scenarios such as allowing the receiving school district to complete a progress report for an IEP from the sending district require a solution for this limitation. Note that progress reports are typically configured to be editable when the document is final. A new “Support Transferring Final Live Documents” document template property has been introduced that can be enabled to allow final documents to be recreated in the receiving database such that the document status is set to final even while ignoring the fact that required staff and other profile reference fields may be empty. However, the progress report scenario will typically depend on certain staff fields being populated which will be empty after the transfer. To resolve this, one can: 1) add a logical field named “FromStudentTransferPackage”, 3) add a document action with a trigger type of “Modify Data For New Transfer Envelope Live Document” (new in 22.6.4.0) that sets the “FromStudentTransferPackage” field to true, 4) have additional content in the progress report (assumed to be editable in final) that allows a user to edit the key staff fields if “FromStudentTransferPackage” is true. For full details on configuring for student transfer envelopes, see the “Configuring Student Data Transfer Envelopes” section on page 127.
 - [New in 22.6.2.0] **Child Template Data Flow:** The system now supports multiple child templates (from same or different document templates) flowing back to the same child profile type, with or without maintaining prior history. If a child template is configured to not retain prior history, then when it flows back to the child profile, earlier child profiles that came from the same child template will be deleted. Previously, flowing more than one child template back to the same child

profiles would only work if all child templates had the same setting for maintaining prior history.

- **Standard Reports:**

- [New in 22.6.4.0] When setting up report parameters for a standard report, it is now possible to specify default values for non-optional report parameters. The default values are specified in the form of a Globals profile expression. This is useful, for example, when needing to set default values for date parameters based on key dates stored in the Globals profile or the current date.
- [New in 22.6.3.0] It is now possible to display the value of a formula expression based on the Globals profile in the report header of a standard report. This is done by inserting a macro in the following format anywhere in the header:
`{=@globals_formula}.`

- **Digital Signature Configuration:**

- [New in 22.6.4.0] The “Pdf Creation Options” Digital Signature setting was added. This setting lets the user create multiple signed PDFs instead of combining all signatures into one PDF.
- [New in 22.6.4.0] Other Signer roles can have a “Response Priority Expression” defined for them that will determine which signer’s data will be selected as the document’s data when multiple signers have data for the same fields.
- [New in 22.6.3.0] The Digital Signature process has been modified so that data is saved directly to the document as the signer completes the process. After the signer is finished, the background service will update the document with the final signature data if necessary.
- [New in 22.6.3.0] The Digital Signature “Sign Now” feature was added.
- [New in 22.6.2.0] The “Notify staff when a signed document is modified” Digital Signature setting was added. When checked, staff users will see an info message when accessing a document that has been modified after it was signed.
- [New in 22.6.1.0] The way “Prevent Section Completion” section actions are run by the signing process has been changed. The “Only when signing document”, “When editing and signing document”, or “Only when signing if Digital Signature is enabled; otherwise, when editing” options will still take effect, however the system will also only run section actions where one or more fields referenced by the trigger criteria are editable by the signer. If none of the fields referenced by the trigger criteria are editable by the signer, then the section action is not checked for that signer.

What's New in Version 21.11

- **Keyword Tables**
 - [New In 21.11.1.0] A convention has emerged that keyword tables have an “InUse” logical field to track which keywords are still in use (field has Yes value) versus deprecated (field has No value). This has now been formalized such that the platform will: 1) automatically set the default value of such columns to true, 2) automatically add this column to new model keyword tables, and 3) automatically filter on this column for keyword fields in profiles and documents. Previously, the F“InUse” modifier was applied to field directives for such keyword fields (e.g. {KeywordField:F“InUse”}), but this is no longer necessary.
- **Profiles**
 - [New in 21.11.3.0] A new function (DataFormAPI.validateAddress) added to verify address against MAR API. See the “JavaScript” section for more details.
 - [New in 21.11.2.0] Profile staff fields now have a new checkbox property labeled “Preset to Current User”, which comes into play when a staff user manually adds a new profile or child profile. A staff field with this property will be preset to the current staff user who is manually adding the profile. The field will be left empty if 1) user adding the profile is not a staff user (i.e., is an ADMIN or CONSULTANT user), or if 2) the profile was not added manually (i.e., via integration), or 3) the staff field is not on the form actually used to add the profile (so therefore the field must be on the form used to add the profile).
 - [New in 21.11.2.0] Profile forms now support two new directives that make it easy to include HTML formatting only if the current user is specifically a staff user. The two new directives are {#IF_STAFFUSER} and {#IF_EDITANDSTAFFUSER}.
 - There is new type of constraint action labeled “Create Document” that creates a new document from a specified document template and lands the user in the document for editing. The trigger criteria should typically check if a document already exists to avoid creating multiple documents. Furthermore, the constraint will only trigger if the current user is authorized to create the document. This type of constraint is not available for bulk operation constraints.
 - To support easier messaging from students/parents from the student/parent portal, student profile staff fields can be marked with new checkbox properties labeled “Enable Messaging from Students” and “Enable Messaging from Parents Guardians” respectively. When students/parent users compose a message, the “To/Cc/Bcc” popup will recommend any staff from staff fields marked with these

properties. Note that it is possible for system administrator to override these properties.

- Introduced a new profile type property labeled “Allow Add New Profiles?” which, when set to false, prevents staff and ADMIN users from manually adding new profiles. The property does not restrict consultant users from manually adding profiles.
- **Update Scripts:**
 - [New in 21.11.1.0] When inserting target profiles from source profiles in an update script, a new “CAPTURE” syntax can now be used to record in each source profile the target profile that was inserted from it. The “Capture” syntax, shown below, requires the source profile type to have a profile reference field to the target profile type, and the name of this profile reference field must be specified immediate after “CAPTURE” as shown below. The pattern below includes “WHERE *profile_reference_field IS EMPTY*” to ensure that if the update script is executed multiple times, that only one target profile is ever inserted from each source profile.

```
INSERT target_profile_type_name (target_columns) FROM  
source_profiletype_name CAPTURE profile_reference_field (source_columns)  
WHERE profile_reference_field IS EMPTY AND other_conditions
```

- **Documents**
 - [New in 21.11.2.0] Document forms now support a new directive that make it easy to include HTML formatting only if the current user is specifically a staff user. The new directive is{#IF_EDITANDSTAFFUSER}.
 - [New in 21.11.1.0] A new child template data flowback property has been introduced labeled “**Retain data flow history from past documents**” on the child template properties screen. When child documents flow back to child profiles, by default, any existing child profiles are wiped out and replaced by a new set of child profiles copied from the child documents. However, if you enable this property, then only child profiles that may have flowed earlier from the same document are replaced by a new set of child profiles copied from that document. This allows a history to accumulate across multiple documents. Note that prior to version 21.11.1.0, you would need to enable the “Child Document Data Flow History Enabled” property in the child profile type to achieve the same effect, but that child profile property is no longer needed and has been deprecated and removed as of 21.11.1.0. The upgrade to 21.11.1.0 will automatically set the new

child template property such that the actual behavior of existing child templates remains the same.

- [New in 21.11.1.0] One can now reference DocStatus (document status) from #JSIF directives using syntax such as DocStatus.Review, DocStatus.Draft/Review (document is draft or final), or ~DocStatus.Final (document is not final).
- [New in 20.11.1.0] Documents can now insert and update certain top-level profiles using “Modify Document Data Via Script” document/section actions. For example, you can insert top level profiles from child documents, and if the child template has a profile reference field to the target top-level profile type, you can capture each inserted top-level profile back in the child document using the CAPTURE syntax (see capture syntax introduced for profile update scripts). The captured profile reference field can then be used to potentially update the top-level profiles later from the child documents.

```
INSERT top_level_profile_type (target_columns) FROM  
#doctemplateid#childtemplateid CAPTURE profile_reference_field  
(source_columns) WHERE profile_reference_field IS EMPTY
```

```
UPDATE top_level_profile_type FROM #doctemplateid#childtemplateid SET  
targetfield1=sourcefield1, targetfield2=sourcefield2, ... WHERE  
THIS=profile_reference_field
```

- It is now possible for student/parent portal users, when reviewing a document, to send a message linked to the document in the same way that staff users can. However, the option is only available if one or more staff fields in the document template are marked with the new “Enable Messaging from Students” and “Enable Messaging from Parents/Guardians” checkbox field properties. When students/parent users compose a message with a document linked to it, the “To/Cc/Bcc” popup will recommend any staff from staff fields marked with these properties. Note that it is possible for system administrator to override these properties.
- The formula language now supports a special value “OriginalDocumentIDT” in the context of document templates that support revision documents. It gives access to the unique identifier (integer value) of the original document if the current document

is a revision document, or EMPTY if the current document is not a revision document.

- **Digital Signature Configuration:**

- Required fields in signature areas will no longer prevent completion of a section when Digital Signature is enabled. This is to allow users to ignore those fields when pre-filling data, but still allow them to be required when signing or if Digital Signature is off.
- Prevent Section Completion section actions have had new options added to them. You can select from among “Only when editing document”, “Only when signing document”, “When editing and signing document”, or “Only when signing if Digital Signature is enabled; otherwise, when editing”. This setting will control whether this section action will prevent section completion when editing, signing, or some combination of the two.
- The document action “Modify Data when Docu Sign Status Changes” has been renamed to “Modify Data when Electronic Signature Status Changes”, to account for the fact that it applies to both DocuSign and Digital Signature requests.
- The section property “Omit from DocuSign” has been renamed to “Omit from Electronic Signature” to account for the fact that it applies to both DocuSign and Digital Signature requests.

What's New in Version 21.6

- **Standard Reports**

- [New in 21.6.4.0] There is a new list report column property labeled “Include in Data Export Only” that may be useful in list reports designed primarily for exporting data (e.g., state reporting) and especially where there are many columns for export that make the report awkward to view on the screen. The property can be applied to a subset of columns that are technical and/or not particularly helpful to end users. Columns with this property will be visible when the report definition is being edited but not when the report is being viewed or printed. Such columns will of course be

included in data exports. To see this new property in the popup for a columns's properties, you may need to click the "More Options" button.

- [New in 21.6.3.0] The functionality to allow configuration of a list report-based reporting snapshot has been enhanced to support a due date for the snapshot approval. See the "Reporting Snapshot Approval Process" section on page 140 for details.
- **Formula Language**
 - [New in 21.6.4.0] There is a new JoinCharacterValueOf formula function that can be used to return a list of character values taken from multiple child profiles of a top-level profile, or multiple child documents of a parent document. The character values are returned as a single character string with all the values in it separated by a delimiter that you specify as a parameter. This new function has the same structure as the TopOneValueOf function except that it requires specification of the value delimiter as the second parameter. Also, the third parameter identifying the values being joined into the list must be a character expression (note that other data types can be converted to a character type with data type conversion functions). The JoinCharacterValueOf function may return duplicate values if they exist in the source records. There is an alternate function named JoinDistinctCharacterValueOf that will suppress duplicate values. JoinDistinctCharacterValueOf has the same syntax as JoinCharacterValueOf with the exception that it does not support a sort order parameter. Instead, the distinct version of the function always sorts by the character values being joined.

Examples:

```
JoinCharacterValueOf(ServiceRecords, ',', Service.Keyword)  
JoinCharacterValueOf(ServiceRecords, ',', Service.Keyword, Service IS NOT EMPTY:  
Service.Keyword)  
JoinDistinctCharacterValueOf(ServiceRecords, ',', Service.Keyword, Service IS NOT  
EMPTY)
```

- [New in 21.6.3.0] The formula reference in the online user help has been improved to fully document the pattern matching capabilities of the LIKE operator, which are already available in earlier versions. The like operator is commonly used with the % wildcard character which matches any of zero or more characters, for example:

FirstName LIKE “%John%”. But the following matching lesser-known methods of pattern matching are also available:

- _ (underscore) matches any one character
- [...] matches any single character within the specified range (e.g. [0-9]) or set (e.g. [abcdef])
- [^...] matches any single character that is NOT within the specified range (e.g. [^0-9]) or set (e.g. [^abcdef])

- **[New in 21.6.3.0]** Formula syntax decoration has been introduced into many view-only configuration screens that show formulas. For example, field names within formulas have tool tips, retired field names have a strike-through, and more. This provides a richer and more informative experience examining the formula.
- When child profiles have the same field names as top-level fields (e.g. Student Contacts and Student Profile) there is now better support for qualifying which profile is being referenced using a dot operator as a qualifier (e.g. Students.EmailAddress versus StudentContacts.EmailAddress). As of version 21.6.2.0, similar qualification is available for child template fields.
- The SectionCompleted function now works for repeating sections, and in this case, is true only if the repeating section is included in the document and every instance of the repeating section is completed.
- A new formula editor has been introduced throughout the application including the configuration area. It assists with writing any kind of expression (document/section action expressions, calculated fields, update scripts, etc). The new editor provides consistent labeling across the application to provide configurators with immediate information on the exact context and nature of each formula. Please be aware that if the auto-complete feature is not offering the expected field, the following may be explanations:
 - The formula editor never recommends retired fields.
 - The formula editor recommends “Internal Value Only” fields only when the current user type is Consultant.
 - Normally, the unique field name and caption are very similar, but if not, attempts to use the caption to filter to the field may not bring up the field.
- **Profile Configuration:**
 - **[New in 21.6.4.0]** Previously, a {=^formula} directive could be used in a child profile form to specify that the formula is in the context of the parent profile. This allowed the directive to function even when the form is being used to add a new child profile, whereas the standard {=formula} directive does not in this case. Similar directives have now been introduced for conditional directives as follows:

- {#IF^ formula}content{#ENDIF} *Show content if parent profile formula is true (regardless of view/edit mode)*
- {#IFVIEWAND^ formula}content{#ENDIF} *Show content if in view mode AND if parent profile formula is true*
- {#IFVIEWOR^ formula}content{#ENDIF} *Show content if in view mode OR if parent profile formula is true*
- {#IFEDITAND^ formula}content{#ENDIF} *Show content if in edit mode AND if parent profile formula is true*
- {#IFEDITOR^ formula}content{#ENDIF} *Show content if in edit mode OR if parent profile formula is true*
- {#IF_RO^ formula}content{#ENDIF} *Force any otherwise editable fields in the content to be read only if parent profile formula is true*

- **Document Configuration:**

- [New in 21.6.1.0] There may be certain cases where a child template calculated field flows back to a child profile data field, and the calculated field formula refers to a top-level field editable in a different section that does not contain the child template. In this edge case, it may be desirable to have the child documents flow back when the section completed where the top-level field is editable, given that this field will influence or alter the data that flows back. You can accomplish this by enabling the “Trigger Child Profiles Flowback When Referenced Fields Edited” field property of the child template calculated field. Note that this property is only visible when the child template is configured to flow back to child profiles when the section containing the child template is completed.
- The document “Verify All” function has been improved to report any use of retired fields on non-retired sections.
- The existing “Behavior Option” document field property allows the system administrator to preset the field’s value in all newly created documents, and normally the ADMIN has the option of setting the field’s value in existing document as well. Version 21.6 introduces a “Prevent Applying to Existing Documents” sub-option (see below) that can be used to prevent system administrators from applying the behavior to existing documents when that would not be appropriate.

Behavior Option

(Prevent Applying to Existing Documents)

- **Digital Signature Configuration:**

- Configurators are able to configure a document for use with PowerSchool Special Program’s new Digital Signature functionality. See the “Configuring Digital Signature” section for more information.

- Documents configured for DocuSign are now compatible with PowerSchool Special Program's new Digital Signature functionality. See the "Using DocuSign Configuration with Digital Signature" section for more information.
- **[New in 21.6.2.0]** When a signed document is accessed, the text 'Signed by <signer name>' or 'Initialed by <signer name>' will be displayed where the signer signed or initialed the document.
- **[New in 21.6.2.0]** The signing process has been internationalized. When sending a signature request, the signer can select which language the signer's request email will be sent in. When signing a document, the signer's browser settings will determine what language the document is rendered in (assuming the signer's browser language matches one of the available translations).
- **[New in 21.6.3.0]** The :L"label" modifier can be used with the :S or :I modifiers to automatically display a line and label underneath the signature or initials field.
- **[New in 21.6.3.0]** #IFSIGN areas are now editable by staff users before a signature request is created, just as if they were standard document areas.
- **[New in 21.6.4.0]** The eSignature Email Subject and Blurb will be used by the Digital Signature system when sending a signature request.
- **[New in 21.6.4.0]** When a document is configured to "Allow Submit Review Document", it can be submitted for Digital Signature while the document is in Draft. The document will be promoted to Review before being submitted for Digital Signature.

What's New in Version 20.11

- **Formula Language**
 - **[New in 20.11.4.0]** Previously there was a limitation that one could not use a dot operator with a keyword field within the expression of certain aggregate functions like SumOf. This limitation has been removed.
- **Profile Configuration:**
 - **[New in 20.11.4.0]** When modifying the sort order of a child profile, ascending or descending order can be selected for any data type.
 - If child profiles of a particular type will never be manually added, edited, or deleted by staff users, there is a new child profile type property named "Hide Add/Edit/Delete Privileges?" that allows you to hide editing privileges from any security privilege assignment screens. This is useful when child profile data is readonly and only comes in via integration. This property is overridable by the system administrator.

- **Document Configuration:**

- [New in 20.11.1.0] A “custom” attribute has been introduced for child templates and grand child templates.
- [New in 20.11.1.0] To facilitate using a section action to insert a repeating row in the current instance of a repeating section, an “Insert Child Document” section action for a repeating section now supports inserting a grand-child document associated with the repeating section.

- DocuSign Configuration Enhancements
 - [New in 20.11.1.0] The directive for an “initial here” signing field now supports an “O” modifier to make it optional.
 - [New in 20.11.1.0] If signing fields for any configured signers are omitted from the document for any reason (section inclusion, #IF or #JSIF logic, etc.), the system now automatically excludes such signers when submitting to DocuSign. This should simplify the overall DocuSign configuration for document templates by allowing simpler formulas to be used for non-staff signers, and by allowing staff fields to be used in more cases.
 - [New in 20.11.1.0] Additional DocuSign-specific directives have been introduced to make configuring document templates:
 - {#IF_DOCUSIGN_ACCOUNT_EXISTS}content{#ENDIF} includes the content only if a DocuSign account has been configured by the customer.
 - {#IF_NOT_DOCUSIGN_ACCOUNT_EXISTS}content{#ENDIF} includes the content only if the customer has not configured a DocuSign account.
 - {#IF_DOCUSIGN_STAFF_SIGNING_ENABLED}content{#ENDIF} includes the content only if the customer has not disabled staff signing for the document template.
 - {#IF_DOCUSIGN_STAFF_SIGNING_DISABLED}content{#ENDIF} includes the content only if the customer has disabled staff signing for the document template.
 - There is a new “Omit from DocuSign” section property that, when enabled, prevents the section from being sent to DocuSign. This option can be overridden by the system administrator.
 - The DocuSign email blurb defined in the document template DocuSign configuration setup can now use macros such as {FirstName} and {LastName}.
 - In the DocuSign configuration, signer roles previously known as “Non-Staff Signer Roles” are now labeled as “Other Signer Roles” and there is a new checkbox option for these roles to indicate whether they route as staff or as non-staff.
 - Named blocks (i.e. defined with the {#BLOCK-blockname} and consumed with the {>blockname} directive) can now be defined in repeating sections and then used in other repeating sections based on the same child template.

- Documents now support a `{=^profileformula}` syntax that allows embedding of a formula directly in the context of the profile (i.e. student profile for a student document). This directive will render as expected when using the print blank document feature, but otherwise this syntax should be used sparingly both for performance reasons and to preserve the historical nature of the document.
- Configuration Management Tool
 - [New in 20.11.4.0] A CMT release script can now be manually modified to insert additional logic needed to handle certain special cases, as follows:
 - To conditionally execute certain lines if a condition based on the globals profile is true, wrap the lines in conditional logic as follows:

```
if (Versioning.IfGlobalsFormulaIsTrue("globals_conditional_formula")) {  
    conditional changes go here  
}
```
 - To preset the behavior override (that the ADMIN normally sets) to a specific true/false value, insert a statement like this:

```
Documents.SetDocFieldBehaviorOverride("document_template_id",  
    "behavior_field_name", true/false);
```

What's New in Version 20.6

- **General improvements:**
 - [New in 20.6.3.0] You can now view a report of all custom model elements in a database by selecting Administration > Configuration > Profile Types > More > Show All Customizations.
 - [New in 20.6.2.0] In the field properties of a document field or profile field, you can now change the data type itself in the following additional ways:
 - An integer or numeric field can now be directly changed to a character field without data loss. For example, the number 99.9 becomes “99.9” as characters. This is not reversible in the sense that a character field cannot be changed directly to an integer field (although you could create a separate integer field and copy/convert the data into it).
 - A date and time field can now be changed directly to a date field. In this case, a warning appears on the screen that there could be data loss of the time component. Note that the system previously supported changing a date field to a date and time field, which never involves data loss.
- **Documents Templates:**
 - [New in 20.6.4.0] Additional CMT Transformation syntax for document data are now available that allow you to specify a specific child template source to insert or update (grand)child documents from. See the “New in 20.6.4.0” references in the “CMT Data Transformations” section on page 381 for details.
 - [New in 20.6.4.0] **New Document/Section Action:** Introduced a new data modification method named “Modify Document Data Via Script” that supports a data modification script with one or more statements with the same syntax as “CMT Data Transformations” scripts, but with the modifications automatically restricted to the current document. This is intended to support advanced data transformation scenarios within a document that are not covered by other types of document/section actions. The “Configuring Section and Document Actions” section on page 215 has a detailed example of its use.

- [New in 20.6.3.0] To support a new and more robust approach to snippets which will culminate in a major new feature set called “Easy Edit” in the future version 20.11, the system now automatically injects snippet-id attributes into key HTML tags of document section formats, but only if the database is on a specific model version. When you create snippets for document templates with 20.6.3.0, there are now three main choices: Beginning of Section, Tag with Snippet ID, and End of Section. Only “Tag with Snippet ID” is new, and when you select this choice, you can enter the target tag’s snippet ID attribute value and select one of the following actions:
 - Insert Above Outside:** Inserts the new snippet content above the outside of the target tag. For example if the target tag is `<p snippet-id="p-21">contents</p>` and you specify the snippet content as a new p tag, your new p tag will be inserted above the target p tag (not inside of it).
 - Insert Above Inside:** If the target tag is `<p snippet-id="p-21">contents</p>` the snippet content you specify will be inserted as `<p snippet-id="p-21">(your snippet)contents</p>`.
 - Replace:** If the target tag is `<p snippet-id="p-21">contents</p>` the snippet content you specify will be used as a replacement : `<p snippet-id="p-21">(your snippet)</p>`.
 - Insert Below Inside:** If the target tag is `<p snippet-id="p-21">contents</p>` the snippet content you specify will be inserted as `<p snippet-id="p-21">contents(your snippet)</p>`.
 - Insert Below Outside:** Inserts the new snippet content below the outside of the target tag. For example if the target tag is `<p snippet-id="p-21">contents</p>` and you specify the snippet content as a new p tag, your new p tag will be inserted below the target p tag (not inside of it).
- [New in 20.6.2.0] In the field properties of a document field, you can now change the data type of a field in the following additional ways:
 - Integer and numeric fields can now be directly changed to character fields without data loss. For example, the number 99.9 becomes “99.9” as characters. This is not reversible in the sense that a character field cannot be changed directly to an integer field (although you could create a separate integer field and copy/convert the data into it).
 - Date/time fields can be directly changed to date fields noting that data loss of the time component of the original field will be lost.

- [New in 20.6.2.0] If you add a staff field named “DocumentCreatedBy” to a document template, the system will automatically recognize this naming convention and populate it automatically with the staff user who created the document (including retroactively for existing documents). If a document is created by the ADMIN or CONSULTANT, the field will contain EMPTY. Such a field may be useful for reporting or notifications. An alternative is to create the DocumentCreatedBy with the data type character (length=100), and the system will capture the user name and ID in the character field (ADMIN and CONSULTANT included).
- Introduced a new revision document option labeled “Allow Revising Current and Past School Year Documents Only” that appears on the document template properties screen. This option defaults to true but can be turned off for document templates that may require documents more than one school year old to be revised (typical for programs outside of Special Education). This property can also be overridden by the system administrator.
- Introduced a new document field property named “Audit Value Changes” that defaults to true for all non-calculated document fields but can be turned off for certain fields to make the audit log less noisy.
- Introduced a new field properties tab to the document template configuration screen which allows certain field properties to be set for multiple fields at once in an intuitive fashion. The field properties settable in this fashion are as follows: Required, Stylized Text (*long text fields only*), Reset Value If Section Not In Final Document, Prevent Manual Copying From Other Document, Retired, Copy From Preceding Workflow Document, and Audit Value Changes. For more information on these field properties, refer to “Adding Fields to a Template Section” on page 180.
- The system administrator (ADMIN) may also use the new field properties tab to override certain field properties. The overrideable field properties are as follows: Required, Auto-Notify to Review Acknowledge Document, Auto-Notify When Document Set to Review, Auto-Notify When Document Set to Final. If it is desirable to block the ADMIN from overriding any of these properties for particular fields, it is possible for the CONSULTANT to lock those properties from the individual field properties screen.

- Security rights can now be assigned directly to document template categories such that each document template in the category inherits rights from the category. For example, if a category has 10 document templates and the security access is the same for every document template in the category, the effort to assign and maintain the rights will be one-tenth of what it was previously. The following rules apply to document templates inheriting rights from the category:
 - a. Any rights assigned at the category level are automatically applied to each document template in the category, although if there is a document template in a category that needs to have distinctly different rights than other document templates in the same category, it is possible to disable inheritance for an individual document template via a “Do Not Inherit Rights from Category” option described later in this section.
 - b. Any rights that have not been assigned to a category can be assigned directly to one or more document templates in that category. Overall, the actual rights applied to a document template may include those assigned to the category plus those directly assigned to the document template.
 - c. Upon upgrade to 20.6, existing rights will remain assigned to document templates to preserve existing access.
- **Profile Types:**
 - [New in 20.6.3.0] A new “Alert Type” attribute is now available for logical fields with the ‘profile tag’ attribute set and an associated icon. The ‘Alert Type’ property provides the icon with a behavior similar to Special Programs integrated alerts in PowerSchool SIS and Unified Classroom. When the end-user clicks the icon in the Special Programs user interface, a popup appears with a hyperlink to the document or student list of documents.
 - Individual profile field properties (view mode) and data flow report now show SIS integration export/write-back layouts.

What's New in Version 19.11

- **Configuration Tasks**
 - Configuration tasks now have four states (Development, Testing, Acceptance and Closed) as described in the “Configuration Task Workflow” section on page 54.
- **Configuration Management Tool (CMT)**
 - [New in 19.11.5.0] Standard reports under configuration management now support a prior names field to allow the CMT to handle the renaming of standard reports.
 - Profile update scripts now support a prior names field to allow the CMT to handle the renaming of profile update scripts.

- The CMT now synchronizes changes to profile update scripts when synchronizing a single configuration task.
- Corrected issue where CMT generates synchronization scripts that omits changes in placement definitions (added missing placement options to CMT UI).
- **Profile Types**
 - [New in 19.11.3.0] A child profile type can now have its primary sort field set to a character field.
 - In model databases only (database name ends with “MODEL”), the Consultant user can now maintain design notes for a profile type. From the main configuration screen for a profile type, select “View/Edit Design Notes” from the “Setup: More” dropdown menu.
- **Documents**
 - There is a new simplified screen for creating and editing the properties of “file attachments only” document templates.
 -
 - for document templates now support “self-anchoring” snippets which are snippets that can be introduced into the main HTML format based on patterns in the main HTML format and without the need for pre-specified snippet directives. Self-anchoring snippets can be used to replace content in the main HTML format, whereas previously snippets could only be used to insert new content.
 - In model databases only (database name ends with “MODEL”), the Consultant user can now maintain design notes for a document template. From the main configuration screen for a document template, select “View/Edit Design Notes” from the “Setup: More” dropdown menu.
- **Events (profile > events tab)**
 - [New in 19.11.3.0] Custom event fields now have new checkbox properties:
 - Filter Field: Available for keyword and logical fields. When a field has this property, a filter dropdown appears in the profile > events tab allowing end users to filter the events on that field. For keyword fields, the dropdown will automatically filter out keywords where “InUse” column is set to false or alternatively a “Retired” column is set to true.
 - Retired: Fields with this property no longer show when manually creating new events, but the data is maintained in existing events.

- **Keyword Tables**

- Keyword tables that allow custom keywords can now be synchronized with a PowerSchool SIS “code set” (SIS version 12.1+ only) by identifying a code set name in the keyword table’s properties. Once this property is set, a “Refresh from SIS” button appears in the keyword table view when the database is integrated with a SIS instance. Any new keywords introduced by refreshing from the SIS code set are automatically set to custom since they were not part of the original model. Any already existing keywords that match items from the code set will be unchanged. The code set name property will be synchronized by the CMT just like any other keyword table property.

What's New in Version 19.4

- **Model Versioning**

- Model versions are automatically converted to the new standard PowerSchool format given below. Note that the feature segment of the schema is set to zero while upgrading since that was not in the model version format used prior to 19.4.
<state>_<year:YY>.<feature:0-12>.<maintenance:0+>.<iteration:0+>.

- **General**

- **[New in 19.4.1.0]** For profile and document fields, the system now supports field descriptions up to 3000 characters. Moving forward only, the system now also tracks the date fields are created and retired (date when retired flag is enabled).
- **[New in 19.4.1.0]** Pronoun support for non-binary gender identities has been introduced. Pronoun macros such as { .He,She } can now utilize non-binary gender identities. For backward compatibility, the macros themselves are the same as before, but such macros can now output non-binary gendered pronouns. Additional gender identities may be added as rows to the gender keyword table, and new field/columns have been added to the keyword table to delineate the pronouns to be used for each defined gender identity. The newly introduced pronoun field/columns are as follows:
 - Objective him, her, etc.)
 - PossessiveAdjective (his, her, etc.)
 - PossessivePronoun (his, hers, etc.)
 - Reflexive (himself, herself, etc.)
 - Subjective (he, she, etc.)

- **Documents**

- [New in 19.4.4.0] Introduced a new modification action for document actions labeled “Insert Revision Document” that can be used to automate the process of creating a revision/amendment document.
- [New in 19.4.3.0] Introduced a new {>namedblock} directive that integrates a read-only clone of a named block of content marked in another source section within the same document template. The named block is marked in the source section via block directives as follows:

```
{#BLOCK-<name>}  
content  
{#ENDBLOCK}
```

These directives will only work in the HTML format of a non-repeating section. Blocks cannot be nested. In this context, “read only” means that fields will not be editable in the cloned copy of the content.
- [New in 19.4.2.0] A PDF page header and/or footer format can now be configured for a document template and is used when documents from the document template are generated in PDF format. To configure this, go to the relevant document template and select “Edit PDF Header/Footer” from the topmost More dropdown. If the menu option is not there, a PDF header/footer may already be defined, in which case you can view it by selecting PDF Header/Footer from the section flyout menu. The PDF header and/or footer format can include directives and supports a ### placeholder for the page number.
- [New in 19.4.2.0] File fields in document templates now have a new checkbox property labeled “Show File as Read Only File Attachment”. When this property is enabled, the file stored in the field will also show as a file attachment just like any other file attachment with the exception that the file can only be modified or removed via the file field. This property is only available in the context of the top level document template and not in child templates.
- [New in 19.4.2.0] A new “SIS/UC Alert” property has been added to document templates. This is used to establish routing from alerts in SIS or UC to specific documents in PowerSchool Special Programs. Note that more than one document template may share the same alert, in which case, a user clicking that alert in SIS or UC will route to the latest “active” document from any document template routed to that alert, or if no active document is found, then it will route to the latest non-active document restricted to the current or prior school year.

- [New in 19.4.2.0] Document/Section actions with the “Send Message” data modification method previously required a comma separated list of staff profile reference fields or expressions identifying staff to receive the message. Now there is an alternate approach of simply specifying an asterisk (*) wildcard to include all staff referenced in any staff profile reference field with exceptions defined by those staff profile reference fields that have the “No Staff Notification” property. If no child template is selected in the document/section action, the message will be sent to all eligible staff fields even if embedded in nested child templates. However, one can select a specific child template to limit the scope to staff fields in a particular child template. Note that if a specific staff user is referenced in more than one such field, that staff user will only receive one message per document/section action. In some cases, it may be desirable to only send a message via a particular staff field if other conditions are true. If one is configuring staff expressions, one can do this by including IF(<condition>, stafffield) as an expression. But if using the * wildcard, one can accomplish the same thing by marking the staff field with the “No Staff Notification” property, but then creating a corresponding calculated staff field that includes the condition and which does not have “No Staff Notification”.
- [New in 19.4.1.0] It is now possible to configure a date range (minimum and/or maximum date) for a date or date/time field in a way that provides immediate client-side validation in the textbox and calendar popup, and server-side validation as a backup. Previously date range validation could be done using a section action, but section actions do not provide immediate client-side validation. To use this new approach, configure a calculated date field named <datefieldname>_MinRange and/or <datefieldname>_MaxRange where <datefieldname> is the name of the date or date/time field being validated. The _MinRange and/or _MaxRange field must be configured as date fields (not date/time fields) and typically have the “Internal Value” property set. The validation message that is shown to users when a date is out of range is taken from the description property of the _MinRange/_MaxRange field, which may be localized.
- Document templates now support a “View Final Only” document-wide right. Users who have this right can view final documents but not draft/review documents. A typical use case is when a school district wants to give external lawyers or auditors access to final documents, but not draft documents.

- **Formula Language**

- The formula language now supports an advanced NthVALUEOF function that accepts the same parameters as the TopOneValueOf function but with an additional parameter that allows one to obtain the Nth value. The additional parameter is zero-based, so if it is zero, that is equivalent to the TopOneValueOf function. If the additional parameter is one, then the second value is obtained.

- **Profiles**

- **[New in 19.4.2.0]** Profile constraints have been enhanced so that child profiles can be updated or deleted by a profile constraint configured for the corresponding top-level profile type, and which utilizes criteria and expressions referencing both the top level profile type and the target child profile type. Additionally child profiles can be updated or deleted by a profile constraint configured for a different child profile type that shares the same top-level profile type, and which utilizes criteria and expressions referencing both the top level profile type, the target child profile type, and the context child profile type.
- **[New in 19.4.2.0]** Additional usage information is now shown when viewing a profile field properties detail view, including what calculated fields may be referencing the field being viewed.
- **[New in 19.4.1.0]** The system now supports additional Canadian alerts and allows the relevancy of alerts to be specified at the model level. Additionally, PowerQueries and SIS field declarations can now be specified at the model level.
- Profile update scripts now support updating a top level profile type directly from a child profile type. This can be used, for example, to update flat contact fields in the student profile from the student contacts child profile type.
- The ordering of top level profile types and child profile types (within top level profile types) can now be explicitly set in the model. Changing the order of top level profile types affects the order in the “Search” menu. Changing the order of child profile types affects the ordering in the profile flyout menu.
- There is a new type of profile constraint action that can be configured to insert a new child profile from the context of either a top level profile or another type of child profile that shares the same top level profile. When inserting a child profile from a top level profile, the fields can be set from the values of the top level profile. When inserting a child profile from another type of child profile, the fields can be set from values of either the originating child profile and/or the top level profile.

- In profile forms, the selection for a profile reference field to a child profile can now be embedded inline in form rather than as a lookup. In a profile form for a top level profile, one can configure a profile reference field to a child profile of the current top level profile. Similarly, in a profile form for a child profile, one can configure a profile reference field to a different child profile type that shares the same top level profile. In either case, the profile reference field by default appears with a lookup link that opens a model dialog box allowing the user to select a particular child profile. In some cases, it may be desirable to present the selection embedded inline in the form itself so that the selection is always visible, and this is what is new in 19.4. An example would be a service record that has a field named “Mandate” that references a different “Mandates” child profile. In this case, the directive {Mandate:Z“inline”} presents the selection inline in the form. If you would like to have control over the size of the inline selection in the form, use a format like {Mandate:Z“inline,800-400”} to specify the width and height of the area. If the inline selection is dependent on other fields on the same form that may dynamically change, you can programmatically cause the inline selection to refresh using the DataFormAPI.invokeLookup method.
- It is now possible to configure a constraint to control/restrict whether particular students can be added to a student transfer envelope. A typical example is a requirement that a parental consent date be filled before student information can be sent. Configure a logical field (typically calculated) in the student profile named CanSendInTransferEnvelope that is true if the student can be sent or false if not. The description of the field should be set to the preferred message to the end user if the field is false and the student cannot be sent. For example, the logical field formula might be “TransferParentalConsentDate IS NOT EMPTY” and the description might be “Parental consent has not been given.” Note that the field description can be localized into other languages.
- **Integration Enhancements**
 - A “Student Contacts” special child profile type compatible with PowerSchool SIS student contacts can now be added to a model with a few clicks. It automatically comes with a “detail” form and import layouts to populate student contacts from PowerSchool SIS.
 - When adding the StudentGuardian child profile type used in SIS student/parent portal integration, the import layout will now be automatically added.

What's New in Version 18.1

- **General Configuration Enhancements:**
 - Character fields can now be up to 400 characters.
 - Keywords in keyword tables can now be up to 30 characters.
 - The CMT now synchronizes all “prior names” properties.
- **User Help Resources:**
 - User help resources can now be marked as “under configuration management”. The CMT will not only synchronize help resources under configuration management.
- **Document Templates Enhancements:**
 - A document template can now be written directly in another language and tagged to that language so that correctly translated keywords show in documents created from that document template. Such document templates do not allow “translation” because they are directly in another language with no translation needed.
 - Document and child “templates now support an additional progress monitoring property labeled “Include Data Point Comments”. When enabled, an additional column is presented to the end user to allow the user to specify a comment for each progress monitoring data point. The comments do not show on the progress monitoring graph.
 - [New in 18.1.0.501] There is a new property associated with document-based progress monitoring that allows overriding the standard header of the “Strategy/Intervention” column for data points. The new property is labeled “Strategy/Intervention Column Header”. Translation/localization of this header is supported.
 - [New in 18.1.0.501] DocuSign text box fields now support multi-line word wrapping.
- **Integration Enhancements:**
 - [New in 18.1.0.501] In the context of eSchoolPlus integration, PDF Streaming can now be controlled by document configuration (e.g. program participation). If a document template has a logical field named “CanStream” (which will typically be a calculated field), then the document that would otherwise stream will not stream. A custom logical field named “_CanStream” will be also recognized and used for the same purpose regardless of whether “CanStream” exists or not. If both exist, “_CanStream” will take precedence.

What's New in Version 18.0.1

- **Profile Enhancements**

- [Version 18.0.1.501 or later] Profile constraints now support an action of “Set Field Values in Referenced Profile” which allows for setting field value(s) in a profile that is referenced directly or indirectly from the current profile. For example, if there is a staff profile reference field in the student profile, a constraint can now be configured to set field values in the staff profile. The “target profile criteria” is typically just the name of a profile reference field.
- [Version 18.0.1.501 or later] Keyword table character fields now support Unicode property. Translation columns set up in the future will now default to Unicode.
- [Version 18.0.1.501 or later] Profile character and long text fields now support Unicode property.
- Profile section names now support a “Prior Names” property that allows a section to be renamed by the CMT (rather than deleting under the old name and re-adding under the new name).
- Profile section names can now have a portion of the name that does not show for end users, specifically anything including and following the first tilde (~) character. This may be useful for having different versions of the same section present to end users under the same name (e.g. Demographics ~ Admin).

- **Document Configuration Enhancements**

- Document DocuSign integration supports mapping signer fields back to field values. This was not supported in earlier versions.
- The “Verify All” check now detects cases where stylized text long text fields data flow into non-stylized long text fields, which is not advised.
- Document templates that have the “Allow Bulk Document Creation” document template property can now be bulk created from a list report that has the following options set in report properties:
 - a. Report Is Under "Configuration Management" Control
 - b. Enable Users to Apply Data Modification Constraints via This Report
- Document/section actions that send a message to specified staff can now be configured to send a message to staff in repeating rows/sections (child documents).
- Document template section names can now have a portion of the name that does not show for end users, specifically anything including and following the first tilde (~) character. This is very useful for retiring sections in such a way that they continue to have a unique section name for the CMT but continue to show to end users under the old name (e.g. Goals and Objectives ~ Retired May2017).

- It is now possible to configure a document template such that a document can be printed specifically for a parent/guardian with certain content added or removed, and similarly for a student. When a student/parent/guardian user views the document through the student/parent portal, the user will view the appropriate content as well. See the “Printing Special Document Content for Student/Parent/Guardian” section on page 312 for more details.
- [Version 18.0.1.501 Only] A document action can now be configured to send messages to staff referenced in grandchild templates.
- **Configuration Management Tool**
 - Document transformations are now supported in cases where data has to be migrated from retiring field structures to new field structures. See “CMT Data Transformations” on page 381 for additional information.

What's New in Version 18.0

- **Profile Configuration Enhancements**
 - In earlier versions, for a profile reference field, the ability to introduce a “non-lookup” was limited to documents. Now it is available in profiles as well.
- **Document Configuration Enhancements**
 - In a repeating row layout using the “fast add” style configuration, it is now possible to supplement the add/plus icon with a text label using a directive like {^EDITCONTROL:L"Add Item"}.
 - In certain circumstances, you may want to stop progress monitoring (stop the collection of data points while continuing to show the already collected data points and chart). To do this, use a construct as follows:

```
{#IF ProgressMonitoringStopped}
  {&ProgressMonitoring:R}
{#ELSE}
  {&ProgressMonitoring}
{#ENDIF}
```

The {&ProgressMonitoring:R} includes progress monitoring in read only mode only.

- DocuSign document integration can now be configured to allow a top level signer role to sign every instance of a repeating section.
- **Data Form API**

- A new function (DataFormAPI.getFieldKeywordTableValues) to obtain the keyword table row values associated with a keyword field has been introduced. See the “JavaScript” section for more details.

What's New in Version 17.1

- **Data Dictionary Configuration Enhancements**
 - Logical fields can now be changed to keyword fields by changing the data type. However, the change is only accepted if the keyword table has a column that maps the old logical values to the new keywords. The keyword table must be named “OldValues and have a character data type. The keyword row that logical false values will map to should have the value “No” or “False” (case insensitive) in this column, and the value “Yes” or “True” for logical true values. The Configuration Management Tool has full support for such data type changes.
 - Character fields can now be up to 300 characters long.
- **Profile Configuration Enhancements:**
 - It is no longer necessary to configure profile form section field associations. Such associations are now maintained automatically.
 - Snippets are now supported for profile section forms. They word identically to snippets for profile sections.
 - SVG format is now supported for profile tags.
- **Document Configuration Enhancements:**
 - The “Verify All” function of document templates now catches directives that are likely to prevent the translation engine from working as expected. After a section format, the same warnings will be immediately displayed. For more information, see the “Best Practices to Support Translations” section on page 317.
 - Document template properties for revision documents now have a new checkbox property labeled “Disable Latest Finalized Document Validation”. By default, revision documents can only be created from the latest finalized document. To disable this default behavior, enable this new option.
 - In certain scenarios, it may be useful to have a keyword field dropdown next to a second editable field, but then upon saving, the value of the second field is displayed embedded in the middle of the keyword description. The new “Z” modifier for the keyword field directive makes it possible to embed macros in keyword descriptions that get evaluated when the form is not in edit mode. For example, a keyword field named “Measure” has a keyword description of “Student will score between {MinScore} and {MaxScore}”, which includes macros for MinScore and MaxScore. The markup for Measure, MinScore, and MaxScore fields can be something like

{Measure:TZ}{#IFEDIT}{MinScore:T},{MaxScore:T}{#ENDIF}. In this fashion, the MinScore and MaxScore fields only appear as separate fields in edit mode. In view mode and due to the “Z” modifier, the MinScore and MaxScore field values appear embedded in the keyword description.

- A new “U” field directive modifier is now available for document character fields (only) to support situations where the value is editable in the section but must be dynamically displayed and updated elsewhere (read only) in the same section. There are three specific scenarios where this is supported:
 - 1) A top level character field is editable in a section but the value must be dynamically displayed in all repeating rows on the same section. In the repeating row, the outer field is referenced via {FieldName:U} where U sets up the dynamic updating. When the value of the editable field is changed, the values are automatically updated in each repeating row when the editable field loses focus.
 - 2) Similarly, in a repeating section, a character field that is editable may be displayed in each repeating row nested within the repeating section.
 - 3) A field is editable in the section but the value must be mirrored in another place on the same section, but not in repeating rows. In this case, the mirrored field should be marked with both R (for read only) and U (for automatic updating).

- **Configuration Management Tool (CMT)**

- Model versions are now explicitly supported in the configuration. To establish the model version, go to Configuration > Profile Types > More > Set Model Version (after selecting a configuration task). Model versions have the format <ALPHA>.<year:YYYY>.<month:1-12>.<iteration>, for example, NY.2017.11.0. The CMT recognizes the model version, and when generating a script that represents a model version upgrade (e.g. NY.2017.11.0 to NY.2017.11.1), the generated script will verify that the target database is on the old model version and then set the new model version on the target database after making all changes.
- The CMT now issues warnings when an attempt is made to synchronize from document template or profile sections that are locked for configuration.

- The CMT now supports profile data updates and transformations, which is sometimes necessary when introducing new fields that replace existing fields. Such transformations must be inserted manually into a generated CMT script by inserting a line at the appropriate point. The inserted line should have the format given below where <inlinescript> has the same syntax as a profile update script.

+Profiles.ExecuteInlineUpdateScript("<inlinescript>");

Note that inserting a new line into a script adds an additional step, and to account for that, the “steps” number in the following line near the beginning of the script should be manually incremented for each step manually inserted.

Initialize(steps: 36);

What's New in Version 17.0.1 (Service Pack #1)

- **Data Dictionary Configuration Enhancements**
 - Character and short text fields can now be changed to long text fields by simply changing the data type property. The Configuration Management Tool has full support for such data type changes.
 - Larger icon sizes are now supported for profile tags. Previously, only 16x16 images were supported. 24x24 icons are now supported also.
- **Document Configuration Enhancements:**
 - Long text fields have a new “Auto Save” property. When enabled, as soon as the user leaves the field, its value is immediately saved, accompanied by a visual effect. Note that auto-save does not work in repeating rows, but does work in repeating sections.

What's New in Version 17.0

- **Profile Configuration Enhancements**
 - (Introduced in 17.0.0.503) Profile grids embedded in profile forms can now have a checkbox column with a button in the column header that applies a specified profile constraint to any profiles in the grid that have been checked. See the “Profile Grid Directives” section on page 345.
- **Document Configuration Enhancements**
 - Document text fields with the “Stylized Text” option can now be configured with an additional “Allow Graphics in Stylized Text” option to show an icon to import graphics into the stylized text.

- There is a new document template option labeled "Enable Streaming to SIS". Currently, this only affects the eSchoolPlus SIS integration and allows documents from the document template to be streamed to the SIS.
- The document field property option labeled "Use for Document Action Macros" is now labeled as "Always Available for Macros". When applied to character fields, this option allows the field to be used as a macro anywhere in the document including in any statement banks. When editing statement banks, the field macro for the character field will appear at the bottom for easy insertion.
- A document action can now be configured to update all grandchild documents in a document (from a particular grandchild template).
- There is a new type of document action modification entitled "Populate Child Documents from Keyword Table" which creates top level repeating rows from a keyword table. It requires that the child template have a keyword data field with the "Child Document Title" property.
- Document actions that have a modification action of "Populate Child Documents from Other Document" now support an "Include Grand Child Documents" checkbox option. If enabled, when the top level child documents are populated, the action will carry along with them any grand children documents constrained to when the grand child templates match by template identifier and there is at least one field that matches by name and data type.

What's New in Version 16.1.2 (Service Pack #2)

- A new property in document templates and child templates allows specification of a minimum "in-session" day interval between progress monitoring score dates. For example, if the number 3 is specified in the property, users will be required to enter scores on dates at least three "in-session" days apart.
- There are new features to improve the experience of editing HTML formats:
 - To save an HTML format while editing, type Ctrl-S.
 - From the HTML editor, to see details of a CSS class name, double-click the class name to highlight it and then type Ctrl-period.
 - For named consultants only: From the HTML editor, click "Fragments" in the toolbar to access, create and share reusable named fragments of HTML formatting that can be accessed across document templates, across databases and even shared with other named consultants. The owner of a fragment specifies whether it is for the current database or any database, and if it will be shared with other named consultants.
 - When editing an HTML Format, there is a "Style Sheet..." button in the toolbar. Clicking this launches a popup window allowing the viewing of the style sheet. From this window, the toolbar button labeled "Select CSS Class" changes the mode of the page listing out simple class names for easy selection. Clicking a style then inserts the name in the HTML format and closes the popup window.

- A new document template option labeled “Suppress Field Description Tool Tips” allows all field description tool tips to be suppressed for a document template. The option can bulk set for multiple document templates at a time.
- For DocuSign integration, it is now possible to configure “other signers” based on formulaic values in the context of child templates.
- When configuring a repeating row layout, there is a new checkbox option associated with the filter formula that is visible in very specific situation where child template A is displayed as repeating rows in a repeating section based on child template B but child template A is a sibling rather than a child of child template B. This checkbox is labeled “Join Repeating Section Child Template” and allows for a join expression between the two child templates, which again do not have a parent child relationship.
- The user caseload customize column screen displays student profile forms with as many fields as possible in the form of checkboxes. In some cases, this may result in unwanted content appearing on that screen. The new directive syntax below can be used to suppress such content on that screen and similar screens:
{#IF_NO_PREVIEW}content to hide{#ENDIF}

What's New in Version 16.1.1 (Service Pack #1)

- Child document fields can be added directly in the context of repeating row layouts.
- Keyword tables can now be translated using Google Language Translation based on the Description or Name column.

What's New in Version 16.1

- **Document Actions:** There is a new document action data modification labeled “Clear Active Status in Other Document” which allows for clearing the active status from the active document in the specified document template.

What's New in Version 16.0

- **Profile Enhancements:** A “student at a glance” flyover can now be configured to present key summary data on students to users viewable from their user home page. The flyover can be differentiated by role, and can be configured to show the most critical status information that role users need to see, such as IEP due date, reevaluation due date, etc. A flyover is configured simply by adding a new profile form section with the appropriate content and setting the section property labeled “Purpose of section” to “Flyover”.
- **Keyword Table Enhancements**

- When editing a keyword table, there is a new “Set Auto-Sort” option which allows specification of one or more sort columns for the keyword table with support for ascending/descending order on any column. Once auto-sort is in place, new keyword rows are always simply added and then automatically set to the correct position to maintain the sort.
- **Document Template Enhancements**
 - There are several new features designed to allow an additional level of self-customization by system administrators without deviating from the model database (i.e. state model). The same features can also be used by configurators to tailor specific databases, again without deviating from the model database.
 - A configurator can mark specific document and section actions to allow them to be disabled by the system administrator. The option to allow disabling is synchronized by the Configuration Management Tool, but the actual disabling by the system administrator is not.
 - A new feature called snippets has been introduced to allow document section formats to be tailored by the system administrator (ADMIN) or CONSULTANT without deviating from the model database. Snippets are fragments of HTML formatting that can be inserted at specific points in the existing HTML section format. Snippets typically reference “custom” document template fields (fields marked as custom in the data dictionary for the document template). An important constraint of snippets is that they can only be inserted at the beginning of a section, end of a section, or at named insertion points in the model section format (identified using snippet directives). If the model database is well-equipped with desirable insertion points using snippet directives, customers will be able to get the most from this form of custom tailoring. For more details, see the “Snippets” section on page 241.
 - Document sections that have deviated from the model database entirely can be marked as custom. Custom sections are not synchronized by the Configuration Management Tool.
 - There is a new document template configuration option labeled “Allow Statement Banks by Organizational Location”: If this option is enabled, then statements banks can be created and maintained for specific organizational locations (e.g. locations, districts, etc.). At the current time, only the outer organizational level is supported. For smaller school districts, the outer organizational level is typically “locations”, therefore statement banks associated with specific locations would be enabled by the option. For counties and integrated school districts, the outer organizational level is typically “districts”, therefore statement banks associated with specific districts would be enabled by the option. To authorize a security group to edit statement banks associated with

their organizational location and not the system-wide statements bank, assign the “Edit Public Statement Banks” system administration privilege to the appropriate security group at the relevant organizational level (as shown in the first screen snippet below). Then assign the corresponding document template right labeled “Edit Public Statement Banks” for the specific document templates you wish to authorize for that security group. Note that this template option is overrideable by the system administrator, so it may be preferable to leave the option off in model databases and let the system administrator enable it when needed.

System Administration Privileges	
Manage Staff Security	n/a
Manage Student Parent Security	n/a
Manage User Sessions	n/a
Access Audit Log	n/a
Log in as Other Users for Support	n/a
Edit Public Statement Banks	Grant (+) Location-wide
Receive Support Requests	n/a
Manage Student Logins for My Classes	n/a
Manage Organizational Calendar	n/a
View All Configuration	n/a
Translate Document Templates	n/a

File Published	Attach Files to Final Documents	Edit Public Statement Banks	Force Finalize
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- New document template configuration options have been introduced to support DocuSign integration. These options can be configured regardless of whether a database is equipped with a DocuSign account. This allows model databases to include configuration support for DocuSign integration even while the configuration is dormant until a DocuSign account is established. Please see the "Configuring DocuSign Integration" section on page 297 for details.

What's New in Version 16.0.1 (Service Pack #1)

- **Form Configuration Enhancements**

- In the directive for a keyword table dropdown menu, it is now possible to filter the keywords based on more than one keyword table field/column. The filter columns are comma-separated as in the sample directive below. This is especially useful when a keyword table has a column that tracks which keywords are currently active versus retired. Please note that this enhancement was released in 16.0.0.501. Earlier versions

will not recognize the syntax and will show an error.

{KeywordField:F"filtercolumn1,filtercolumn2,..."}

- **New {^parent_formula} syntax:** In a child profile form, a calculated field in the parent profile could be referenced through an embedded calculation such as {=Student.Age}. However, this calculation would not show when adding a new child profile because it is based on the current child profile that does not exist yet. The new *{^parent_formula}* syntax allows the formula to be written directly in the context of the parent profile such that it will be shown when adding a new child profile. In the example, {=Student.Age} would become {=^Age}.

- **Document Configuration Enhancements**

- There is a new child template property to data flow child documents to child profiles when the document is set as the active document.

What's New in Version 16.0

- **Profile Configuration Enhancements**

- Previously child profiles supported a logical “AllowEdit” and/or “AllowDelete” field that specify whether individual child profiles can be edited or deleted based on characteristics of the child profile. To round this out, a similar field is now supported to control adding new child profiles. However, the field must be in the parent profile type and named “AllowAdd<childprofiletypename>” where <childprofiletypename> is a placeholder for the plural form of the child profile type name. This field in the parent profile then controls whether it is valid to add child profiles of that type based on characteristics of the parent profile.
- A field set expression can now be configured in a child profile type that applies a field set expression to the parent profile whenever any child profiles are added, updated or deleted. Typically the field set expression, written in the context of the parent profile, will be based on “TOPONEVALUEOF” or other aggregate expressions that reference the status of the child profiles.
- There are new profile constraint actions to deactivate or reactivate a profile. This can be used to automate deactivation or reactivation of profiles. With an evaluation trigger of “bulk operation”, such a constraint can be used to deactivate or reactivate selected profiles in bulk via a list report.

- **Document Configuration Enhancements**

- **Document/Section Actions:** New data modification options have been introduced for document and section actions, as follows:
 - **Populate Child Documents from Other Document:** This enables child documents in the current document to be populated in by copying child

documents from a source document from another template. From a configuration standpoint, you choose the target child template and the source document template. The action requires that the source document template has a child template with the same ID. Only fields with the same name and data type will be copied. To select the specific source document, you specify join criteria in the format ThisDocumentExpression1=SourceDocumentExpression1 AND ThisDocumentExpression2=SourceDocumentExpression2...

- **Populate Child Documents from Top Level Profiles:** This enables child documents in the current document to be populated by copying data from top level profiles of a specified profile type. A join expression with the format “DocumentExpression1=ProfileExpression1 AND DocumentExpression2=ProfileExpression2” is used to identify the source profiles. You also configure a field set expression in the format “ChildDocField1=ProfileExpression1, ChildDocField2=ProfileExpression2,...” to specify the data that is written.
- **Insert Workflow Document Via Child Document Title Field:** This modification, which existed in previous versions, is now also supported in sections that are not repeating sections. In this newly supported scenario, you select the child template and the criteria for which child documents will launch workflow documents. For example, you could use logical field checkboxes in repeating rows to launch workflow documents (be sure to force them as read only once workflow documents are launched).
- **Set Document Active Status:** This modification sets the “active” status of a document to either “active” or “not active”.
- **Staff Document Reassignment:** Staff documents have a new template configuration option labeled “Allow Reassign Documents to Other Staff”. If enabled, a new document security right labeled “Reassign” becomes available for assignment to security groups (typically administrative security group(s) in this case). Users authorized with this right can reassign staff documents from the template to a different staff member for which the user has view staff profile privileges.
- The “Append Signature Images When Printed” template section configuration option has been enhanced. When the document is printed, the signatures of any users who have electronically signed the document will be appended at the bottom of a section with this option enabled. Note that multiple sections can be configured with this option, but when a document is printed, the signatures will be printed only once and at the end of the last printed section that has this behavior option. For example, if every section in the document template has this option, then the signature images will always be printed at the end of the document. Each signature will appear with three elements, the actual image of the user’s signature if available, the printed name, and the printed signature date. Text styling can be applied to the

printed name and date by defining a CSS class named DOCUMENTSIGNATURES in the CSS style sheet. The image of the user's signature can come from two sources: 1) image captured via Topaz signature pad, or 2) a staff image field named either 'SignatureImage' or 'Signature'. If no image is available from any of these sources, the printed name and date will still appear.

Organization of Data in PowerSchool Special Programs

Regardless of the hosting model utilized, each school/district has a separate and distinct database maintained for it by PowerSchool Special Programs. This design makes possible a substantial degree of customization for each school district. Each such database stores all data related to that district as shown below:

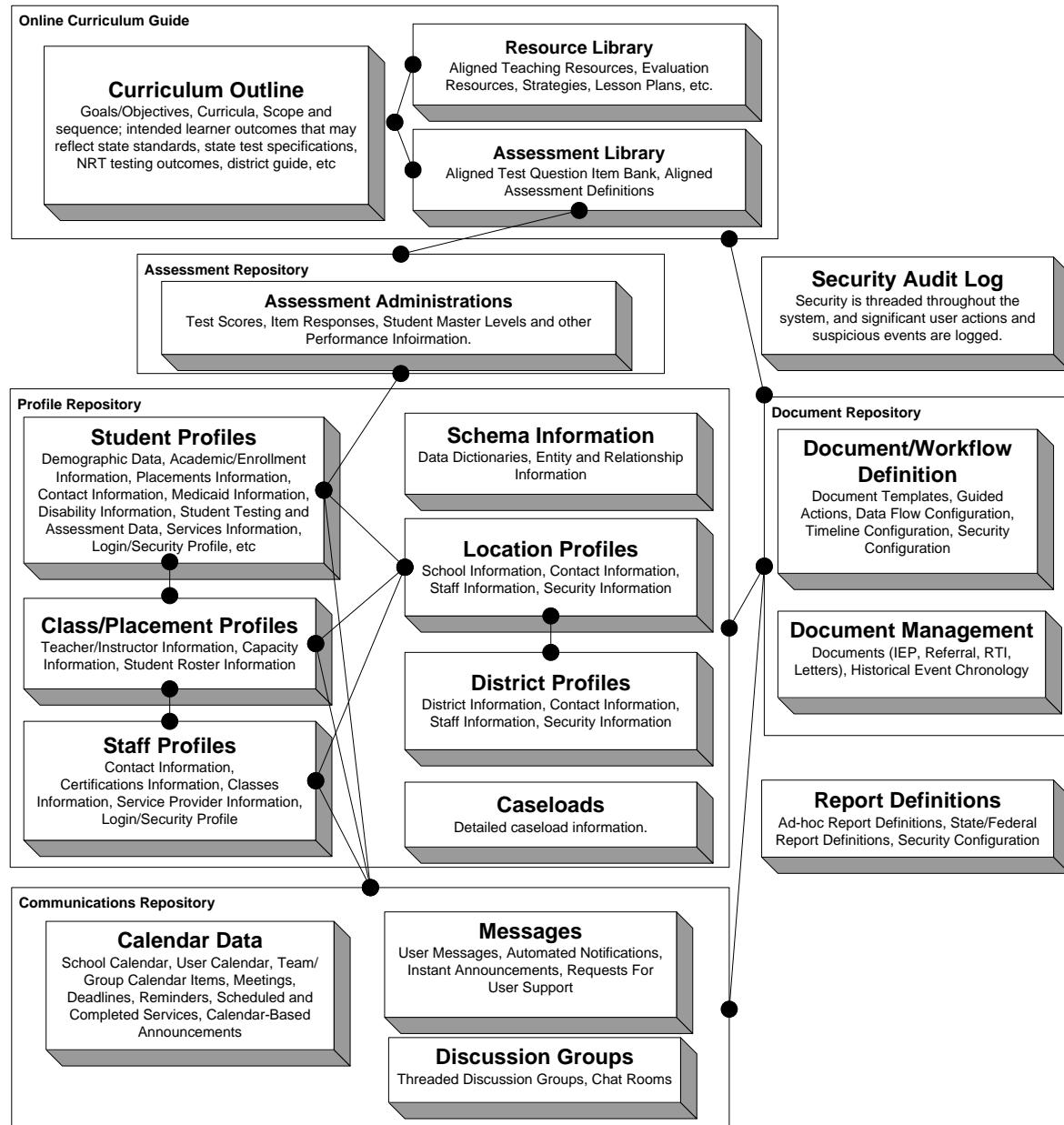


Figure 1A: Organization of Information in PowerSchool Special Programs

In the sections that follow, each major repository in the diagram above is described:

Profile Repository

Each school district database includes a set of profiles which comprise the basic filing system of data on students, staff, locations, classes and districts. Student Profiles contains demographic information, enrollment information, placement information, disability information, medical information, and information on services received. PowerSchool Special Programs also maintains Staff Profiles to track personnel information, Location Profiles to track school/building information, and District Profiles to track district information in scenarios where students are sent to or received from other school districts. Note that Profiles are distinct from “documents” such as IEP documents, although data will typically flow back and forth between profiles and documents using a mechanism called “data flow”.

The major types of profiles used in PowerSchool Special Programs are described in more detail below:

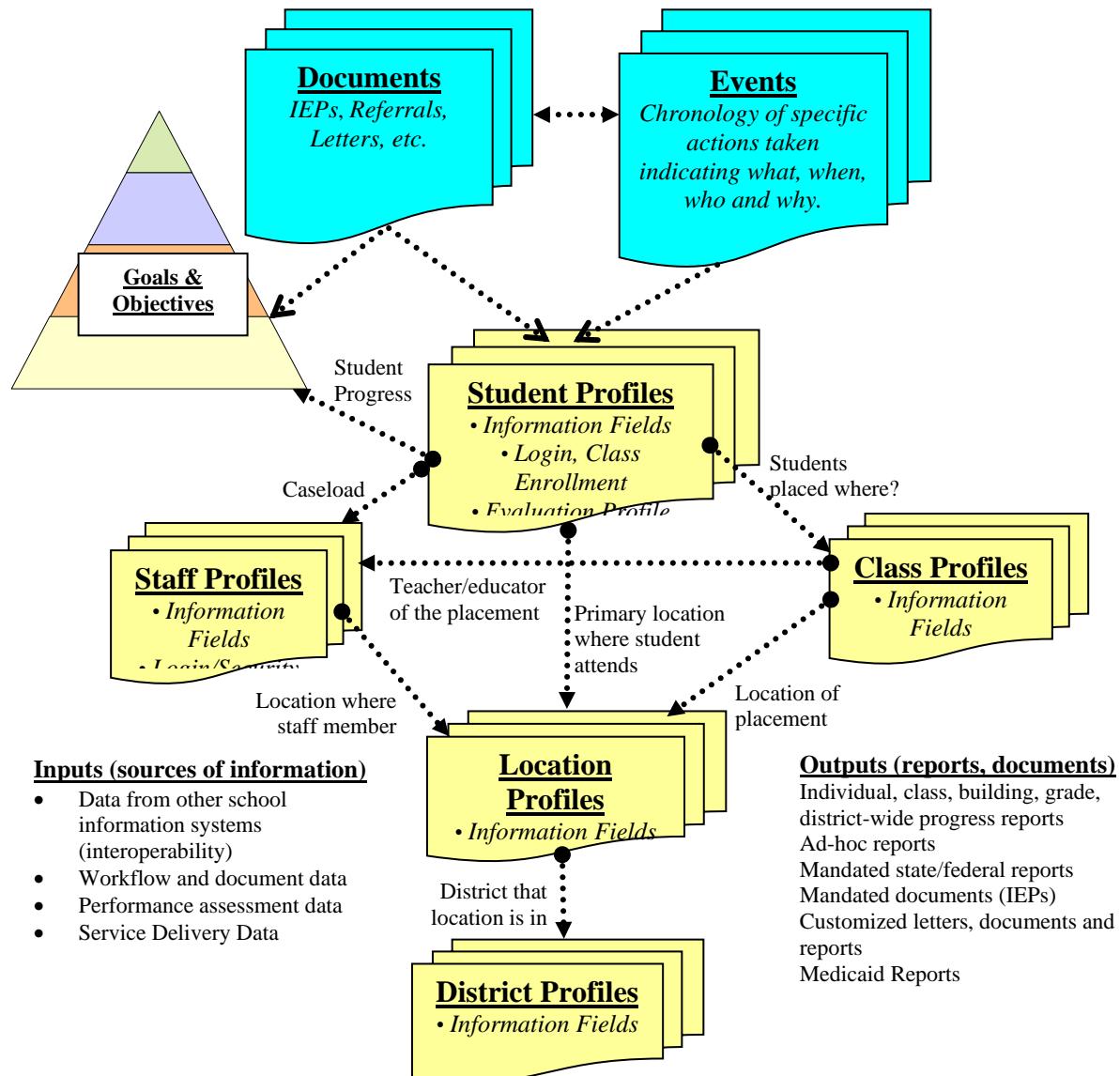
District profiles: Use of this type of profile is only necessary when students are sent to or received from other school districts for instruction or special services. District profiles contain fields that describe a school district such as district code, district name, and district contact.

Location profiles: PowerSchool Special Programs is designed to maintain a location profile for each school, building, or site where students can receive instruction. It contains fields that describe a location such as location ID, location name, address, phone number, and other fields that differentiate and describe each location. Every location profile in PowerSchool Special Programs must have a unique ID. Typically, the location ID is a building code of some kind. In situations where students are being sent to and received from other school districts, it is best to make the location ID a combination of a district code and a school code. Note: On the <http://nces.ed.gov/ccdweb/school/> web site, you can look up identifiers for public school districts and schools.

Staff profiles: There is one staff profile for each teacher, administrator, or other staff member involved in the instructional process. The staff profile contains security information (login, password, security group membership) and information fields such as name, phone number, staff position, work location, etc. Each staff member (teacher, administrator) must have an ID that is unique across the school district. If the school district already uses staff member codes, those can be used in PowerSchool Special Programs. Alternatively, the staff member’s initials can be used (to help ensure uniqueness, include the middle initial).

Student profiles: Generally, there is one student profile for each student enrolled in the school district. Each student profile contains demographic fields such as ID, name, sex, birth date; and academically-oriented fields such as grade level, class enrollment, services received, and enrollment start date. If student/parent logins are enabled, student profiles also contain security and login information for both the student and the parents of the student. Every student profile must have an ID that is unique across the school district. Many school districts have an existing system of assigning student IDs, and if so, the same system should be used in PowerSchool Special Programs. Do not use identification numbers that raise security concerns, such as social security numbers. If the school district allows students to log in to PowerSchool Special Programs, the students will log in by entering their student ID.

Class profiles: This type of profile is the basis for student enrollment in classes. A class profile represents an individual teacher teaching a particular content area to a particular group of students. It contains information fields that describe the class such as class name, teacher(s), program (subject area), building identifier, room number, and period. Every class profile must have an ID that is unique across the school district. Typically, the class ID encodes the content area, period and/or the teacher's last name or initials. If this is a district-wide installation, it is best to include the location ID in the class ID as a prefix. For example, if the location ID is '011', then the class ID of a period 2 math class taught by John Smith might be '011MA2JS'.



The figure above illustrates the types of profiles used in the Special Education Case Management system and the relationships between them. For example, the figure shows that each class profile references (is linked to) the staff profile of the person teaching that class. By default, teachers can only access the

students in their own classes (unless they are given additional privileges). Similarly, the student profiles, class profiles, and staff profiles reference the location profile of their respective school or site. By default, staff can only access information from their own location unless they are given district-wide access privileges.

Data Fields: A profile consists of a set of specific data fields such as student ID, name, ethnic designation, address, etc. To customize a database, it is useful to have a general understanding of data fields in a relational database. The basics are covered in this guide. The definitions of most data fields in a particular client database will originate from the state model database; however, a client database can be customized by adding additional custom data fields to meet local requirements. Additionally, PowerSchool Special Programs supports calculated fields defined using a formula. For example, “Age” would be a calculated field determined from the student’s birth date.

Security Profiles: Each staff profile potentially represents a user who can log into PowerSchool Special Programs. If a login is enabled for a staff profile, a Security Profile is created to store the password(s), security groups and additional privileges for that user. If allowed by the school district, logins can also be enabled for students, in which case those Student Profiles will have corresponding Security Profiles.

Profile Forms: The end-user utilizes profile forms to view, add, edit, and delete profiles. A profile form organizes data in a coherent, easy to understand way for the user so that they can view and edit the information. Typically there might be a form for basic student demographic information, another for parent contact information, another for services received, etc. The definitions of these forms are stored in the database. Developing forms requires a basic level of skill with data fields, formulas, and HTML (hypertext markup language). These skills are covered later in this guide.

Database Schema: The schema consists of the data dictionary that describes the structure of profiles as well as associated lookups, attributes, constraints etc. The schema can be modified by adding, modifying and deleting fields, lookup tables, profile types (entities), etc.

The screenshot shows a web-based application window titled "Profile Type Setup". The URL is <http://localhost:55381/TIENETD/profilitype.aspx?pt=2&sec=13&view=F>. The top navigation bar includes links for Search, Curriculum, Assessment, Communication, Reporting, Administration, and Sample School District. A "Logout" link is also present. Below the navigation, there are tabs for Profile Types, Students, and a setup section with options like Verify All, Add New Section, and More... The main content area is titled "Student fields linked to 'Demographics/Enrollment'" and displays a table with the following data:

	Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description
	ID	ID(10)	Quick Search	Key Reporting Fields	-	Contains the student's id number.
	LastName	Character(30)	Quick Search	Key Reporting Fields	-	Contains the student's name.
	FirstName	Character(30)	Quick Search	Key Reporting Fields	-	Contains the student's name.
	MiddleName	Character(30)	-	-	-	Contains the student's name.
	HomeSchool	Location Profile Reference	Organizational Parent Reference, Quick Search, Assessment Filter Field	Key Reporting Fields	-	Contains the student's home school location (the school the student would attend if not classified). [Special education users: The student's primary special education placement must be at the primary location.]
	Gender	SexTable Keyword Selection	Quick Search, Assessment Filter Field	Key Reporting Fields	-	Contains the student's gender.

Online Data Dictionary

Constraints: Constraints are rules that determine whether profile data is valid or not. They can be configured to act as either warnings or absolute requirements when information is entered into student or other profiles, and serve to ensure that the information entered into the system is complete and accurate. Constraints can act in concert with profile form sections to guide and enforce the integrity of any data entry processes. The process of defining constraints is covered in this guide.

Standard Reports: PowerSchool Special Programs has a built-in ad-hoc reporting capability that makes it easy to develop list reports and multi-dimensional reports. The state model database will typically include a set of pre-formulated ad-hoc reports, although additional ad-hoc reports can be created by any authorized user to meet other needs. As with forms, all ad-hoc report definitions are stored in the client database. Developing standard reports is covered in the “Reporting User Guide”.

Advanced Reports: PowerSchool Special Programs includes an advanced reporting capability that is designed to reproduce complex reports mandated by the government such as the ASSA and Special Education December Counts. Advanced reports can reproduce the exact format specified by the requesting agency.

Document Repository

PowerSchool Special Programs provides a system of templates, documents and events that enable a work-flow process to be implemented in a school district. For Special Education, typically a template is set up for each major step in the IEP process: pre-referral, referral, evaluation, eligibility determination, and the IEP itself. The workflow component of PowerSchool Special Programs includes a guided

process implemented through configurable “Document Actions” and “Section Actions”. The outputs are mandated documents such as the IEP and a chronology of actions and events regarding a student.

Document Templates: Typically a document template is created for each major step in a workflow process. Again, for Special Education, these major steps are typically pre-referral, referral, evaluation, eligibility determination, and the IEP itself. Generally, the templates are created and refined in the state model database so that they can be easily deployed and then customized for individual school districts. Once a template is deployed in a database, documents can be created from that document template.

Document templates include the following major elements:

Document Fields: Document fields are very similar to data fields in profiles. In fact, typically a number of document fields are linked to corresponding profile data fields, and a capability called flow-back allows information to flow back and forth between the profile field and the document field. Other document fields only exist in the document itself and contain information that is not stored in any Profile. A special type of field, called a long text field, allows an unlimited amount of narrative text to be entered when creating documents from the template. Long text fields can exist only in documents.

Sections: Template sections function very similarly to profile form sections. A set of forms sections is filled with data and narrative text, and collectively they make up a document. Hence an IEP template might include a form section for the IEP meeting, another for the goals and objectives, another for the present levels of performance, etc. To print an entire IEP or other document, it is a matter of printing these form sections back to back. A special type of form section, called a letter, allows notification letters related to the process to be sent and stored in the database. Each template form section can also have security information associated with it that determines which users can enter information into the form section.

Child Templates: A frequent requirement is that a document section should include some kind of repeating data, which may take any of the following forms: 1) a group of fields that can be repeated as many times as needed by the end user, 2) a table where the user can add as many rows as needed, or 3) the user needs to repeat the entire section as many times as needed. To support these scenarios, child templates can be configured which define the set of fields (i.e. data dictionary) that is repeatable. Once the child template is defined, along with its data dictionary, it can be presented as repeating data on one or more sections including repeating sections. PSSP allows child templates to be presented in a section as “repeating rows” (repeating group of fields or repeating rows of a table) or “repeating sections”. It is also possible to have a child template nest within a child template (a.k.a grand-child templates).

Document/Section Actions: Section and document actions are added to a template to implement work flow rules. Section actions are configured to function for a specific section of the template, whereas document actions work across the document regardless of what section the user is working with. Such actions are configured to be invoked conditionally depending on the specific data being entered into a document, and/or upon the security profile or identity of the user entering the information. These actions can be programmed to perform a number of different actions including preventing a user from entering invalid data, posting an event to the event chronology (described later), including/excluding a form section from the document, providing the user with explanations and/or warnings and much more.

Document History: For each student, PSSP maintains a history of documents such as IEP documents, referral documents, etc. Documents can be created from templates as described above. Additionally, if documents are coming in from an outside source, these documents can be uploaded into PowerSchool Special Programs and incorporated into a student's document history. A variety of formats are supported for viewing including PDF, Word and multi-page TIFF formats

When an IEP or other document is first created, the status of the document is initially set to draft and the document is readily editable by authorized user(s). When the draft document has been completed, the status of the document can be changed to review which narrows which users can edit the document (normally to those in a supervisory position). When the document is complete, the status is changed to final which prevents any further editing.

Event History: For each student, PowerSchool Special Programs maintains an event chronology which keeps a running history of any relevant actions or events regarding a student. Generally, these events are generated through activity related to documents. For example, finalizing a document or sending a notification letter typically generates an event in the event chronology. When an event is generated, PowerSchool Special Programs is designed to notify appropriate users through the messaging system of PowerSchool Special Programs.

Communications Repository

The communications system in PowerSchool Special Programs supports the following means of communication:

Messages: This is conceptually similar to Internet email in that PowerSchool Special Programs messages are addressed to specific users. In fact, PowerSchool Special Programs messages can be automatically forwarded to Internet email. However, PowerSchool Special Programs messages play a critical role in various workflow processes for the purpose of notification. Such notification messages can be sent by other users or automatically by PowerSchool Special Programs itself. The messaging system can also be used by teachers to submit lesson plans to their supervisors for approval.

Announcements: This is a “broadcast” style of communication where large groups of users all get the same communication (e.g. all users district-wide in a particular security group). Announcements can be linked to the calendar.

Discussion Groups: PowerSchool Special Programs allows you to set up threaded discussion groups and control which users have access through an invitation system.

Calendar Data: This includes the school in-session calendar, calendar-based announcements, group calendar events, and personal calendar events.

Configuration Tasks

Creating and Assigning a Configuration Task

In order to properly audit changes to configuration and associate those changes with a specific change request or “ticket”, PowerSchool Special Programs requires you to create a new configuration task or select a configuration that was previously created. Once a configuration task is selected, then all of your configuration work will be audited within that configuration task. To create a new configuration task, follow these steps:

1. Select “Configuration” from the “Administration” menu and then select the “Config Tasks” tab. Generally, you need to have a configuration task to perform configuration within configuration tabs such as “Profile Types”, “Keyword Tables” and “Document Templates”.
2. You can view open tasks here to see if the task already exists. In the tool bar, you can also view tasks that have been closed and potentially re-open one of them. Click “Add New Task” to add a new one.
3. Enter a unique name for the task in the ‘Task Name’ field (typically a ticket number from a ticketing system). Enter a description of the configuration changes that will be made in the ‘Description’ field. You can optionally enter or look up the User ID of an authorized user in the “Assigned to” field to assign the configuration task to that user (see FYI below). Click “Accept” to proceed and create the new task.
4. You can now select the new task. Any configuration changes performed with this task selected will be audited against it. You can use the edit (pencil) or delete (trashcan) icons to edit or delete the task; however, once audited configuration changes have been made with the task selected, you will not be able to delete it. You can view the quantity of audited configuration changes made for any task by looking at the “Audited Events” column in the list of configuration tasks. This column also has a magnifier icon that allows you to drill in and access both a high-level view and a detailed view of the changes made via the configuration task.

Configuration Task Workflow

The current workflow status of each task is shown in the “Status” column of the configuration tasks view. The possible workflow status values are as follows:

- **Development:** Each new configuration task starts with this status, which continues until the configuration work is completed. When the task is ready for testing, the task is transitioned to the next status, which is “Testing”. To transition the status of a task, make sure it is selected which changes the value in the “Status” column to a dropdown menu used for transitioning the status. When the task is initially transitioned to testing, the developer lands on a verification tab

that shows all the high level items included in the configuration task, listed as links that perform a “Verify All” on the respective item.

- **Testing:** When testing is complete, the tester should transition the task to “Acceptance” if testing is successful or back to “Development” if additional development must be done to address issues found.
- **Acceptance:** This status is intended to identify that the task has been tested sucessfully but has not yet been deployed to the next database environment in the release pipeline (if applicable). If there is no release pipeline where this is applicable, the configuration task can immediately be transitioned to the final state of “Closed”.
- **Closed:** When the configuration task is closed, the configuration task is moved from the “Open Configuration Tasks” tab to the “Closed Configuration Tasks” tab. Note that a closed configuration task can still be transitioned back to “Development”.
- **Deployed:** This state is equivalent to closed. A configuration task is moved from “Acceptance” to “Deployed” only when the CMT is used to push the changes associated with the configuration task to the next database in the pipeline. Configuration tasks that are “deployed” appear in the “Closed Configuration Tasks” tab.

Profile Data Schema Customization (Basics)

Covers how to customize the profile data schema, forms for entering and viewing data, and constraints

Chapter
3

Getting Started (Tour of Profile Configuration)

This section assumes you are authorized to log in as a CONSULTANT and potentially make changes to the database. Follow the steps below to examine an existing profile data schema:

Step 1: Log in as a user authorized to view configuration. Select “Configuration” from the “Administration” menu.

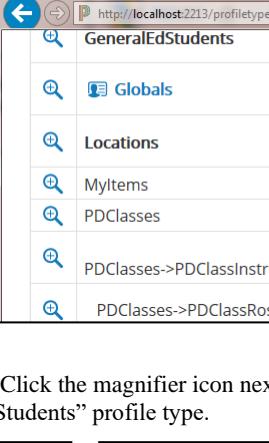
Step 2: Select the “Profile Types” tab.

FYI: All the different types of profiles in the database are listed here. This example may vary from your database.

Name	Caption	Description	Relationship	Count
Classes	Classes	Maintains information about classes.	Many-To-Many Relationship	0
Classes->ClassGeneralEdRoster	Class General Ed Roster	Class General Ed Student Roster	Many-To-Many Relationship	0
Classes->ClassStaffRoster	Class Staff Roster	Class staff roster.	Many-To-Many Relationship	0
Classes->ClassStudentRoster	Class Student Roster	Class student roster.	Many-To-Many Relationship	0
Classes->Reservations	Reservations	sdsd	Many-To-Many Relationship	1
Districts	Districts	Maintains information about Districts.	Many-To-Many Relationship	1

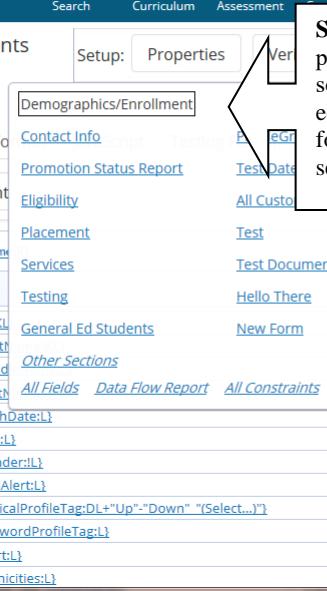
PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 3: Click the magnifier icon next to the “Students” profile type.



GeneralEdStudents	General Ed Students	General Education Students	0
Globals	Globals	Maintains information global to the system.	4
Locations	Locations	Maintains information about Locations.	Organizational Hierarchy Level 2
MyItems	My Items	Item	0
PDClasses	PD Classes	Professional Development Classes	0
PDClasses->PDClassInstructorRoster	PD Class Instructor Roster	Maintains instructor roster for professional development classes.	Many-To-Many Relationship 0
PDClasses->PDClassRoster	PD Class Roster	Maintains roster for professional development classes.	Many-To-Many Relationship 0
	Staff	Maintains information about Staff.	Inactive Profiles Allowed 2
	Alternate Staff Locations	Tracks alternative working locations for staff	Many-To-Many Relationship 0
Caseload	Caseload	Maintains student caseloads for staff members.	Many-To-Many Relationship 0
Students	Students	Maintains information about Students.	Inactive Profiles Allowed 16
Students->GroupPlanRecords	Group Plan Records	Contains selected data about group plans for convenient reporting access.	0
Students->Mandates	Mandates	Hello	0
Students->ServiceRecords	Service Records	Maintains history of service related	Service Capture 2

Step 4: The set of data fields that comprise the student profile (or other profile type) is organized into “form sections” to facilitate organization when viewing and editing information. You will view and work with one form section at a time. Use this fly out menu to select the section you want to work with.



The screenshot shows the "Profile Types > Students" page. On the left, there's a "Demographics/Enrollment" section with "Output Format" set to "HTML Form". Below it is a "Student Information" table with fields for ID, First Name, Middle Name, Last Name, Birth Date, Age, Gender, Red Alert, Logical Profile Tag, Keyword Profile Tag, Alert, and Ethnicsities. To the right, a fly-out menu lists several "Form Sections": Demographics/Enrollment, Contact Info, Promotion Status Report, Eligibility, Placement, Services, Testing, Test, Test Document Grids, Hello There, General Ed Students, Other Sections, All Fields, Data Flow Report, All Constraints, Style Sheet, New Form, Case Number, Address, City, State, Zip Code, Home Phone, LongText, and Test Numeric. A callout box points to the "All Constraints" option in the menu.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 5: By default, you will be looking at the “Output Format” tab of the section. This view gives a sense of what the form will look like when viewed by the end user.

Demographics/ Enrollment

Output Format HTML Format JavaScript Testing Format Section Properties Fields Data Flow Report Constraints

Edit Section... Print

Student Name: [FirstName:R] [LastName:R] Birth Dates: [Birthdate:R]

Student Information

ID {ID:KL}	Case Number {CaseNumber:L}
First Name {FirstName:KL}	Address {Address:L}
Middle Name {MiddleName:L}	
Last Name {LastName:KL}	
Birth Date {BirthDate:L}	
Age {Age:L}	
Gender {Gender:IL}	
Red Alert {RedAlert:L}	
Logical Profile Tag {LogicalProfileTag:DL+“Up”-“Down” “(Select...)”}	
Keyword Profile Tag {KeywordProfileTag:L}	
Alert {Alert:L}	
Ethnicities {Ethnicities:L}	

FYI: This view also shows the names of fields as links. If you click a field name link, you will access a list of fields with the field you clicked scrolled into view and highlighted.

Step 6: Select the “XHTML Format” tab to see the underlying HTML format used to create the look and feel of the form section. An HTML Format consists of standard HTML with various database field references and directives embedded in it.

Demographics/Enrollment

Output Format **HTML Format** JavaScript Testing Format Section Properties Fields Data Flow Report Constraints

Edit Section... Print

```
<TABLE WIDTH=90%>
<TR>
<TD WIDTH=50% CLASS=VERYSMALL ALIGN=left><B>Student Name:</B> {FirstName:R} {LastName:R}</td>
<TD WIDTH=50% CLASS=VERYSMALL ALIGN=right><B>Birth Dates:</B> {Birthdate:R}</td>
</tr>
</table>

<table BORDER=1 CLASS=DATAFORM CELLSPLICING=0 CELLSPADDING=4 WIDTH="90%">
<tr>
<td colspan=2 class="FIELDGRIDHEADING">Student Information</td>
</tr>
<tr>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSPLICING=0 CELLSPADDING=2 WIDTH="100%">
{ID:KL}
{FirstName:KL}{MiddleName:L}{LastName:KL}
{BirthDate:L}{Age:L}
{Gender:IL}{RedAlert:L}{LogicalProfileTag:DL+“Up”-“Down” “(Select...)”}{KeywordProfileTag:L}{Alert:L}{Ethnicities:L}
</table>
```

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Special Education

Profile Types

Demographics/Enrollment

Output Format HTML Format JavaScript **Testing Format** Section Properties Fields Data Flow Report Constraints

Edit Section... Print

Student Name: Birth Dates:

Student Information

ID*	[Redacted]
First Name*	[Redacted]
Middle Name	[Redacted]
Last Name*	[Redacted]
Birth Date*	[Redacted] <input type="button" value="Calendar"/>
Age	[Redacted]
Gender*	(none) <input type="button" value="Select..."/>
Red Alert	<input type="checkbox"/>
Logical Profile Tag	(Select...) <input type="button" value="Select..."/>
Keyword Profile Tag	(none) <input type="button" value="Select..."/>

Case Number	[Redacted]
Address	[Redacted]
City	[Redacted]
State	(none) <input type="button" value="Select..."/>
Zip Code	[Redacted]
Home Phone	[Redacted]

B I U A

Special Education

Profile Types > Students

Demographics/Enrollment

Output Format HTML Format JavaScript Testing Format Section Properties **Fields** Data Flow Report Constraints

Verify Fields Edit Section... Print

Student fields linked to 'Demographics/Enrollment'

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Field Name	Data Type (Length)	Properties	Attributes														
ID	ID(10)	Quick Search	Reporting Fields	Contains the student's name.													
LastName	Character(51)	Quick Search	-	Contains the student's name.x													
FirstName	Character(50)	Quick Search	-	Contains the student's name.													
MiddleName	Character(30)	-	-	Contains the student's name.													
HomeSchool	Location Profile Reference	Organizational Parent Reference, Quick Search,	-	Contains the student's home school location (the school the student would attend if not classified). [Special education users: The													

FYI: Each field has a name, a data type, a description and various other properties. These will be described in more detail in the “Understanding the Data Dictionary” on page 63”.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 9: Select the “Constraints” tab to see a list of all constraints linked to this particular form section.

Student constraints linked to 'Demographics/Enrollment'				
Name	Evaluated When (For)	Action Taken	User Message	Custom?
Birth Date Required	Section Edited (Do not evaluate for ADMIN/CONSULTANT)	Warn User	Student must have a birth date for state reporting	No
Invalid Birthdate	Section Edited	Warn User		
Deactivate 1	Bulk operation(edit security bypass) (For security group: Case Managers)	Deactivate		
Deactivate 2	Bulk operation(edit security bypass) (For security group: Case Managers)	Deactivate		
Remove Active Status	Section Edited	Remove Active Status		

FYI – About Constraints: A constraint is a rule that is used to identify and prevent cases of incorrect or invalid data, or in some cases to update data based on other data just entered. Constraints are covered in detail later in this chapter.

Step 10: If you click “All Fields” or “All Constraints” from this dropdown, you will see a list of all fields or constraints for the profile type

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

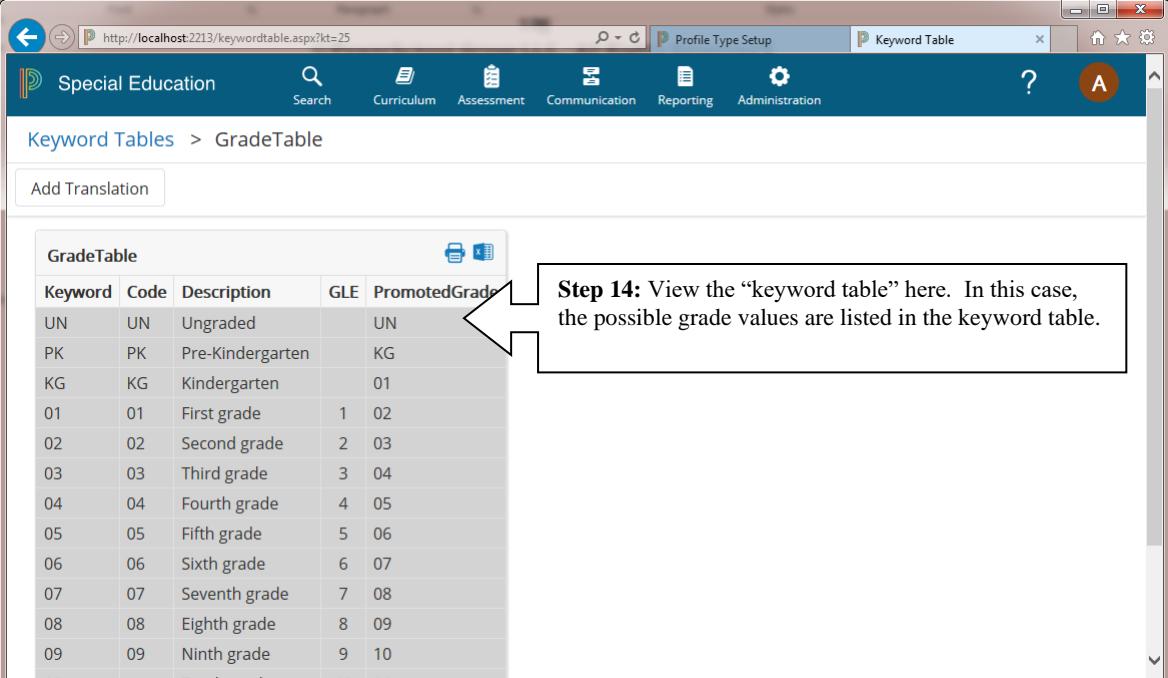
The screenshot shows the 'Profile Types > Students' section. A callout box points to the 'FYI' note: 'FYI: A field or constraint can be linked to more than one form section. In the "All Fields" or "All Constraints" display, you can see all the sections that a field or constraint is linked to.' Below this is a grid titled 'All Student Fields' with columns for Field Name, Data Type (Length), Properties, Attributes, Custom Field, Description, and Linked To Sections.

Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description	Linked To Sections
A2	A2 Keyword Selection	-	-	-	A2	Demographics/Enrollment
A3	A3 Keyword Selection	-	-	-	A3	Demographics/Enrollment
aasas	Long Text	-	-	-	dsdssdds	Demographics/Enrollment
Address	Character(50)	-	Reporting Fields	-	Contains the student's address.	Demographics/Enrollment, Demographics
Age	Calculated Integer	Internal Value Only	Reporting Fields	-	Calculates the student's age.x	Demographics/Enrollment, Demographics
AgeAsOfSept1	Calculated Integer	-	-	-	Calculates the student's age as of September 1st of the current school year	Demographics/Enrollment

The screenshot shows the 'Demographics/Enrollment' section under 'All Fields'. A callout box points to 'Step 11: Return to the first section using the fly out menu.' Another callout box points to 'Step 12: Select the "Fields" tab.' Below is a grid of fields with a callout pointing to 'Step 13: Look in the "Data Type" column and find any "Keyword Selection" field. Fields of this data type can store a value from a pre-defined list of possible values. To see the "keyword table" that delineates the possible values, click the corresponding link here.' The grid has columns for Field Name, Data Type (Length), Properties, Attributes, Custom Field, and Description.

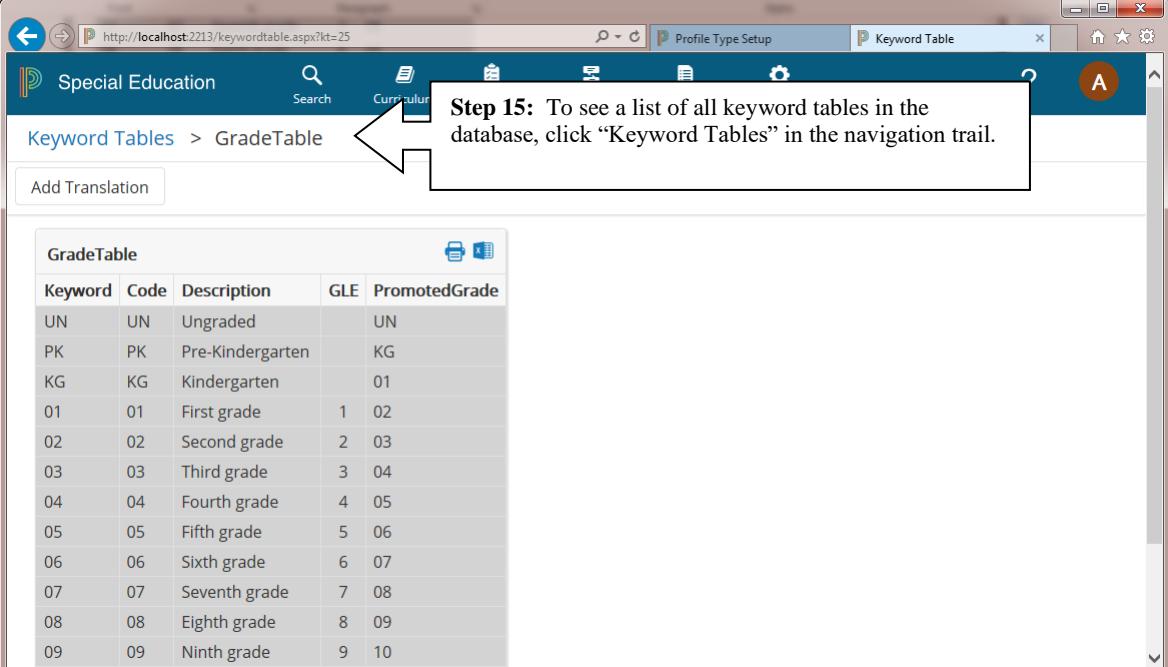
Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description
ID	ID(10)	Quick Search	Reporting Fields	-	Contains the student's id number.
LastName	Character(51)	Quick Search	-	-	Contains the student's name.
FirstName	Character(50)	Quick Search	-	-	Contains the student's name.x
MiddleName	Character(30)	-	-	-	Contains the student's name.
HomeSchool	Location Profile Reference	Organizational Parent Reference, Quick Search, Assessment Field	-	-	Contains the student's home school location (the school the student would attend if not classified). [Special education users: The
Gender	GenderTable Keyword Selection	Field	Field	-	possible values, click the corresponding link here.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)



The screenshot shows the "GradeTable" keyword table. The table has columns: Keyword, Code, Description, GLE, and PromotedGrade. The data includes rows for UN (Ungraded), PK (Pre-Kindergarten), KG (Kindergarten), and grades 1 through 9. A callout box points to the table with the text: "Step 14: View the ‘keyword table’ here. In this case, the possible grade values are listed in the keyword table."

GradeTable				
Keyword	Code	Description	GLE	PromotedGrade
UN	UN	Ungraded		UN
PK	PK	Pre-Kindergarten		KG
KG	KG	Kindergarten		01
01	01	First grade	1	02
02	02	Second grade	2	03
03	03	Third grade	3	04
04	04	Fourth grade	4	05
05	05	Fifth grade	5	06
06	06	Sixth grade	6	07
07	07	Seventh grade	7	08
08	08	Eighth grade	8	09
09	09	Ninth grade	9	10
10	Tenth grade		10	11



The screenshot shows the "GradeTable" keyword table, identical to the one in the previous screenshot. A callout box points to the "Keyword Tables" link in the navigation trail with the text: "Step 15: To see a list of all keyword tables in the database, click ‘Keyword Tables’ in the navigation trail."

GradeTable				
Keyword	Code	Description	GLE	PromotedGrade
UN	UN	Ungraded		UN
PK	PK	Pre-Kindergarten		KG
KG	KG	Kindergarten		01
01	01	First grade	1	02
02	02	Second grade	2	03
03	03	Third grade	3	04
04	04	Fourth grade	4	05
05	05	Fifth grade	5	06
06	06	Sixth grade	6	07
07	07	Seventh grade	7	08
08	08	Eighth grade	8	09
09	09	Ninth grade	9	10
10	Tenth grade		10	11

The screenshot shows a software interface titled "Profile Type Setup" with a tab labeled "Keyword Tables" selected. The main area displays a grid of keyword tables, each with a magnifying glass icon next to its name. A callout bubble with the text "Step 16: You can view any other keyword table listed by clicking the magnifier icon next to its name." points to one of these icons.

Name	Description	Used For Fields
GenPurpClasses	Supports the "Classes" grid on the "General Purpose Field Status Report". Contains an entry for each field which might appear in the grid.	
GenPurpDistricts	Supports the "Districts" grid on the "General Purpose Field Status Report". Contains an entry for each field which might appear in the grid.	
GenPurpLocations	Supports the "Locations" grid on the "General Purpose Field Status Report". Contains an entry for each field which might appear in the grid.	
GenPurpPersonnel	Supports the "Personnel" grid on the "General Purpose Field Status Report". Contains an entry for each field which might appear in the grid.	
	Supports the "Students" grid on the "General Purpose Field Status Report".	

Understanding the Data Dictionary

The previous section covered how to access the data schema including the list of defined database fields (otherwise known as the “data dictionary”). This section describes in more detail the various aspects of individual database fields and their properties. Before proceeding, return to the list of fields for students as described in the previous section.

Field Name: Each field has a name that is unique within the type of profile it is for (e.g. student profiles). A field name can be 25 characters long with letters, digits and the underscore character (but no spaces or other punctuation).

Data Type (Length): Each field stores a specific type of data. For some data types, such as “Character”, a field length is also indicated. An explanation of each data type is given below.

Field Name	Data Type (Length)	Properties	Custom
ID	ID(10)		
LastName	Character(51)		
FirstName	Character(50)		
MiddleName	Character(30)		
HomeSchool	Location Profile Reference	Organizational Parent Reference, Quick Search,	Contains the student's home school location (the school the student would attend if not classified). [Special education users: The

FYI – About Data Types: Each field in the data dictionary has a “data type” that identifies the kind of information it stores. The possible data types are described below:

ID (Length): Most types of profiles (students, staff, etc) include an ID field that serves as the system-wide unique identifier for profiles of that type. Each such ID field has a maximum character length associated with it. When this type of field is included in a profile form section as an editable field, it shows up as a text box in which the user can enter the ID. However, if PowerSchool Special Programs is integrated with the school district’s Student Information System (SIS), this field will generally be set as read only.

Character (Length): This type of field is used to store text. Each character fields allows a single line of text with a maximum number of characters (specified as part of the field definition when the field was created or modified). A character field is typically used to store information such as names, addresses, phone numbers, and ZIP codes. When included in a profile form section, a character field appears as a text box into which users can type.

Logical: This type of field is used to store a true/false (yes/no) value. You might use this field to show whether a student is low income or receives a certain service. When included in a profile form section, a logical field appears as a checkbox; however, it is also possible to display it as a dropdown menu.

Date: A field used to store a date. A date field appears on forms as a text box into which users can type, but with a link next to it to bring up a popup calendar.

Date/Time: Similar to date but stores a time (hours and minutes) as well.

Integer: Stores an integer or whole number value. A minimum and maximum value can optionally be specified in the field definition. By default, an integer field is presented as a text box in edit mode. However, if the integer field has both a minimum and maximum value, it can optionally be presented as a dropdown menu.

Numeric: A field used to store a number that can include decimal places. Be careful not to assume that all numbers should go into Numeric fields. For example, ZIP codes beginning with a zero would be stored without the leading zero; after all, 07701 as a number is the same as 7701. Also, if numbers in this field will not have decimal places, the integer data type should be used instead. A numeric field appears on forms as a box into which users can type. Of course, only valid numbers are accepted. A minimum and maximum value can optionally be specified in the field definition.

Numeric fields can be marked with a currency option, which modifies the behavior of the numeric field in two ways: 1) expands capacity to support 10 digits left of the decimal point, 2) displayed as currency (e.g. \$9.10) on forms.

Numeric fields can also be marked as containing a score which helps PowerSchool Special Programs interpret the values as scores.

Short Text (Length): Short text fields are similar to character fields except they allow more than one line of text. As with a character field, the maximum number of characters is specified as part of the field's definition. A short text field appears on forms as a multi-line text box into which users can type.

Long Text: Long text fields are similar to short text fields except that they allow an unlimited amount of text. Use of long text fields may have a larger performance impact on the servers, and so should be used sparingly in profile data.

Profile Reference: This kind of field references a profile of another profile type. For example, the student profile might have a “Staff Reference” field to identify the student’s case manager and a “Location Reference” field to identify the student’s home school. A Profile Reference field appears on forms as a text box with a look-up link next to it. Users can use the look-up link to fill this field, or can type in the ID number if the referenced profile type has an ID field.

Note that profile reference fields can be marked with a “Display as Hyperlink” option. When this option is enabled, the field will be displayed on any forms as a hyperlink that the user can click to open a popup window showing the corresponding profile. For example if this option is enabled for a staff profile reference field, the field will be displayed as a hyperlink that the user can click to view the staff profile.

Keyword Selection: A keyword selection field contains a single value selected from a pre-defined list of possible values (e.g. Gender, Ethnic, Grade Level, etc). Keyword selection fields are presented to the user as a dropdown menu. The items in the dropdown menu are populated from a “keyword table”, introduced to you in the previous section.

Image: Stores a web-ready image type (GIF, JPG, PNG or SVG format), useful for storing a photo of a person or a logo. When this type of field is on a form in edit mode, the user can select an image file for

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

uploading into the field. Pasting a graphic into the field is also supported in certain browsers: Chrome, Edge, and Firefox. Safari does not support pasting graphic images at the time of writing.

File: Stores a file, including the file name and contents. When this type of field is on a form in edit mode, the user can select a file for uploading into the field.

The screenshot shows the 'Profile Type Setup' application with the URL <http://localhost:2213/profiletype.aspx?pt=2&sec=13&view=F>. The 'Fields' tab is selected. A callout box highlights the 'EducPlanWaivers' field, which is described as a 'Logical Value (prevent empty values)' field. It is multi-valued on the 'EducPlanWaiverTable'. The callout box contains the following text:

Multiple-Value Field: Most fields in the data dictionary will store a single value; however, if you look through the data dictionary, you may find some fields that are multiple-value. A multiple-value field is a field that can store more than one value, and each value has a name associated with it. In this illustration, the "EducPlanWaivers" field is a logical (yes/no) multiple-value field. Each individual value stored in this field is associated with a specific name, and these names are defined as keywords in an associated keyword table. You can view the keyword table by clicking its name.

The screenshot shows the 'Profile Type Setup' application with the URL <http://localhost:2213/profiletype.aspx?pt=2&sec=13&view=F>. The 'Fields' tab is selected. A callout box highlights the 'Properties' column for the 'ID' field, which is described as having 'Quick Search' properties. The callout box contains the following text:

The "Properties" column shows special properties of each database field:

- Quick Search:** Fields with this property appear on the quick search form and contain a database index to speed searches.
- Organizational Parent Reference:** Fields with this property are used to control organizational security when users are accessing this type of profile. In other words, the value of this field in a profile determines where that profile is located for security purposes.
- Assessment Filter Field:** Fields with this property are used as an analytical field in the assessment repository.
- Internal Value Only:** Fields with this property are used to hold temporary values in a process, but are not of interest from a viewing or reporting perspective.

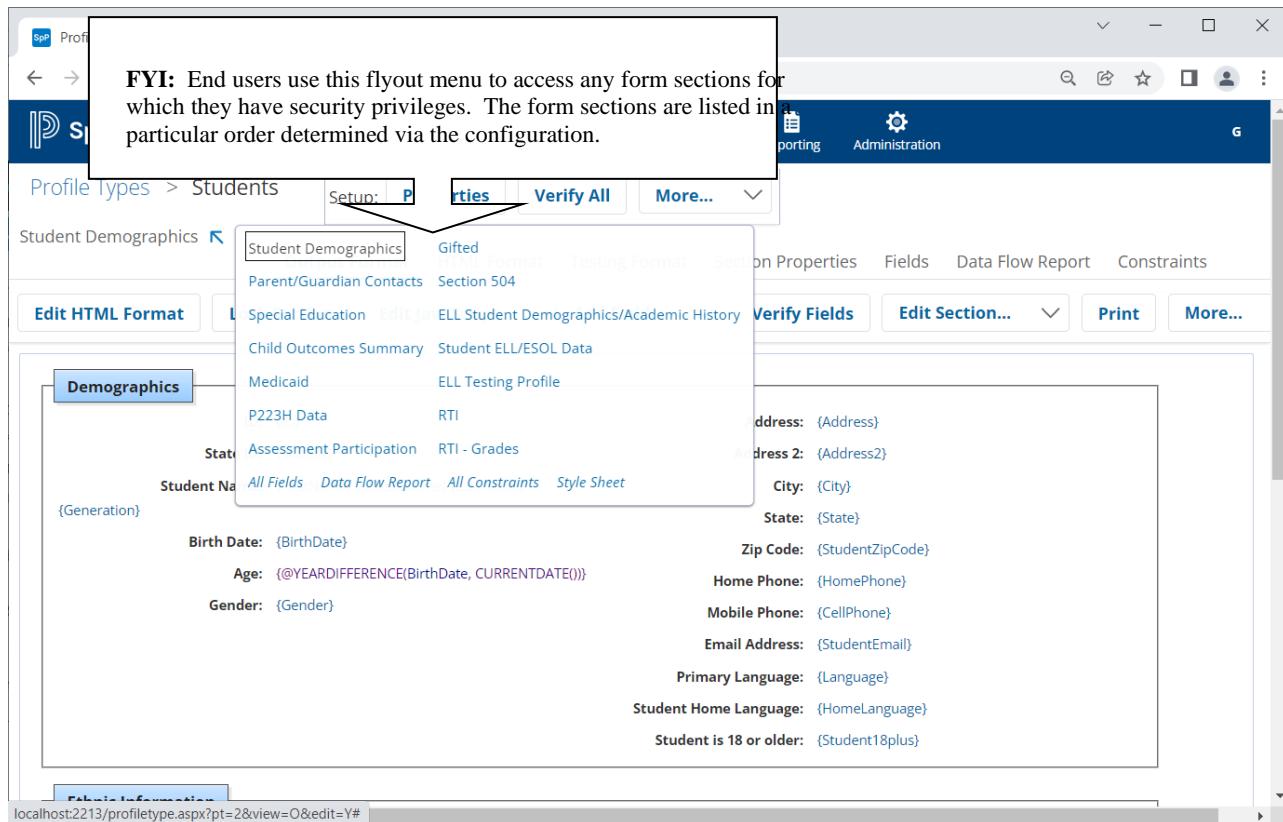
PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

The screenshot shows a web-based application titled "Special Education" with a navigation bar for Search, Curriculum, Assessment, Communication, Reporting, and Administration. The main content area is titled "Profile Types > Students". A sub-menu "Demographics/Enrollment" is open, showing options like "Output Format", "HTML Format", and "Java". Below this are buttons for "Verify Fields" and "Edit Section...". A callout box highlights the "Custom Field" section, which states: "As a general rule, fields defined in the state model database are NOT custom fields, but fields added locally by the school district are custom fields and so will have a 'Yes' in this column. The school district ADMIN user can only delete or modify custom fields, since these are not used on state/federal mandated reports." Another callout box highlights the "Attributes" column, stating: "The 'Attributes' column shows 'attributes' associated with each field. Attributes are themselves customizable and new ones can be defined and associated with fields for purposes of categorization and field level security." A table below lists student fields linked to Demographics, including columns for Name, Attributes, Custom Field, and Description. The table shows two rows: one for Student ID (which is a reporting field) and one for IsLabStudent (which is a logical value). The IsLabStudent row has a note indicating it prevents empty values.

Name	Attributes	Custom Field	Description
StudentID	Reporting Fields	-	Contains the student's id number.
IsLabStudent	Logical Value Only	-	Indicates whether the student is a lab training student.

Adding a Profile Form Section

Before attempting to add or customize profile form sections, it is helpful to understand how they are accessed by the end user. This is shown in the illustration below.



The best way to familiarize yourself with customizing profile form sections is to add a new profile form section and experiment with it. The steps below will lead you through the process. Note that in this section, you will be adding a new form for existing fields. The next section will cover adding a new data field.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

The screenshot shows the 'Profile Type Setup' interface for 'Students'. The 'Demographics' section is selected. A callout box contains the following text:

Step 1: Access the configuration for the profile type for which you want to add a section as described in the previous section. If necessary, select a configuration task. Then click Add New Section here.

Below the callout box, the 'Demographics' section is shown with various fields and their expressions:

- ID: {ID}
- State ID: {StateID}
- Student Name: {FirstName} {MiddleName} {LastName} {Generation}
- Birth Date: {BirthDate}
- Age: {@YEARDIFFERENCE(BirthDate, CURRENTDATE())}
- Gender: {Gender}
- Email Address: {StudentEmail}
- Primary Language: {Language}
- Student Home Language: {HomeLanguage}
- Student is 18 or older: {Student18plus}

The screenshot shows the 'Profile Form Section Properties' screen for 'Students'. A callout box contains the following text:

Step 2: Enter a name for the section.

The 'Demographics' section properties are displayed:

- Section Name: Demographics
- Prior Name(s): (used by CMT to rename from prior section name or semicolon-separated names)
- Purpose of section: View and Edit
- Options:
 - White Background
 - No Page Break When Multiple Forms Printed
 - Show Profile References With IDs
 - Include Fields In TransferEnvelope
 - Mobile Device Only
 - Mobile Device Only

A second callout box contains the following text:

Step 3: By default, the “purpose” of the form section is “View and Edit”. If you select “View Only”, the form can be used to view a profile, but not for editing. If you select “Print Only”, the form will go into print mode as soon as the end user clicks its link.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Screenshot of the "Profile Form Section Properties" page in the Special Programs application.

The page title is "Profile Form Section Properties". The URL is "localhost:2213/profileformsectionprops.aspx?pt=2&sec=new&secret=13&view=0&gm=0".

The main content area is titled "Add New Form Section For Students". It contains the following fields:

- Section Name: Demographics
- Prior Name(s): (used by CMT to rename from prior section name or semicolon-separated names)
- Purpose of section: View and Edit
- Options:
 - White Background
 - No Page Break When Multiple Forms Printed

A callout box labeled "Step 4: Use this dropdown to control the position in which this profile form section is displayed, relative to other form sections." points to the "Purpose of section" dropdown.

Below the dropdown, there is a note: "(Custom form) is not synchronized by default".

Applicable Packages:

- Service Capture
- RTI
- Special Education
- Section 504
- ELL
- Gifted Talented

Insert Section Where?: Between 'RTI - Grades ' and end

Buttons: Accept (highlighted), Cancel.

A callout box labeled "Step 5: Click "Accept" to add the new form section." points to the "Accept" button.

Screenshot of the "Profile Type Setup" page in the Special Programs application.

The page title is "Profile Type Setup". The URL is "localhost:2213/profiletype.aspx?pt=2&sec=119&view=0&edit=Y".

The main content area is titled "Profile Types > Students". It shows "Student Demographics 2" and "Edit HTML Format".

A callout box labeled "Step 6: To see how the form can be customized further, click Edit HTML Format." points to the "Edit HTML Format" button.

Below the table, there is a note: "Fields Data Flow Report Constraints".

Buttons: Edit Section..., Print, More...

Table: Student Demographics 2

ID	{ID:RL}
Last Name	{LastName:RL}
First Name	{FirstName:RL}
Middle Name	{MiddleName:L}
School	{School:L}
Birth Date	{BirthDate:L}
Case Manager	{CaseManager:L}
Gender	{Gender:L}
Grade	{Grade:L}

Step 7: You can now edit the section format directly. The section format is simply HTML with PSSP “directives” mixed in. Such directives are enclosed with {squiggly} brackets and directs the system to include fields and other data elements in the form. For example, the {LastName:RL} directive directs the system to include the LastName field with the “R” and “L” modifiers. The “R” modifier ensures that the field is shown read only (not editable) in this form section. The “L” modifier directs the system to automatically include an entire HTML table row for this field where the left cell contains the label and the right cell contains the field. The system generates a label automatically using the field name, however, you can specify a non-default field label by using a format such as {LastName:RL“Student’s Last Name”}. A complete listing of form directives and modifies is found in Appendix E.

If you make any changes to the format, be sure to click the “Save” button to save them.

Using External HTML Editors: Some users who are configuring complex HTML formats prefer to use their own HTML editor. This is perfectly fine. The user simply copies and pastes the current format from PowerSchool Special Programs into their favorite editor and then, after editing, pastes the result back into PowerSchool Special Programs and saves. However, if there is more than one user doing configuration, a user using an external editor can easily overwrite another user’s work. To prevent this, a user can click Lock Section to lock the HTML format while working on it with an external editor. Other users will see a bright red message like the one shown below and will therefore know another user is working on it. In an emergency another user can still edit the section format even if it is locked, but this other user must unlock it first. Unlocking by another user is an action will be audited and if the original user returns, that the original user’s lock will no longer be there, which provides an implicit warning.

Profile Form Section Property reference:

- **Section Name:** Unique name of section
- **Prior Name(s):** Generally, this is only relevant when the section has been released to end user tenants, and then is renamed. The old name should be copied into this field so that the CMT synchronizer can recognize this change as a rename instead of a delete and an add. If several renames are done over time, multiple previous names can be captured in this field separated by semi-colons.
- **Purpose of section:** Indicates what the section will be used for (viewing only, viewing and editing, printing, etc).
- **Options:** Various checkbox options that modify the behavior of the section, as follows –

- **White Background:** Enabling this option displays the form within a thin rectangular outline within which the background color is white. At the current time, the overall application background is white; therefore, the effect of this option is not as noticeable as it once was when the background color was a light gray.
- **No Page Break When Multiple Forms Printed:** When the form is printed for multiple students at a time, enabling this option removes the default page-break between forms, which can save paper.
- **Show Profile References with IDs:** When a section has profile fields (e.g., staff fields), normally just the name shows when the form is in view mode. Enabling this option causes such fields to show both the name and the ID.
- **Include Fields in Transfer Envelope:** Relevant for student profile forms only. Indicates that any fields on this section should be included in student transfer envelopes.
- **Mobile Device Accessible:** This indicates the profile form section is appropriately formatted for display on a mobile phone OS.
- **Custom?** This is used to mark a profile form section is custom for a specific production tenant and is not considered part of the model.
- **Applicable Packages?** If any packages are checked here, then the form will only be accessible if a tenant is licensed for any of the checked packages.
- **Insert Section Where?** Use this to indicate where this section should appear in the flyout relative to other sections.

Creating and Modifying Keyword Tables

Keyword tables (introduced briefly in previous sections) serve three major purposes:

1. Keyword tables are used to define the possible selections for a “keyword selection” field, examples of which include Gender and Grade fields. When data entering a keyword selection field, the user selects a value from a dropdown menu. The items in the dropdown menu correspond to the keywords in the keyword table.
2. Keyword tables are used to define the values of a multiple-value field (introduced in previous sections). An example of a multiple value field is a “Service” logical field, with one yes/no value for each service the school district offers. In this example, there would be a keyword table that defines the various services.
3. Keyword tables can also be used to track relatively static kinds of information that can be referenced from PowerSchool Special Programs formulas. An example of this use will be included in this section.

For the purpose of illustrating how to create a keyword table, we will follow an example where we want to add the ability to track students' hair color. While this example is somewhat whimsical, it is a good example of how new data tracking requirements can readily be met by customizing the profile data schema. In this section, we will create a keyword table delineating the possible hair colors (e.g. blonde, brunette). In the next section, we will create a new student profile data field called "HairColor".

The screenshot shows a web-based configuration tool for PowerSchool. At the top, there's a navigation bar with links for Configuration, Settings, Curricula, Integration, Config Tasks, Profile Types, Keyword Tables (which is highlighted), and Document Templates. A sub-menu for 'Keyword Tables' is open, showing options like 'Add New Keyword Table', 'Import', and 'Export'. Below this is a table titled 'Keyword Tables' with columns for Name and Description. The table lists several existing keyword tables:

Name	Description
AddtionalProperties	Contains an entry for each additional property of a record in a record list.
AgeToGradeConvTable	A table showing the grade to which an Ungraded student should be assigned for state reporting based on their age.
AtRiskTypeTable	Contains the types of at risk service.
AtRiskValueTable	Contains the possible values of at risk service.
AttendanceTypeTable	AttendanceTypeTable

Annotations with callouts point to specific areas:

- Step 1:** Points to the 'Configuration' link in the top navigation bar and the 'Keyword Tables' tab in the sub-menu.
- Step 2:** Points to the table of existing keyword tables, indicating that even if the desired table doesn't exist, reviewing this list is worth doing.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 3: Click Add New Keyword Table.

Name	Description	Used For Fields
AdditlProperties	Contains an entry for each additional property of a record in a record list.	Classes AdditionalProperties Districts AdditionalProperties
AgeToGradeConvTable	A table showing the grade to which an Ungraded student should be assigned for state reporting, based on the student's age.	
AtRiskTypeTable	Contains the types of at risk service.	
AtRiskValueTable	Contains the possible values of at risk service.	Students.FreeReducedLunch Students.AtRiskService
AttendanceTypeTable	AttendanceTypeTable	Students.Attendance_Q1 Students.Attendance_Q2 Students.Attendance_Q3 Students.Attendance_Q4

Step 5: Enter a unique name (30 alpha-numeric characters including the underscore character, but no spaces) for the keyword table, and a more detailed description. A standard practice has been to suffix the name with the word "Table" and it is recommended that you follow this practice for consistency. See the notes below about selecting the "Customizability" option. When done with the form click "Accept" to continue.

Keyword Table Properties

Accept Cancel

Name: HairColorTable

Description: Identifies hair color options

Customizability (by ADMIN): Model Keyword Table (ADMIN cannot add custom keywords)

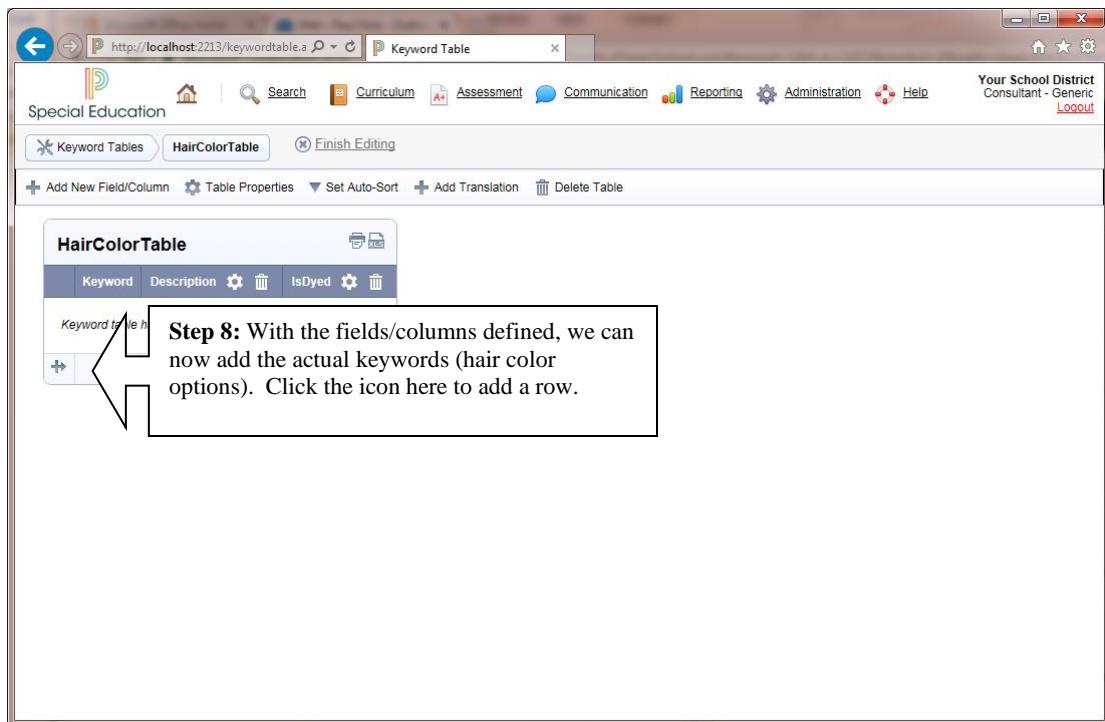
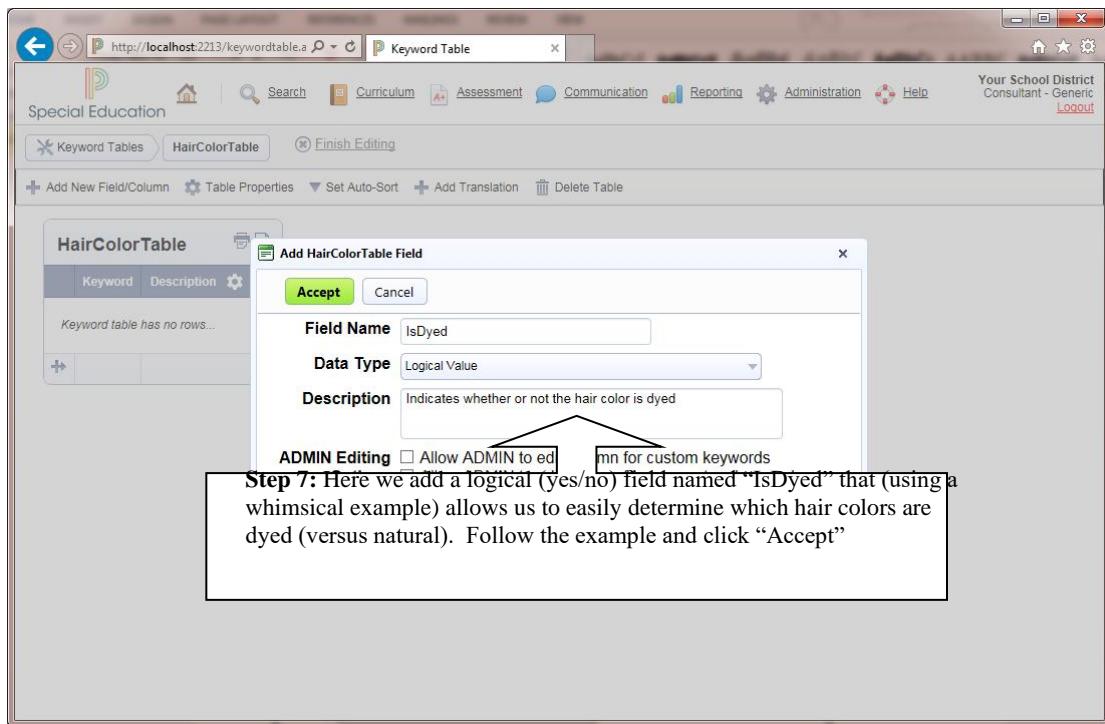
FYI – About the Customizability Dropdown (see Step 7 above): If you create a keyword table when logged in as the CONSULTANT, you can control whether the keyword table is considered to be a “custom” keyword table or a “model” keyword table. A custom keyword table is considered to be a customization for a particular school district, and therefore not part of the “state model” data schema. If you are logged in as the ADMIN and create a keyword table, the keyword table is assumed to be custom. However, if you are logged in as the CONSULTANT, the keyword table by default will be a model keyword table. For a model keyword table, there is an option to control whether or not the ADMIN can create individual custom keywords within the model keyword table. Note that if a database is synchronized with a state model database, only model data schema elements (keyword tables, keywords, etc) are synchronized.

Step 6: The initially empty keyword table appears. To flesh out this example, we will demonstrate adding a field/column to this keyword table before adding the actual hair colors. Click [Add New Field/Column](#) to proceed.

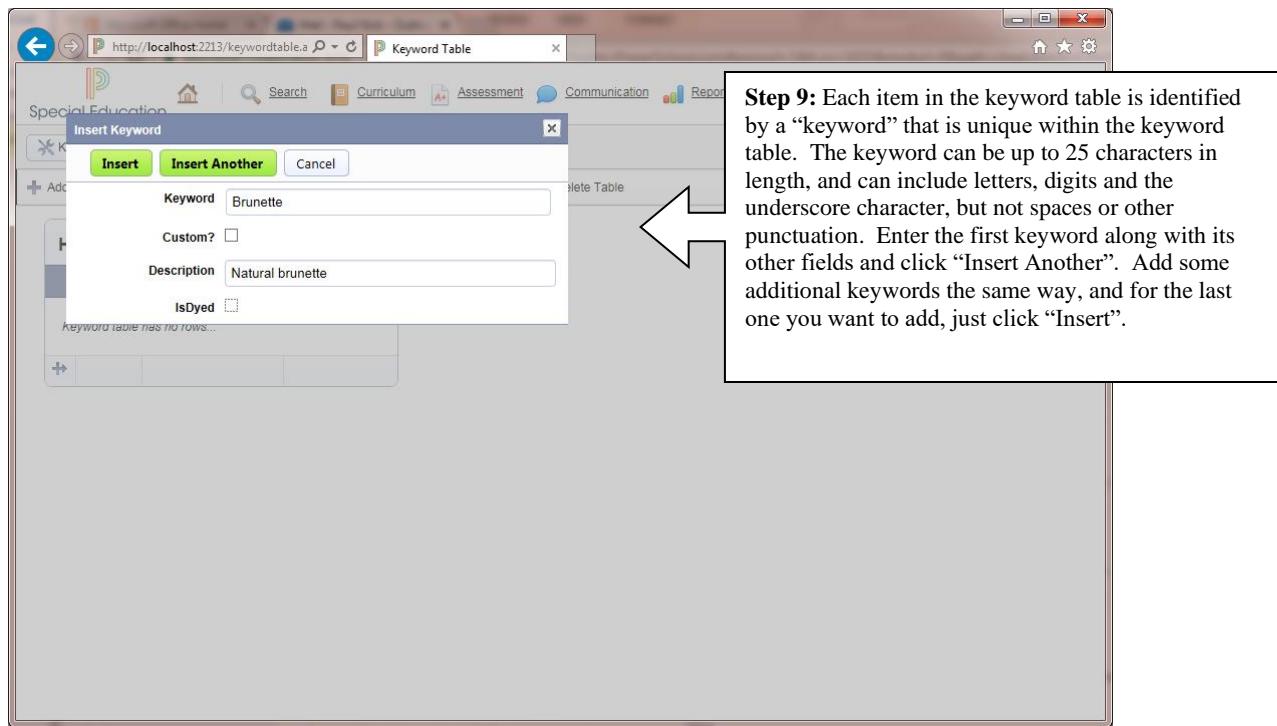
FYI: The unique keywords that we will later be adding to identify the specific hair color options are limited to 25 characters including letters, digits and the underscore character, but no spaces or punctuation. By default, these keywords will appear in the dropdown menu the end user will see. To provide more control and flexibility over what appears in the dropdown menu, a character field/column named ‘Description’ is automatically added to the table. The system will recognize a column named either ‘Description’ or ‘Name’ and use its text values in the dropdown in place of the keywords.

Optimizing the Keyword Table: By default, the keywords will appear in the dropdown menu the end user will see whenever the keyword table is used for a keyword selection field. However, if you want more control and flexibility over what appears in the dropdown menu, you can add a field/column to the keyword table with a name of “Name” or “Description” and with a data type of “Character”. PowerSchool Special Programs will automatically use the contents of this field/column to populate the dropdown. Additionally, if the keyword table will be used for foreign language translation or localization, you can provide translations in a similar column named as “Name_*languagecode*” or “Description_*languagecode*” where *languagecode* is the two-character ISO language code. For example, use Name_es or Description_es to keep Spanish translations.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)



PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)



Step 13: When you have added the necessary keywords, click Finish Editing.

FYI: To illustrate the usefulness of fields like this “IsDyed” field, consider the case where you have created and saved an ad-hoc report that shows all students who have dyed hair based on the HairColor field. Without the “IsDyed” field, the selection formula would need to be something like “HairColor=Green OR HairColor=Pink”. But if you later added Purple to the keyword table, the formula would immediately become incorrect. However, the “IsDyed” field allows you to use a more robust formula, in this case, simply “HairColor.IsDyed=true”.

Editing: The keyword table can be edited a row at a time, or alternatively it may be more efficient to click the edit icon in a column header and edit an entire column all at once. There are four ways to change the order of keywords: 1) clicking the up/down arrows at the left end of each keyword row, 2)

clicking the edit icon for a keyword row and setting the keyword position via the dropdown menu at the bottom, 3) drag and drop, or 4) setting the auto-sort for the keyword table. The “Set Auto-Sort” option in the toolbar allows specification of one or more sort columns for the keyword table with support for ascending/descending order on any column. Once auto-sort is in place, new keyword rows are always simply added and then automatically set to the correct position to maintain the sort (and the up/down icons are not available).

“InUse” keyword field: A convention has emerged that keyword tables have an “InUse” logical field to track which keywords are still in use (field has Yes value) versus deprecated (field has No value). This has now been formalized such that the platform will: 1) automatically set the default value of such columns to true, 2) automatically add this column to new model keyword tables, and 3) automatically filter on this column for keyword fields in profiles and documents. Previously, the F“InUse” modifier was applied to such keyword fields, but this is no longer necessary.

Optional Display Codes: A character-based column in a keyword table can now be designated as containing optional “display codes”. Via the user options screen, either the end user or the system administrator can enable these display codes to be shown in dropdown menus (along with the standard descriptive text) when the end user is completing forms. In any keyword table, the display codes column can either be the “Keywords” column or a specific character field. Only one column per keyword table can be designated to contain display codes. To designate the “Keywords” column to contain display codes, check the “Keywords Contain Optional Display Codes” option in “Keyword Table Properties”. To designate a character-based column to contain display codes, check the “Display Codes” option in the field properties for that field.

Keyword Tables and Configuration Management Tool (CMT): Whereas adding keyword tables, keyword fields and keyword rows automatically sync as expected using the CMT, renaming such items requires a little more care. If you rename a keyword table that has previously been synced with the CMT, you must place the prior name into the “Prior Names” property. The CMT uses this information to understand that the situation requires renaming a keyword table rather than adding a new keyword table and removing the old. Keyword table fields similarly have a “Prior Field Names” property to support renaming keyword table fields. Finally, you can create such a property for keyword rows by adding an “OldKeywords” character field column. The content of this column is used by the CMT as a prior keyword names property.

Adding a New Data Field

This section will demonstrate how to add a new data field and include it in an existing form section. Before attempting to add a field, decide what you want to name it and what the data type will be. The various data types were covered in the “Understanding the Data Dictionary” section on Page 63. For purposes of illustration, we will continue the example started in the previous section and add a field to track students’ hair color.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Important: Generally, if you are adding a new keyword selection field, make sure the necessary keyword table exists before attempting to add the keyword selection field. Adding a keyword table was described in the previous section.

Step 1: Access the configuration for the profile type for which you want to add a new data field. It is also helpful to pre-select the form section to which you want to add the data field. You can use the field in more than one form, but in this case, just pre-select the first one.

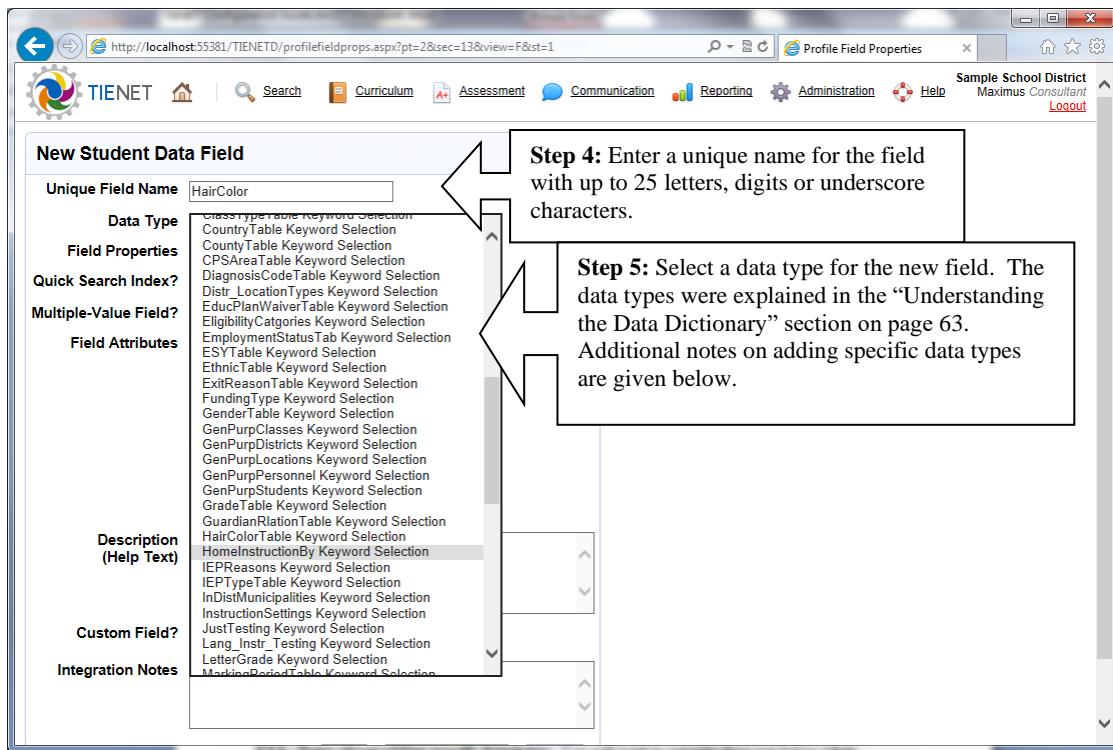
The screenshot shows the TIENET Profile Type Setup interface. The 'Demographics/Enrollment' tab is selected. On the left, under 'Student Information', there is a 'House' icon with arrows pointing to 'All Fields' and 'Constraints'. A callout box points to the 'All Fields' link. The right side of the screen displays a grid of data fields with their corresponding database column names in parentheses. The fields include:

Birth Date	: {Birthdate.R}
Case Number	: {CaseNumber.L}
Address	: {Address.L}
City	: {City.L}
State	: {State.L}
Zip Code	: {ZipCode.L}
Home Phone	: {HomePhone.L}
Resident	: {Resident.L}
Parent teaches in	: {ParentTeachesInDist.L} "Parent teaches in district"
Home Instruction By	: {HomeInstructionBy.L}
Home Instruction Hours	: {HomeInstructionHours.L}
NonPublic	: {NonPublic.L} "NonPublic"
At Risk Service	: {AtRiskService.L}
Low Income Family	: {LowIncomeFamily.L}
Migrant	: {Migrant.L}
Homeless	: {Homeless.L}

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

The screenshot shows a web browser window for 'Profile Type Setup' at the URL <http://localhost:55381/TIENET/profiletype.aspx?pt=2&sec=13&view=F>. The interface includes a navigation bar with links for Profile Types, Students, Curriculum, Administration, Help, Sample School District, Maximus Consultant, and Logout. A search bar and a 'Demographics/Enrollment' dropdown are also present. The main content area displays a table of student fields under the 'Fields' tab. A callout box labeled 'Step 2: Select the Fields tab.' points to the 'Fields' tab in the top navigation bar. Another callout box labeled 'Step 3: Click Add Data Field.' points to the 'Add Data Field' button in the toolbar above the table. The table has columns for Field Name, Data Type (Length), Properties, Attributes, Custom Field, and Description. The 'Description' column contains detailed information about each field, such as 'Contains the student's id number.' for the ID field and 'Contains the student's name.' for the LastName field.

	Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description
	ID	ID(10)	Quick Search	Key Reporting Fields	-	Contains the student's id number.
	LastName	Character(30)	Quick Search	Key Reporting Fields	-	Contains the student's name.
	FirstName	Character(30)	Quick Search	Key Reporting Fields	-	Contains the student's name.
	MiddleName	Character(30)	-	-	-	Contains the student's name.
	HomeSchool	Location Profile Reference	Organizational Parent Reference, Quick Search, Assessment Filter Field	Key Reporting Fields	-	Contains the student's home school location (the school the student would attend if not classified). [Special education users: The student's primary special education placement must be at the primary location.]
	Gender	GenderTable Keyword Selection	Quick Search, Assessment Filter Field	Key Reporting Fields	-	Contains the student's gender.



FYI – Notes about adding specific data types: You will want to consider the notes below when adding a field of the corresponding data type:

- **Character:** When adding a character field, you will need to specify a reasonable maximum length for the field. You also have the option of specifying a default value for the field.
- **Logical:** When adding a logical field, you indicate whether or not the field allows EMPTY values. A logical field that does not allow EMPTY values can be either TRUE (i.e. yes) or FALSE (i.e. no) and is normally represented by a standard checkbox. However, a logical field that allows EMPTY values can be TRUE, FALSE or EMPTY. In this case, EMPTY means no value at all (neither true nor false). This kind of logical field is represented by a dropdown menu (with all three choices) or, alternatively, two separate but linked check boxes corresponding to Yes or No. This allows for determination of whether a TRUE or FALSE value has been explicitly entered into the field.

When adding a logical field, you can optionally specify that it is a profile filter field. Each logical field that is a “profile filter field” becomes an easily selectable filtering criterion for ad-hoc reports, advanced reports, etc.

- **Integer, Numeric, Score:** When adding a data field of these types, you have the option of preventing EMPTY values. In this case, the field is always defaulted to zero or some other explicitly defined value.

The screenshot shows the 'Profile Field Properties' window with the following details:

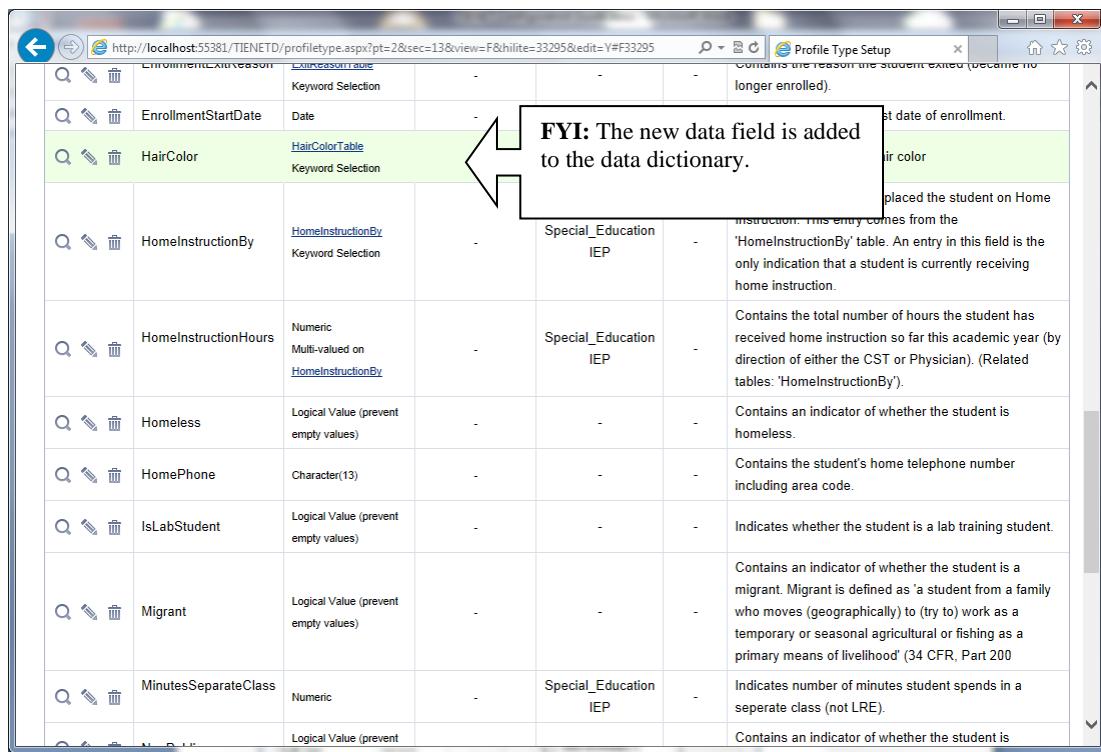
- Unique Field Name:** HairColor
- Data Type:** HairColorTable Keyword Selection
- Default Value:** (dropdown menu)
- Field Properties:** Internal Value Only (unchecked)
- Quick Search Index?**: No (unchecked)
- Multiple-Value Field?**: No (unchecked)
- Field Attributes:**
 - Weak Field Security Attributes: Basic_Skills, Special_Education, IEP, Bilingual
 - Strong Field Security Attributes: Sensitive
 - Reporting Attributes: Key Reporting Fields
- Description (Help Text):** Identifies the student's hair color
- Custom Field?**: Yes (radio button)
- Integration Notes:** (empty text area)

Step 6: Specify the rest of the properties as described in the notes below, and click the "Save" button.

FYI - About data field properties:

- Default Value:** Establishes the default value of the field for new profiles. Additionally, when a field is added, the field will assume the default value in existing profiles.
- Field Properties:** These properties can vary based on the data type. The “internal value only” option is generally available for all data types and indicates a field that is used internally but will not appear on any profile form section. PowerSchool Special Programs will suggest fields for deletion that do not appear on any form, and setting this option protects the field from being suggested.
- Quick Search Index:** If you designate the field as a “Quick Search” field, it will appear on the quick search form. Additionally, PowerSchool Special Programs will create a database index for the field to improve the performance of searches based on it. However, maintaining the index incurs a small performance penalty when profiles of this type are added, edited or deleted. If you want to have a quick search index on a field, but omit it from the quick search form, you can assign the “Omit from Quick Search Form” field property.
- Multiple-Value Field?** If you were adding a multiple-value field, you would use this dropdown to identify the keyword table that delineates the values. Multiple-value fields potentially take up a lot of space and should be used very sparingly.

- Is Profile Tag?** This property only appears for logical fields or keyword fields, and allows fields to be configured as special “profile tags” that appear to the user as status indicators or alerts for profiles of that type. For more information, see the “Configuring Profile Tags” section on page 120.
- Field Attributes:** These are customizable attributes can be associated with data fields to establish field-level security and forms of categorization. Attributes and their uses are covered in more detail in a later section.
- Description (Help Text):** This is a more detailed description of the field used for help text.
- Custom Field:** This option allows you to specify whether the field is customization specific to a school district, or is part of a state model database schema.
- Integration Notes:** Optional notes used, for example, to describe the authoritative source of this field data.



The screenshot shows a Microsoft Internet Explorer window titled "Profile Type Setup" with the URL <http://localhost:55381/TIENETD/profiletype.aspx?pt=2&sec=13&view=F&hidite=Y#F33295>. The page displays a table of profile fields. A tooltip box with the text "FYI: The new data field is added to the data dictionary." is overlaid on the "HairColorTable" row, which is highlighted in green. The table rows represent various profile fields with their descriptions and data types.

	EnrollmentExitReason	Keyword Selection				Contains the reason the student exited (became no longer enrolled).
	EnrollmentStartDate	Date				Start date of enrollment.
	HairColor	HairColorTable Keyword Selection				Hair color
	HomeInstructionBy	HomeInstructionBy Keyword Selection		Special_Education IEP		placed the student on Home instruction. This entry comes from the 'HomeInstructionBy' table. An entry in this field is the only indication that a student is currently receiving home instruction.
	HomeInstructionHours	Numeric Multi-valued on HomeInstructionBy		Special_Education IEP		Contains the total number of hours the student has received home instruction so far this academic year (by direction of either the CST or Physician). (Related tables: 'HomeInstructionBy').
	Homeless	Logical Value (prevent empty values)				Contains an indicator of whether the student is homeless.
	HomePhone	Character(13)				Contains the student's home telephone number including area code.
	IsLabStudent	Logical Value (prevent empty values)				Indicates whether the student is a lab training student.
	Migrant	Logical Value (prevent empty values)				Contains an indicator of whether the student is a migrant. Migrant is defined as 'a student from a family who moves (geographically) to (try to) work as a temporary or seasonal agricultural or fishing as a primary means of livelihood' (34 CFR, Part 200).
	MinutesSeparateClass	Numeric		Special_Education IEP		Indicates number of minutes student spends in a separate class (not LRE).
		Logical Value (prevent empty values)				Contains an indicator of whether the student is

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 7: Next you will want to add the field to the profile form section. Select the Output Format tab to bring up the form.

Step 8: Click Edit HTML Format.

The screenshot shows the 'Demographics/Enrollment' section of the profile form. It includes fields for Student Name, Birth Date, Middle Name, Last Name, Birth Date, Age, Gender, and Ethnicity. Below this is the 'Student Enrollment Information' section, which includes fields for District Enrollment Start Date, District Enrollment Exit Date, District Enrollment Exit Reason, Home School, Grade, On Roll Time, Educ Plan Waivers, Received From, Case Number, Address, City, State, Zip Code, Home Phone, Resident, Parent teaches in district, Home Instruction By, Home Instruction Hours, NonPublic, At Risk Service, Low Income Family, Migrant, and Homeless.

Step 9: When the HTML format comes for editing, click the “Fields” button to access a list of fields.

The screenshot shows the HTML code for the 'Demographics/Enrollment' section. It includes a table for student information and a table for enrollment details. A callout points to the right side of the code area, where a box contains the following text:

FYI: A full explanation of the HTML format is outside the scope of this section; however, you can refer to the “HTML Formatting Tutorial” chapter or “HTML Format Directives Reference” for more information.

```

<TABLE WIDTH=90%>
<TR>
<TD WIDTH=50% CLASS=VERYSMALL ALIGN=left><B>Student Name:</B> {FirstName.R} {LastName.R}</td>
<TD WIDTH=50% CLASS=VERYSMALL ALIGN=right><B>Birth Date:</B> {Birthdate.R}</td>
</tr>
</table>

<table BORDER=1 CLASS=DATAFORM CELLSspacing=0 CELLpadding=4 WIDTH="90%">
<tr>
<td colspan=2 class="FIELDGRIDHEADING">Student Information</td>
</tr>
<tr>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSspacing=0 CELLpadding=2>
<tr>
<td>{ID.KL}</td>
<td>{FirstName.KL}{MiddleName.L}{LastName.KL}</td>
<td>{BirthDate.L}{Age.L}</td>
<td>{Gender.IL}{Ethnic.L}</td>
</tr>
</table>
</td>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSspacing=0 CELLpadding=2 WIDTH="100%">
<tr>
<td>{CaseNumber.L}</td>
<td>{Address.L}{City.L}{State.L}{ZipCode.L}</td>
<td>{HomePhone.L}</td>
</tr>
</table>
</td>
</tr>
<tr>
<td colspan=2 class="FIELDGRIDHEADING">Student Enrollment Information</td>
</tr>

```

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Select Field - Windows Internet Explorer
http://localhost:7966/TIENETD/selectfield.aspx?field=ctl00_TNPageContent_txtHtmlFormat&pt=2&sec=65&prefix={&suffix=;L}

Select:	Student Field	Filter By Section:	Demographics	Use Form...
Name	Data Type	Attributes	Description	
ID	ID	Reporting Fields	Contains the student's id number.	
LastName	Character		Contains the student's name.	
FirstName	Character		Contains the student's name.	
HomeSchool	Location Reference		Contains the student's home school location (the school the student would attend if not classified). [Special education users: The student's primary special education placement must be at the primary location.]	
Gender	Keyword Selection		Contains the student's gender.	
Grade	Keyword Selection		Contains the grade level of the student. Entry comes from the 'GradeTable'.	
BirthDate	Date		Contains the student's birthdate.	
CaseManager	Staff Reference			
PrimaryLocation	Location Reference			
Ethnic	Keyword Selection			
FreeReducedLunch	Keyword Selection			
Address	Character	Required		
HairColor	Keyword Selection			

Step 10: A popup list of fields appears which only includes those fields linked to the profile form section you are editing. Click the name of the field you added to insert it.

Note: If a field is not listed, it is likely not linked to this particular profile form section. To link a field to the form, you must edit the section's properties and mark the checkbox corresponding to the field.

Edit XHTML for Demographics/Enrollment: [Save, Done Editing](#) [Save, Continue Editing](#) [Cancel](#)

Symbols... Directives... Fields... Functions...

```
{CaseNumber:L} {Address:L}{City:L}{State:L}{ZipCode:L}
{HomePhone:L}
</table>
</td>
</tr>
<td colspan=2 class="FIELDGRIDHEADING">Student Enrollment Information</td>
</tr>
<tr>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSPACING=0 CELLPADDING=2 WIDTH="100%">
{EnrollmentStartDate:KL}"District Enrollment Start Date"
{EnrollmentExitDate:L"District Enrollment Exit Date"
{EnrollmentExitReason:L"District Enrollment Exit Reason"
{HomeSchool:L}{Grade:KL}{OnRollTime:KL}
{EducPlanWaivers:L}
{ReceivedFrom:L}{ReceivedTime:L}{ReceivedTuition:L"Tuition received from sending district"
</table>
</td>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSPACING=0 CELLPADDING=2 WIDTH="100%">
{Resident:L}{ParentTeachesInDist:L"Parent teaches in district"
{HomeInstructionBy:L}{HomeInstructionBy:L"Home instruction by"
{AtRiskService:L}{LowIncome:L}
</table>
</td>
</tr>
<tr>
<td>{HairColor:L}</td>
```

Step 11: A directive for the new field has been inserted at the end of the format. However, you will want to cut and paste the directive to a more appropriate point in the format as shown in the next illustration.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 13: Click the “Save, Done Editing” button to save the format.

Save, Done Editing

```
<TABLE WIDTH=90%>
<TR>
<TD WIDTH=50% CLASS=VERYSMALL ALIGN=left><B>Student Name:</B> {FirstName.R} {LastName.R}</td>
<TD WIDTH=50% CLASS=VERYSMALL ALIGN=right><B>Birth Date:</B> {Birthdate.R}</td>
</tr>
</table>

<table BORDER=1 CLASS=DATAFORM CELLSPECING=0 CELLSPADDING=4 WIDTH="90%">
<tr>
<td colspan=2 class="FIELDGRIDHEADING">Student Information</td>
</tr>
<tr>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSPECING=0 CELLSPADDING=2 WIDTH="100%">
<tr>
<td>{ID.KL}</td>
<td>{FirstName.L}{MiddleName.L}{LastName.L}</td>
<td>{BirthDate.L}{Age.L}</td>
<td>{Gender.L}{Ethnic.L} {HairColor.L}</td>
</tr>
</table>
</td>
<td>
<td valign=top>
<table class="FIELDGRID" BORDER=1 CELLSPECING=0 CELLSPADDING=2 WIDTH="100%">
<tr>
<td>{CaseNumber.L} {Address.L}{City.L}{State.L}{ZipCode.L}</td>
<td>{HomePhone.L}</td>
</tr>
</table>
</td>
</tr>
<tr>
<td colspan=2 class="FIELDGRIDHEADING">Student Enrollment Information</td>
</tr>

```

Save, Done Editing Save, Continue Editing Cancel

Step 12: Paste the directive at an appropriate point relative to other directives in the format.

Step 14: Note that the new field is now part of the form.

Profile Types Students

Demographics/Enrollment Output Format XHMTL Format Testing Format Fields Data Flow Report Constraints

Edit XHMTL Format Lock Section Edit Section... Print More...

Student Name: Birth Date:

Student Information

ID*	
First Name*	
Middle Name	
Last Name*	
Birth Date*	
Age	
Gender*	(none) <input type="button" value="▼"/>
Ethnic*	(none) <input type="button" value="▼"/>
Hair Color	(none) <input type="button" value="▼"/>

Student Enrollment Information

District Enrollment Start Date*	
District Enrollment Exit Date	
District Enrollment Exit Reason	(none) <input type="button" value="▼"/>
Home School*	lookup <input type="button" value="..."/>
Grade*	(none) <input type="button" value="▼"/>

Case Number
Address
City
State (none)
Zip Code
Home Phone

Resident
Parent teaches in district
Home Instruction By (none)
Home Instruction Hours
Child Study Team
Physician

Adding Profile Constraints

A profile constraint is a rule that validates data, and if there is a problem, either prevents the data from being accepted or presents a warning to the user who is performing data entry. If you want to implement a “required field”, do NOT use a constraint for this purpose. It is easier and much more efficient to simply add the required field (!) modifier to the field directive in the HTML format, e.g. {Ethnicity:L!}. Constraints are designed to implement more complicated rules, beyond required fields.

Step 1: As illustrated in previous sections, access the specific profile section to which you want to add a constraint, and make sure a configuration task is selected. Then click the Constraints tab.

The screenshot shows the TIENET Profile Type configuration interface. The URL is <http://localhost:60054/TIENET/profiletype.aspx?pt=2>. The top navigation bar includes links for Profile Types, Students, Setup, Verify All, Add New Section, More..., Demographics/Enrollment, Output Format, XHTML Format, Testing Format, Fields, Data Flow Report, and Constraints (which is highlighted). Below the navigation is a toolbar with Edit XHTML Format, Lock Section, Edit Section..., Print, and More... buttons. The main content area displays two tables: "Student Information" and "Student Enrollment Information". The "Student Information" table contains fields for ID, First Name, Middle Name, Last Name, Birth Date, Age, Gender, Ethnic, and Red Alert. The "Student Enrollment Information" table contains fields for District Enrollment Start Date, District Enrollment Exit Date, District Enrollment Exit Reason, Home School, Grade, On Roll Time, Educ Plan Waivers, Received From, Received Time, Case Number, Address, City, State, Zip Code, Home Phone, Resident, Parent teaches in, Home Instruction By, Home Instruction Hours, NonPublic, At Risk Service, Low Income Family, Migrant, and Homeless.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 2: Click Add New Constraint.

	Name	Evaluated When (For)	Action Taken	User Message	Custom?
	Birth Date Required	Section Edited (Do not evaluate for ADMIN/CONSULTANT)	Warn User	Student must have a birth date for state reporting	No

Step 3: Enter a unique name for the constraint. This is for identification purposes only. The end user will not see this name.

Step 4: By default, the constraint is evaluated whenever the user attempts to save any form section to which the constraint is linked. You can use these options to set specific conditions under which the constraint is evaluated (see FYI below).

FYI: Here you can specify that the constraint will only be evaluated for members of a specified security group, or that it should not be evaluated for the ADMIN and CONSULTANT.

Unique Name: Invalid Birthdate

Is Evaluated: Section edited
 Section edited, no previously evaluated "stop data submission" constraints have been triggered
 Section edited, no previously evaluated "warn user" or "stop data submission" constraints have been triggered
 Section edited, the "preceding" constraint (selected below) has NOT been triggered
 Section edited, the "preceding" constraint (selected below) has been triggered
 Preceding Constraint: N/A

When...:
 User explicitly clicks button
 User explicitly clicks button (with edit security bypass)
 Bulk operation (with edit security bypass)
 Profile about to be deactivated
 Profile about to be reactivated
 Profile about to be deleted

Is Evaluated For...: User in security group: Any/All
 Do not evaluate for ADMIN/CONSULTANT

Trigger Formula: Student Formula
 Select Field: BirthDate
 Functions: Functions
 BirthDate > AddYears(GetDate(), -3)
 Anchor constraint message to first field that is referenced in formula (applies to section editing constraints only)

Action Taken: Warn User

User Message: (Example: "The end date cannot be before the start date.")

FYI – About conditional constraint evaluation: You can specify if and when a constraint is evaluated. First, when multiple constraints are linked to a section, you can specify the order of the constraints as explained in the last step of this procedure. Additionally, you can specify that a constraint will only be executed if:

- no previously evaluated “stop data submission” constraints have been triggered.
- no previously evaluated “warn user” or “stop data submission” constraints have been triggered.
- the specified “preceding” constraint has not been triggered (note: the preceding constraint is selected from the dropdown)
- the specified “preceding” constraint has been triggered
- when the user explicitly clicks a button (see FYI – About Constraints Buttons later in this section)

The screenshot shows the 'Student Constraint Properties' page with the following details:

- Unique Name:** Invalid Birthdate
- Is Evaluated:** Section edited (radio button selected)
- When...**: Section edited, no previously evaluated "stop data submission" constraints have been triggered
- Action Taken:** Warn User (selected from a dropdown menu)
- User Message:** cannot be below the start date
- Trigger Formula:** Student Formula (dropdown selected)
 - Select Field: BirthDate
 - Functions: AddYears(GetDate(), -3)
- Is Evaluated For...**: User in security group (dropdown selected)
 - Up: Any/All
 - Do not check: ADMIN/CONSULTANT
- Custom?**: No (radio button selected)
- Applies to Sections:** Demographics/Enrollment (checkbox checked), Contact Info, Services, ProfileGrid, Promotion Status Report, Testing, Test Dates, Eligibility, Medical (checkboxes unselected)

Step 6: Enter the criteria that specifies when the constraint is triggered. This formula should be true when the constraint should be executed and false when there is not. By default, constraint messages are displayed above the form. Use the “Anchor Constraint Message” checkbox below the formula to display the message next to the first form field referenced in the trigger formula.

Step 7: Select the action to be taken when the constraint is triggered (see FYI below).

FYI – About Constraint Actions: A constraint can take one of the following actions when triggered:

- **Warn User:** When this action is triggered, the message you specify is presented to the user, but the data is accepted (unless data acceptance is stopped by another constraint)

- **Stop Data Submission:** When any constraint of this type is triggered, no changes are made to the database and you can specify a message to be presented to the user.
- **Set Field Values:** This kind of constraint, when triggered, sets one or more field values. The field values being set in the constraint can be calculated based on the new, updated profile field values, or even based on new values being applied by a previously triggered “Set Field Values” constraint. If a “Stop Data Submission” constraint is later executed, the changes will not be applied. The optional message is useful for debugging purposes, but is generally undesirable in the end, because if a user message is generated, and then later a “Stop Data Submission” constraint is executed, the message will still appear, but the set values will not be applied.
- **Deactivate Profile:** Deactivates the current profile.
- **Reactive Profile:** Reactivates the current profile.
- **Go To Other Section:** When a constraint of this type is triggered, the user switches to the specified section provided that no “Stop Data Submission” constraints have been triggered. A message presented to the user is optional.
- **Insert Child Profile.** Inserts a new child profile from the context of either a top level profile or another type of child profile that shares the same top level profile. When inserting a child profile from a top level profile, the fields can be set from the values of the top level profile. When inserting a child profile from another type of child profile, the fields can be set from values of either the originating child profile and/or the top level profile.
- **Update Child Profiles:** Updates specified field value(s) in specified child profiles of the current profile.
- **Delete Child Profiles:** Deletes specified field value(s) in specified child profiles of the current profile.
- **Delete Child Profiles:** Set field value(s) in a profile that is referenced directly or indirectly from the current profile. For example, if there is a staff profile reference field in the student profile, a constraint can now be configured to set field values in the staff profile. The “target profile criteria” is typically just the name of a profile reference field.
- **Insert Child Profile as Secondary Parent [New in 22.6.1.0]:** This is very similar to the “Insert Child Profile” constraint, but with a key difference. “Insert Child Profile as Secondary Parent” constraint allows you to insert a new child profile for any child profile type for which the current top level profile type is the secondary parent. For example, you can configure a student profile constraint with ‘Insert Child Profile as Secondary Parent’ to insert the student into the caseload for a specified staff user (which is impossible using “Insert Child Profile”). You can also configure such a constraint in a child profile of students to insert the student in the caseload for a specified staff user.

- **Set Field Values in Referenced Profile:** Allows for setting field value(s) in a profile that is referenced directly or indirectly from the current profile. For example, if there is a staff field in the student profile, this type of constraint can be used to set field values in the staff profile. The “target profile criteria” is typically just the name of a profile reference field that identifies the target profile that will have its field values set.
- **Remove Active Status from All Documents:** If the profile has any “active” documents (e.g. active IEP for a student profile), the active status will be removed.
- **Create Document:** Creates a new document of the specified document template and lands the user in the new document for editing. The trigger criteria should typically check if a document already exists to avoid creating multiple documents. Furthermore, the constraint will only trigger if the current user is authorized to create the document. This type of constraint is not available for bulk operation constraints.
- **Send Message [New in 22.6.1.0]:** Sends a specified message when triggered. When configuring a “Send message” profile constraint, you specify a message subject and body (which both support the same macros as the user message) and can optionally mark the message as high priority. To prevent sending overly frequent messages, it is often helpful to pair this section action with another action that sets some type of flag that indicates that a notification is needed. The “Send Message” constraint supports an optional “field set” expression that can be configured to clear the flag after sending the message. Alternatively, the “Send Message” constraint can be directly linked to a preceding constraint via the “Preceding Constraint” property, which is visible when “Is evaluated when...” is set to “Section edited, the preceding constraint has been triggered”.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

Step 8: Enter the message that should be displayed to the end user when the constraint is triggered by a data problem.

Action Taken: Warn User

User Message: Please check the birthdate field.

Custom? Yes

Applies to Sections: Demographics/Enrollment

Step 9: By default the constraint is applied to the profile form section you selected earlier.

Step 10: Click "Accept" to save the constraint.

Step 11: The constraints linked to a particular form section are listed in the order they will be evaluated and displayed. After adding the new constraint, you can use these arrow icons to change the order of evaluation. Alternatively, you can drag and drop the constraints into the desired order.

Student constraints linked to 'Demographics/Enrollment'					
	Name	Evaluated When (For)	Action Taken	User Message	Custom?
	Birth Date Required	Section Edited (Do not evaluate for ADMIN/CONSULTANT)	Warn User	Student must have a birth date for state reporting	No
	Invalid Birthdate	Section Edited	Warn User	Please check the birthdate field.	No

Verify Existing Profile Constraints: If you have made a lot of field changes, it is advised that you verify the constraints to make sure they are all still valid. To do this, bring up the list of constraints and click Verify Constraints. You can check them a section at a time, or you can check them all at once by first clicking All constraints.

Advanced Anchoring of Constraints: In addition to automatically anchoring a constraint to a field (referenced in the trigger formula), you can also precisely position where a constraint is displayed in the form. To do this, embed in the HTML format a directive in the form `{*constraintname}`.

About Macros [New in 22.6.1.0]: When configuring profile constraints, macros are supported in the “User Message” constraint property, allowing for dynamic user messages that include the values of fields. For student profile constraints, the following macros are supported by default: {ID}, {FirstName}, {LastName}, {MiddleName} and {FirstLastName}. You can make additional fields available for use in macros by setting the ‘Always Available for Macros’ profile field property for those fields. When configuring a profile constraint, the available macros are listed above the “User Message” field as links that can be clicked for easy insertion. Only macros that are listed will work.

About Constraint Buttons: For constraints that have been configured to be evaluated when the user clicks a button, you need to place a button directive in the profile section HTML layout that references the constraint using the format `{*constraintname}`. By default, the button appears only in edit mode, uses the name of the constraint as the button label, and uses “CONSTRAINTBUTTON” as the CSS class of the button (you can supply a style for this in the style sheet). However, additional modifiers allow you to customize the button directive further.

- `{*constraintname:V}` shows the button only in view mode, not edit mode.
- `{*constraintname:EV}` shows the button in both view and edit mode.
- `{*constraintname:L"alternate label"}` specifies a specific label for the button (other than the constraint name). Note that if you are translating the form into alternative language, you should provide an explicit label for all buttons so that the label can be translated.
- `{*constraintname:C"classname"}` specifies a CSS class other than the default of “CONSTRAINTBUTTON”.

Various combinations of the above can be used. The modifiers can appear in any order.

The button will not appear at all if a security group has been configured for the constraint and the user is not in the security group, and will not appear for ADMIN or CONSULTANT if the constraint is configured to be skipped for ADMIN and CONSULTANT. The button will appear regardless of whether or not the trigger formula is met, since the trigger formula is only checked when there is an attempt to evaluate and execute the constraint. If the constraint is a “field set” constraint and the button is designed to appear in view mode, the button will appear only if either the user either has edit access to the profile, or if the constraint is configured to have the “edit security bypass” option.

When the form is in edit mode and the user clicks a constraint button, the form is saved as if the user clicked “Accept Changes”; however, the constraint for the button is executed along with any other constraints that are normally executed.

When the form is in view mode and the user clicks a constraint button, only that constraint is executed. Normally, such a constraint would be a “field set” constraint since the other action types will have little effect other than showing a message or switching to another section. If a field set constraint is executed, an entry is made in the audit log.

FYI – Using Constraints for Interactive “Bulk” Data Modifications: Profile update scripts are a useful feature to apply pre-defined data modifications to profiles in bulk on either a scheduled or manually triggered basis; however the set of profiles a script operates on are determined strictly using a formula. When interactive control over the set of profiles is needed, a profile constraint evaluation method “Bulk Operations (edit security bypass)” is very useful. This feature can be configured to enable users to apply a pre-defined data modification via a quick/advanced search or a list report using checkboxes to control which profiles are in the affected set

First you create the profile constraint, which must meet the following criteria: 1) evaluated for “Bulk operation with (edit security bypass)”, 2) is specifically targeted to the security group for which to enable this capability, and 3) set field values in the profile. This is enough to allow the user to apply the constraint to selected profiles via the quick/advanced search screen. Note that regardless of which profiles the user selects, the data modification still will only be applied to profiles that meet the trigger criteria defined in the constraint. Additionally, the data modification will be applied even if such users are not authorized to modify the profiles directly.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

The screenshot shows the 'New Student Constraint' configuration page. Key fields include:

- Unique Name:** Approval Process
- Is Evaluated When...**:
 - Section edited
 - Section edited, no previously evaluated "stop data submission" constraints have been triggered
 - Section edited, no previously evaluated "warn user" or "stop data submission" constraints have been triggered
 - Section edited, the "preceding" constraint (selected below) has NOT been triggered
 - Section edited, the "preceding" constraint (selected below) has been triggered
- Preceding Constraint:** N/A
- Is Evaluated For...**:
 - User explicitly clicks button
 - User explicitly clicks button (with edit security bypass)
 - Bulk operation (with edit security bypass)
 - Profile about to be deactivated
 - Profile about to be reactivated
 - Profile about to be deleted
- User in security group:** Provider Supervisors
- Action Taken:** Set Field Values
- Syntax:** Field1=Value1, Field2=Value2
- Trigger Formula:** Student Formula: Approved=False

Annotations with arrows point to specific fields:

- An arrow points from the text "Choose ‘Bulk operation’ here." to the radio button for "User explicitly clicks button (with edit security bypass)".
- An arrow points from the text "Choose the target security group here." to the dropdown menu for "User in security group".
- An arrow points from the text "Choose ‘Set Field Values’ here." to the dropdown menu for "Action Taken".

Occasionally, you may want to use a constraint both for bulk operations and for use as a button as described in the previous section. Note the profile constraints configured as a bulk operation can also be referenced and used as a button.

Once the constraint is defined, you can also allow users to apply the constraint's data modification via a list report. Create the list report, authorize the target security group to access the list report, and also enable the following options in Report Properties: Report is Under "Configuration Management" Control and "Enable User to Apply Data Modification Operations via This Report".

Ownership/Category	Public Report -> Optional Category: <input type="text" value="Case Manager Reports"/> <input type="button" value="▼"/>
	<input checked="" type="checkbox"/> Report Is Under "Configuration Management" Control
Data Refresh Options	<input type="checkbox"/> Auto Refresh System-Wide Report View Overnight, Also At: <input type="text" value="N/A"/> <input type="button" value="▼"/> Data can be as old as: <input type="text" value="5 Minutes"/> <input type="button" value="▼"/> (Not applied to auto-refreshed system-wide report view)
	<input checked="" type="checkbox"/> Allow Manual Refreshing
	<input type="checkbox"/> Execute on Mirror Database
Output Options	<input type="checkbox"/> Allow Generate As PDF <input type="checkbox"/> Allow Users to Render Report Directly On Their Home Page Highlight Report Name in Report Menus Using Color: <input type="text" value="□"/> <input type="button" value="▼"/> <input checked="" type="checkbox"/> Enable Users to Apply Data Modification Operations via This Report
Editing Control Options	<input type="checkbox"/> Hide Data When Editing Report <input type="checkbox"/> Allow Users To Make Private Copies of Report
<input type="button" value="Accept"/> <input type="button" value="Cancel"/>	

Adding an “Automatic” Profile Form Section

You can create a profile form section that has no HTML format, but rather simply includes all data fields that meet certain criteria. Of course, with such a form, you will have no control over the look and feel of the form, but it can be convenient for certain purposes. For example, you can create an “automatic” profile form section that automatically includes any “custom” data fields that the school district’s ADMIN has added. To add an “automatic” profile form section, follow the steps below:

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

The screenshot shows the TIENET Profile Type Setup interface. In the center, there's a form titled "Student Information" with fields for ID, First Name, Middle Name, Last Name, Birth Date, Age, Gender, Ethnic, and Red Alert. To the right of the form is a sidebar with various configuration options like Translations, Edit Style Sheet, Copy Section(s) From Other Database, Set List Columns for Student Profiles, Edit Profile Type Properties, Set Profile Field Set Expressions, Add Auto-Generated Section, View My Profile, Create Indexes, and Constraints. A callout box highlights the "Add Auto-Generated Section" option under the "More..." dropdown menu.

Step 1: After accessing the configuration for the desired profile type (and after selecting a Configuration Task), select Add Auto-Generated Section from the “More...” dropdown menu

The screenshot shows the "Add New Form Section For Students" dialog. It includes fields for Section Name (All Custom Data Fields), Category (Top-Level Section), Purpose of section (View and edit), Content generation method (Auto Generated), and checkboxes for Include Custom Fields Only? and Include Data Types. The Include Data Types section lists various field types: ID, Character, Logical, Date, Date/Time, Integer, Numeric, Score, School Year, Short Text, Long Text, Keyword Selection, and Profile Reference. A dropdown for Insert section where? is set to "Between 'Test Dates' and end". At the bottom are Accept and Cancel buttons. Callout boxes highlight the "Section Name" field, the "Content generation method" dropdown, the "Include Data Types" checkbox, and the "Accept" button.

Step 2: Specify the name, category and purpose of the form section (same as with an HTML-based section).

Step 3: To include only “custom” fields that the school district ADMIN has created, mark this checkbox.

Step 4: Choose the date types for the fields you want to include. Then click the “Accept” button at the bottom.

PROFILE DATA SCHEMA CUSTOMIZATION (BASICS)

The screenshot shows a web-based application titled "Profile Type Setup" from the "TIENET" platform. The URL is <http://localhost:60054/TIENETD/profiletype.aspx?pt=2&sec=74&view=0&edit=Y>. The top navigation bar includes links for Search, Curriculum, Assessment, Communication, Reporting, Administration, Help, and a user account for "Your School District Maximus Consultant Logout". The main menu has tabs for "Profile Types" and "Students", with "Students" currently selected. Below the menu are buttons for "Setup", "Verify All", "Add New Section", and "More...". A sub-menu for "All Custom Data Fields" is open, showing options like "Add Data Field", "Add Calculated Field", "Verify Fields", "Edit Section...", "Print", and "More...". The main content area displays a table titled "Student fields linked to 'All Custom Data Fields'". The table has columns for "Field Name", "Data Type (Length)", "Properties", "Attributes", "Custom Field", and "Description". One row is visible, showing "HasSecondaryLoc" as the field name, "Logical Value (prevent empty values)" as the data type, and "Yes" as the custom field status. The description is "Secondary location? (For IEP)". A callout box with an arrow points to this row, containing the text "Step 5: You now see a list of data fields included in the new section."

Student fields linked to 'All Custom Data Fields'						
Auto-generate section with custom student fields of type(s): ID, Date, Short Text, Long Text, Keyword Selection, Profile Reference						
	Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description
	HasSecondaryLoc	Logical Value (prevent empty values)	-	-	Yes	Secondary location? (For IEP)

Step 5: You now see a list of data fields included in the new section.

Profile Data Schema Customization

(Advanced)

Adding a Calculated Field

This section will demonstrate how to add a calculated field for a profile type. The benefit of having a calculated field is that it does not need to be entered or imported; rather it automatically calculates its values based on other data fields or even other calculated fields. It is highly likely that your database schema already has calculate fields in it such as “Age” which calculates its value using a student’s birth date and the current date. To illustrate how to add a calculated field, we will add a calculated field named “AgeAsOfSept1” which computes the student’s age as of September 1st in the current school year.

Chapter

4

The screenshot shows the 'Profile Type Setup' application window. The title bar says 'Profile Type Setup'. The menu bar includes 'File', 'Administration', 'Help', 'Your School District', 'Maximus - Consultant', and 'Logout'. The main toolbar has buttons for 'Verify All', 'Add New Section', 'More...', 'Fields', 'Data Flow Report', and 'Constraints'. Below the toolbar, there are buttons for 'Add Data Field', 'Add Calculated Field', 'Verify Fields', 'Edit Section...', and 'Print'. A navigation tree on the left shows 'Profile Type' selected, with 'Students' expanded, and 'Demographics/Enrollment' selected. A callout box labeled 'Step 1: Access the configuration for the profile type for which you want to add a calculated field. Use the flyout menu to select the specific section on which you want to display the calculated field.' points to the 'Demographics/Enrollment' item. Another callout box labeled 'Step 2: Click the "Fields" tab.' points to the 'Fields' tab in the main toolbar. A third callout box labeled 'Step 3: Make sure a configuration task is selected, and then click "Add Calculated Field".' points to the 'Add Calculated Field' button in the toolbar. The main content area displays a table of student fields. The table has columns for Properties, Attributes, Custom Field, and Description. The table includes rows for ID, LastName, FirstName, MiddleName, HomeSchool, and Gender. The 'HomeSchool' row is highlighted with a yellow background. The 'Custom Field' column for HomeSchool contains the text 'Organizational Parent Reference, Quick Search, Assessment Filter Field'. The 'Description' column for HomeSchool contains the text 'Contains the student's home school location (the school the student would attend if not classified). [Special education users: The student's primary special education placement must be at the primary location.]'.

	Properties	Attributes	Custom Field	Description
ID	ID(10)	Quick Search	-	Contains the student's id number.
LastName	Character(30)	Quick Search	-	Contains the student's name.
FirstName	Character(30)	Quick Search	-	Contains the student's name.
MiddleName	Character(30)	-	-	Contains the student's name.
HomeSchool	Location Profile Reference	Organizational Parent Reference, Quick Search, Assessment Filter Field	-	Contains the student's home school location (the school the student would attend if not classified). [Special education users: The student's primary special education placement must be at the primary location.]
Gender	GenderTable Keyword Selection	Quick Search, Assessment Filter Field	-	Contains the student's gender.

New Student Calculated Field

Unique Field Name: AgeAsOfSept1

Data Type: (Select...) Character Logical Value Date Date Time Numeric Integer Long Text School Year Image Profile Reference Class Profile Reference Staff Profile Reference Location Profile Reference District Profile Reference General Ed Student Profile Reference Keyword Selection AdditlProperties Keyword Selection AgeToGradeConvTable Keyword Selection AlternateWorksAt Keyword Selection ATestTable Keyword Selection ARiskTypeTable Keyword Selection ARiskValueTable Keyword Selection AttendanceTypeTable Keyword Selection BillingStatusTable Keyword Selection CalendarTracksTable Keyword Selection Certifications Keyword Selection ChicagoAltAssmt Keyword Selection ChicagoSubjects Keyword Selection ClassTypeTable Keyword Selection

Description (Help Text):

New Student Calculated Field

Unique Field Name: AgeAsOfSept1

Data Type: Integer

Field Properties: Internal Value Only

Multiple-Value Field?: No

Calculation Formula: Student Formula Select Field Functions
YEARDIFFERENCE(BirthDate, AddYears(HistoryYear - 1000, 0))

Field Attributes: Weak Field See... Basic Skills

(Help Text): Step 7: Enter the formula that calculates the value. The "Fields" and "Functions" buttons above allow you to insert fields and functions respectively. Preparing this formula requires an advanced knowledge of PowerSchool Special Programs formulas. The separate "Formula Reference" document may be useful. For an explanation of the formula used in this example, see the note below.

FYI – About Multiple-Value Calculated Fields: Multiple value data fields (i.e., non-calculated) store more than one value in the database, and those values are named and identified in a keyword table. A multiple-value calculated field works in an analogous way, except that instead of storing a value for each corresponding keyword, it calculates a value for each keyword. However, since you only have one formula to work with that must calculate the value for each keyword, the formula accepts a parameter that can be accessed from within the formula using the name “Selector”. So when writing the formula for a multiple-value calculated field, you assume the name “Selector” contains the keyword for which a value is being calculated.

FYI – About the “AgeAsOfSept1” formula: If you are curious about the formula used in the above example, here is an explanation. If you were just calculating the current age, you could use the formula YEARDIFFERENCE(BirthDate, CurrentDate()). The YEARDIFFERENCE function takes two parameters, an earlier date and a later date, and calculates the number of complete years between them. However, we want the age as of September 1st in the current school year. The HistoryYear name used in the formula gives the year that the current school year started as an integer number (e.g. for the 2005/2006 school year, HistoryYear would give the integer 2005). So we replace CurrentDate() with “ADDEARS(HistoryYear-2000, 9/1/2000)” which uses some basic date manipulation to come up with the specific date of September 1 in the current school year.

FYI – About “Profile Filter Fields”: When adding a logical calculated field, you can optionally specify that it is a profile filter field. Each logical field that is a “profile filter field” becomes an easily selectable filtering criterion for ad-hoc reports, advanced reports, etc. A typical example is a “NotExited” field which can be used to allow end users to easily specify that only students who have not exited are to be included in a report.

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

The screenshot shows the 'Profile Field Properties' dialog box. Key fields include:

- Multiple-Value Field?**: No
- Calculation Formula**: Student Formula YEARDIFFERENCE(BirthDate, AddYears(HistoryYear - 2000, 9/1/2000))
- Field Attributes** (Weak Field Security Attributes):
 - Basic_Skills
 - Special_Education
 - IEP
 - Bilingual
 - State Reporting
 Strong Field Security Attributes:
 - Sensitive
 Reporting Attributes:
 - Reporting Fields
- Description (Help Text)**: Calculates the student's age as of September 1st of the current school year.
- Custom Field?**: Yes No
- Integration Notes**

Buttons at the bottom: Save, Save & Add Again, Cancel.

Annotations:

- Step 8:** These customizable attributes can be associated with the new data field. Attributes are covered in a later section.
- Step 9:** Entering a description is required.
- Step 10:** This option allows you to specify whether the field is customization specific to a school district, or is part of a state model database schema.
- Step 11:** Click "Save" to add the new field.

The screenshot shows the 'Profile Type Setup' dialog box with the title 'Student fields linked to 'Demographics/Enrollment''. A table lists fields:

	Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description
<input type="button" value="Search"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	Address	Character(50)	-	Reporting Fields	-	Contains the student's address.
<input type="button" value="Search"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	Age	Calculated Numeric	Internal Value Only	Reporting Fields	-	Calculates the student's age.
<input type="button" value="Search"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	AgeAsOfSept1st	Calculated Integer (prevent empty values)	-	-	-	Calculates the student's age as of September 1st of the current school year.
<input type="button" value="Search"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>	assa	School Year	Calculated AtRiskValue	-	-	dssdsdssdds
<input type="button" value="Search"/> <input type="button" value="Edit"/> <input type="button" value="Delete"/>						Test
						Contains indicators of whether or not the student receives free or reduced milk or meals (under the "National School Lunch" or "Child Nutrition" acts). (Related tables: 'AtRiskTypeTable').

Annotations:

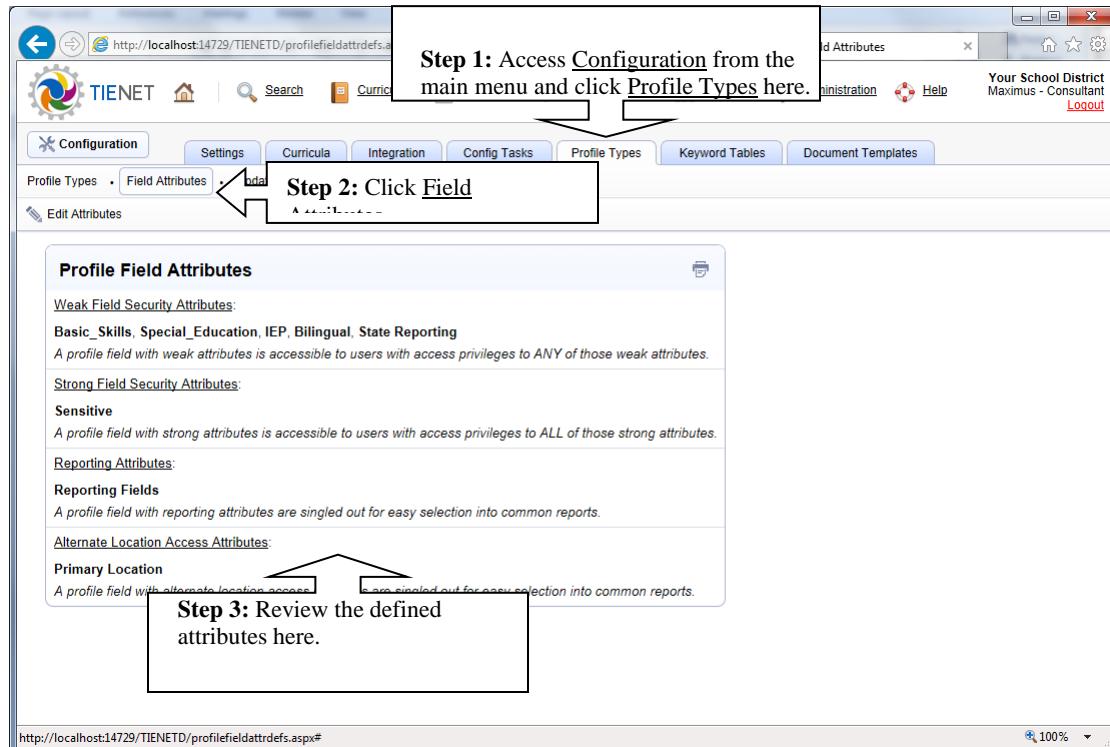
- Step 13:** The calculated field is added to the data dictionary. You can place the new calculated field into a profile form section the same way you do with a data (non-calculated) field as described in the "Adding a New Data Field" on page 78.

Profile Field Attributes

Profile field attributes are named attributes that can be defined and applied to any and all fields in the data dictionary to control various aspects of security and access. The illustration below shows how attributes can be applied to a new data field being added to the schema.

The screenshot shows a web-based application interface titled "Profile Field Properties". The main title bar includes the URL "http://localhost:14729/TIENETD/profilefieldprops.aspx?pt=2&sec=13&view=F&st=1", the page title "Profile Field Properties", and user information "Your School District Maximus - Consultant Logout". The left navigation menu has links for Search, Curriculum, Assessment, Communication, Reporting, Administration, Help, and Logout. The main content area is titled "New Student Data Field". It contains several input fields: "Unique Field Name" (empty), "Data Type" (dropdown menu "Select..."), "Field Properties" (checkbox "Internal Value Only" checked), "Quick Search Index?" (dropdown menu "No" selected), "Multiple-Value Field?" (dropdown menu "No" selected), and "Description (Help Text)" (text area). Below these are sections for "Field Attributes": "Weak Field Security Attributes" (checkboxes for Basic_Skills, Special_Education, IEP, Bilingual, State Reporting), "Strong Field Security Attributes" (checkbox for Sensitive), and "Reporting Attributes" (checkbox for Reporting Fields). At the bottom are "Custom Field?" (radio buttons for Yes and No, Yes selected) and "Integration Notes" (text area). A callout box with a black border and a white background points from the right towards the "Field Attributes" section. Inside the callout box is the text: "Field Attributes: When adding or modifying fields, you can apply profile field attributes with those fields."

To view existing profile field attributes and their uses, follow the steps below:



In the illustration above, there are three categories of profile field attributes as follows:

Weak Attributes: These are utilized for a less restrictive form of field-level security. If a field has one or more weak attributes linked to it, any user who has been granted access to at least one of the corresponding access privileges can access the field.

Strong Attributes: These are utilized for a more restrictive form of field-level security. If a field has one or more strong attributes linked to it, a user can only access the field if the user has all the corresponding access privileges. For example, if a field has associated attributes of “financial” and “highly sensitive”, the user must have both of those corresponding privileges to access the field.

Reporting Attributes: An end user who is building an ad-hoc report can filter the data dictionary by reporting attribute. Use of reporting attributes can make it much easier for end users to build reports quickly.

Weak and strong attributes, which are used to implement field-level security, appear in the access privileges for security groups as seen in the illustration below:

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

FYI: The “Field Level Security Privileges” determine whether members of the security group can view or edit profile fields with the corresponding attribute.

The screenshot shows the Security Group Properties page with the following sections and privileges:

- Security Management Privileges:**
 - Create Edit Public List Reports: n/a
 - Create Edit Public Multi Dimensional Reports: n/a
 - Create Edit Public Group Progress Reports: n/a
 - Admin Public Reports: n/a
 - View Advanced Reports: n/a
 - Process Unprocess Advanced Reports: n/a
- Field Level Security Privileges:**

View Basic_Skills Fields	n/a
View Special_Education Fields	n/a
View IEP Fields	n/a
View Bilingual Fields	n/a
View State Reporting Fields	n/a
View Sensitive Fields	n/a

Edit Basic_Skills Fields	n/a
Edit Special_Education Fields	n/a
Edit IEP Fields	n/a
Edit Bilingual Fields	n/a
Edit State Reporting Fields	n/a
Edit Sensitive Fields	n/a
- Alternate Location Profile Access Privileges:**

View Students	Grant (+) System-Wide
Add Students	Grant (+) System-Wide
Edit Students	Grant (+) System-Wide
Delete Students	Grant (+) System-Wide
Deactivate	Grant (+) System-Wide
Reactivate	Grant (+) System-Wide
Access Documents	Grant (+) System-Wide

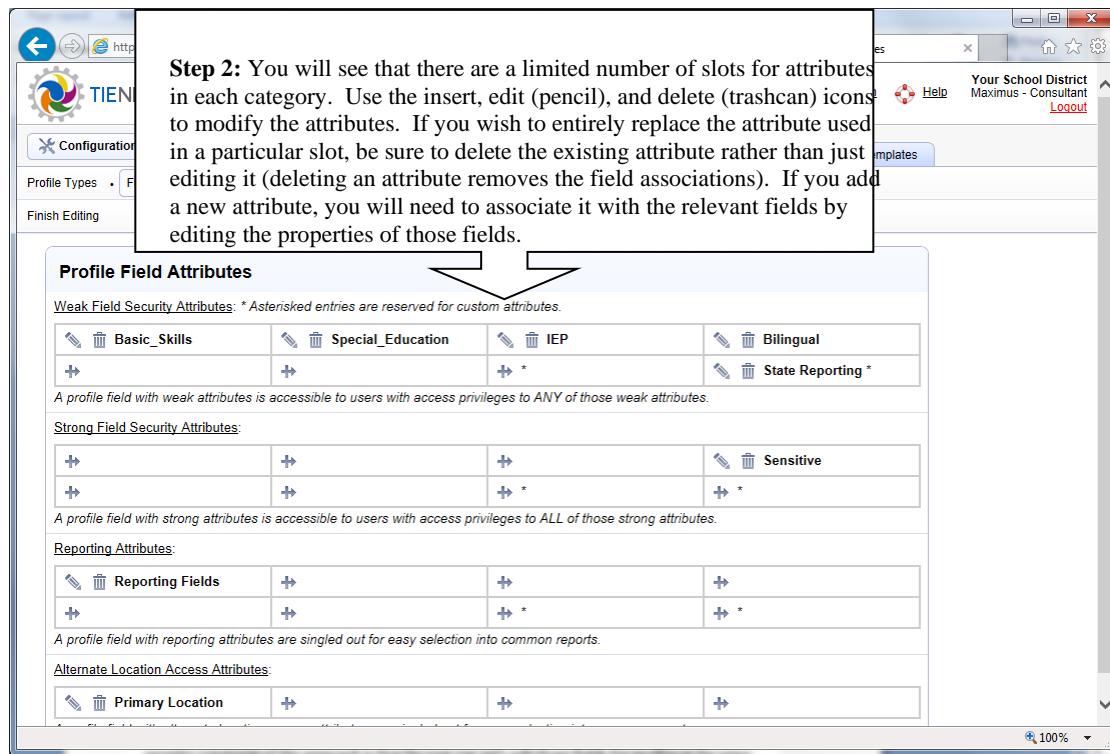
Maintain Own File Based Documents	n/a
Edit File Based Documents From Other Users	n/a
Unfinalize File Based Documents	n/a
Undelete Documents	n/a
View Service Records	n/a
Add Service Records	n/a
Edit Service Records	n/a

To customize field attributes, follow the procedure below:

Step 1: To modify the attributes, make sure you have a configuration task selected, and then click Edit Attributes.

The screenshot shows the Field Attributes configuration page with the following sections:

- Weak Field Security Attributes:** Basic_Skills, Special_Education, IEP, Bilingual, State Reporting
- Strong Field Security Attributes:** Sensitive
- Reporting Attributes:** Reporting Fields
- Alternate Location Access Attributes:** Primary Location



FYI – Enabling restricted editing access to a small subset of fields: There may be cases when you want to give a particular security group editing access to a few fields, but otherwise no editing access to the overall profile. In this case, define a new weak attribute, associate it with the fields, and then give the security group editing access to fields with that attribute, but no overall editing access to the profile. A security constraint of this approach is that the user can only edit those fields for profiles at the same security location.

Creating a Child Profile Type

“Child” profile types enable you to maintain multiple sub-records associated with a main profile record. For example, you might want to keep a set of information on suspensions, and since a student could have more than one suspension, you could create a new “child” profile type called “Suspensions” to track them. In this case, “Suspensions” is a child profile type of the Students profile type, and it would maintain information on multiple suspensions per student.

In this section, we will create a child profile type to track “Suspensions”. Specifically, we will:

Create the “Suspensions” child profile type in such a way that it always sorts with the most recent suspensions at the top.

Add additional fields needed to maintain complete information on each suspension.

Create a form section for viewing and entering suspensions.

Establish how the suspensions will be displayed when shown in list format (by selecting suspension fields and setting the order of those fields).

Establish a unique key so that a student can only have one suspension per day (as an additional data integrity check).

Follow the steps below:

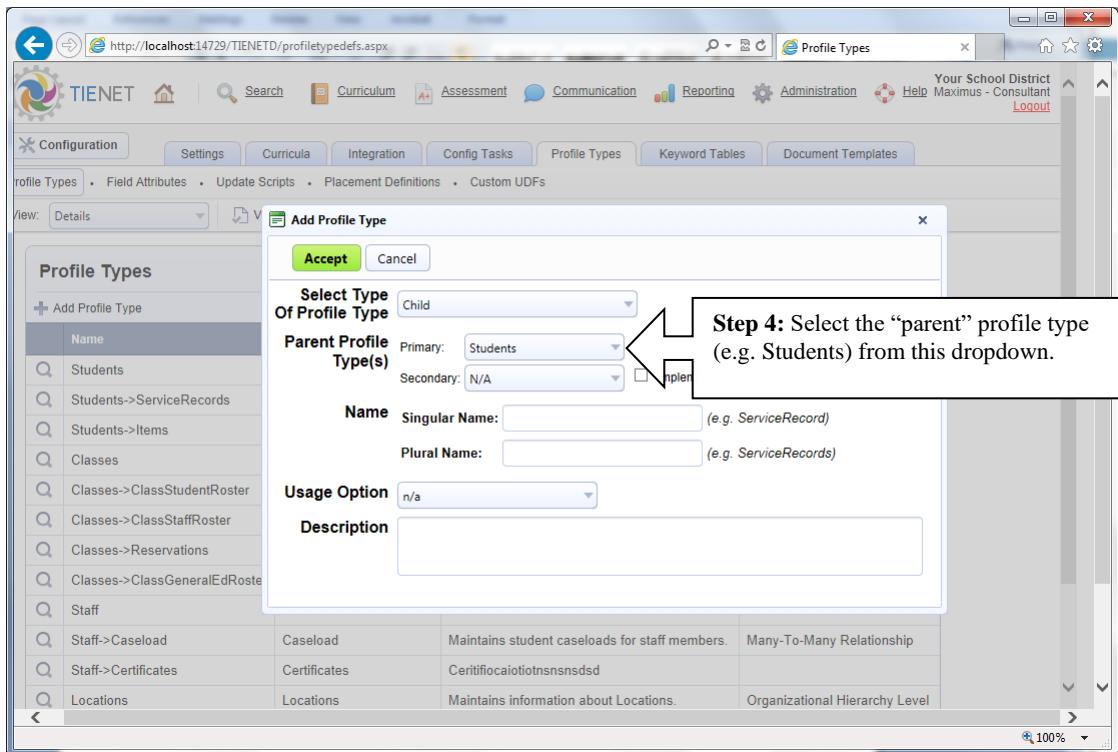
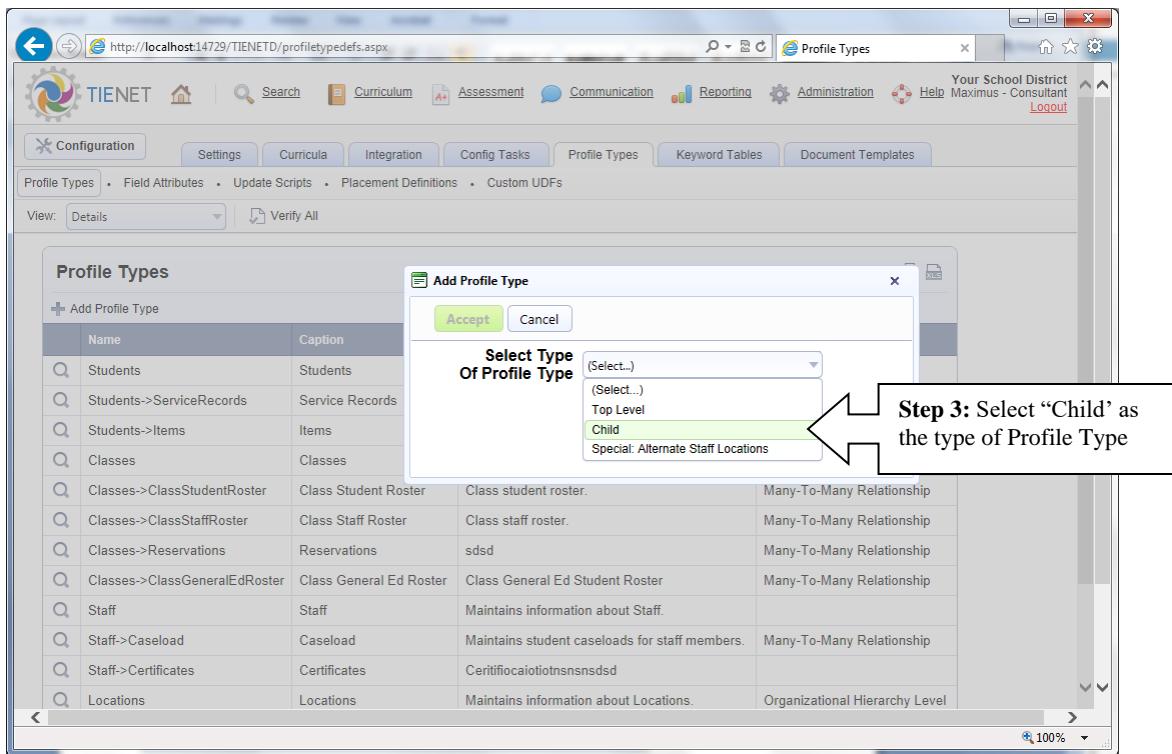
Step 1: Click Configuration in the blue sidebar, and then click Profile Types here.

Step 2: Select a configuration task if necessary, and then click Add Profile Type.

FYI: Note that existing “child” profile types of the “Students” profile type are listed using a format like “Students -> ServiceRecords”.

Name	Description	Properties
Students	Maintains information about Students.	Inactive Profiles Allowed
Students->ServiceRecords	Maintains history of service related events	
Students->Items	Hello There	
Classes	Maintains information about Classes.	
Staff	Maintains information about Staff.	Many-To-Many Relationship
Staff->Caseload	Maintains student caseloads for staff members.	Many-To-Many Relationship
Staff->Certificates	Ceritificatio	Many-To-Many Relationship
Locations	Maintains information about Locations.	Organizational Hierarchy Level

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)



PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

The screenshot shows the TIENET Profile Types configuration interface. A modal dialog titled "Add Profile Type" is open, showing the configuration for a new profile type named "Suspension".

FYI: Child profile types with a secondary parent are outside the scope of this section. Leave this blank.

Step 5: Enter the singular and plural name of the kind of "object" to be tracking with this child profile type.

Step 6: If you wish to maintain a history of events sorted by date with the most recent dates at the top, choose one of these options. In this example, we want to track suspensions by date (not date and time).

Step 7: Enter a description and then click the "Accept" button.

Profile Types List:

- Students
- Students->ServiceRecords
- Students->Items
- Classes
- Classes->ClassStudentRoster
- Classes->ClassStaffRoster
- Classes->Reservations
- Classes->ClassGeneralEdRoster
- Staff
- Staff->Caseload
- Staff->Certificates
- Locations

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

Step 8: The child profile type is created and initially has only a few fields. Click the edit (pencil) icon next to the “event date” field to give it a more meaningful name if desired.

	Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description	Linked To Sections
	Student	Student Profile Reference (prevent empty values)	Critical, Parent Reference, Quick Search	-	-	Identifies the student.	
	SuspensionDate	Date	Quick Search (Sort By)	-	-	The date for this record.	
	Profile_Created_On	Date Time	Critical, Automatic	-	-	Date that profile was created.	
	Profile_Modified_On	Date Time	Critical, Automatic	-	-	Date that profile was last modified.	

Step 9: Now use these add field links to add the rest of the necessary fields as described in the previous chapter.

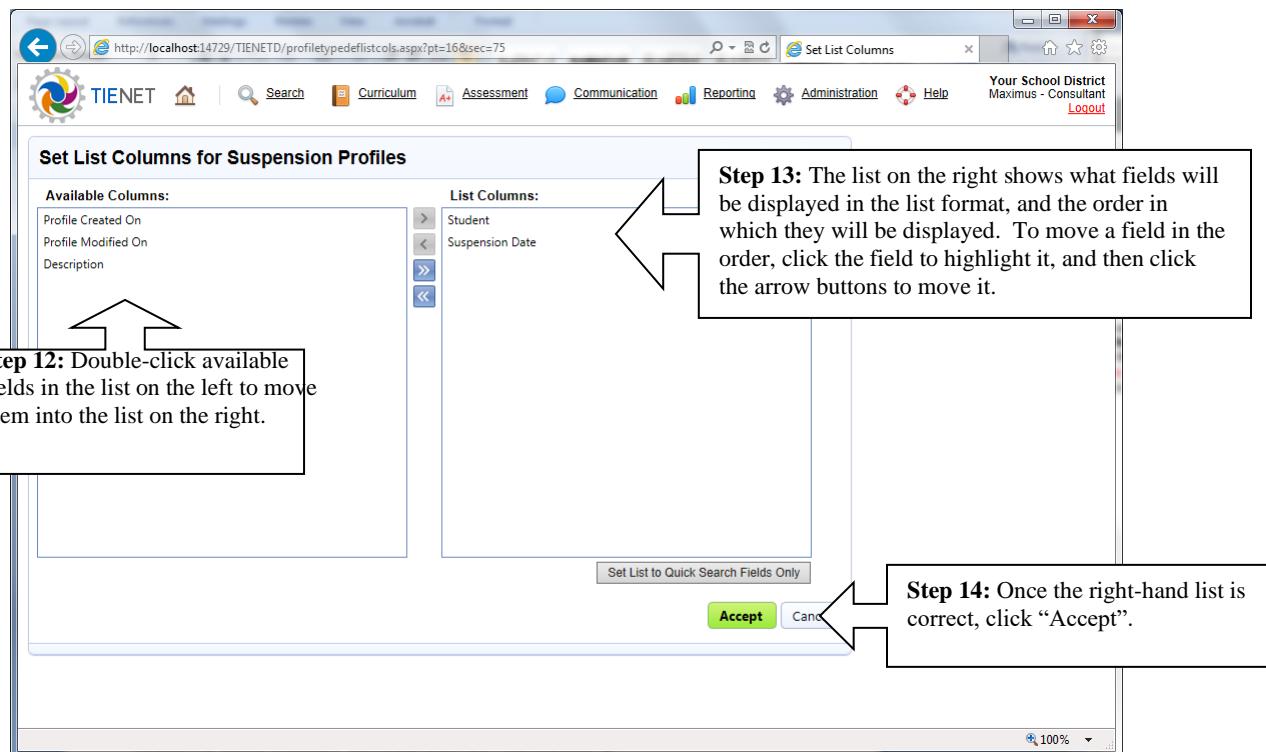
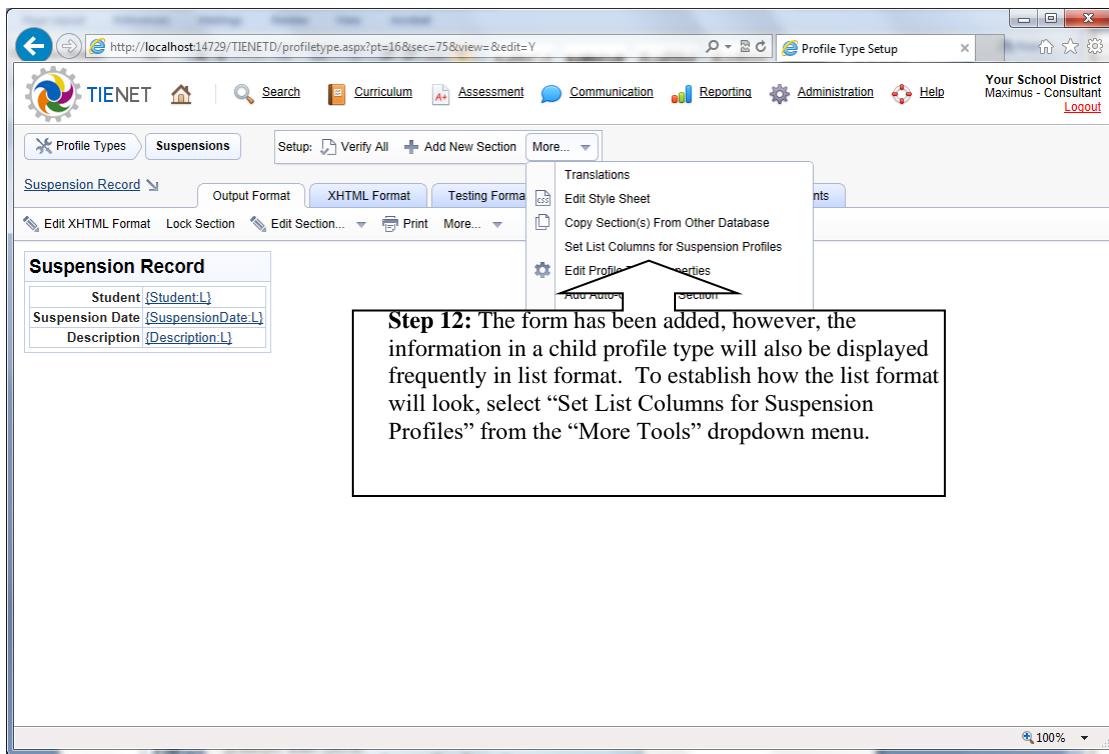
	Field Name	Data Type (Length)	Properties	Attributes	Custom Field	Description	Linked To Sections
	Student	Student Profile Reference (prevent empty values)	Critical, Parent Reference, Quick Search	-	-	Identifies the student.	
	SuspensionDate	Date	Quick Search (Sort By)	-	-	The date for this record.	
	Profile_Created_On	Date Time	Critical, Automatic	-	-	Date that profile was created.	
	Profile_Modified_On	Date Time	Critical, Automatic	-	-	Date that profile was last modified.	

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

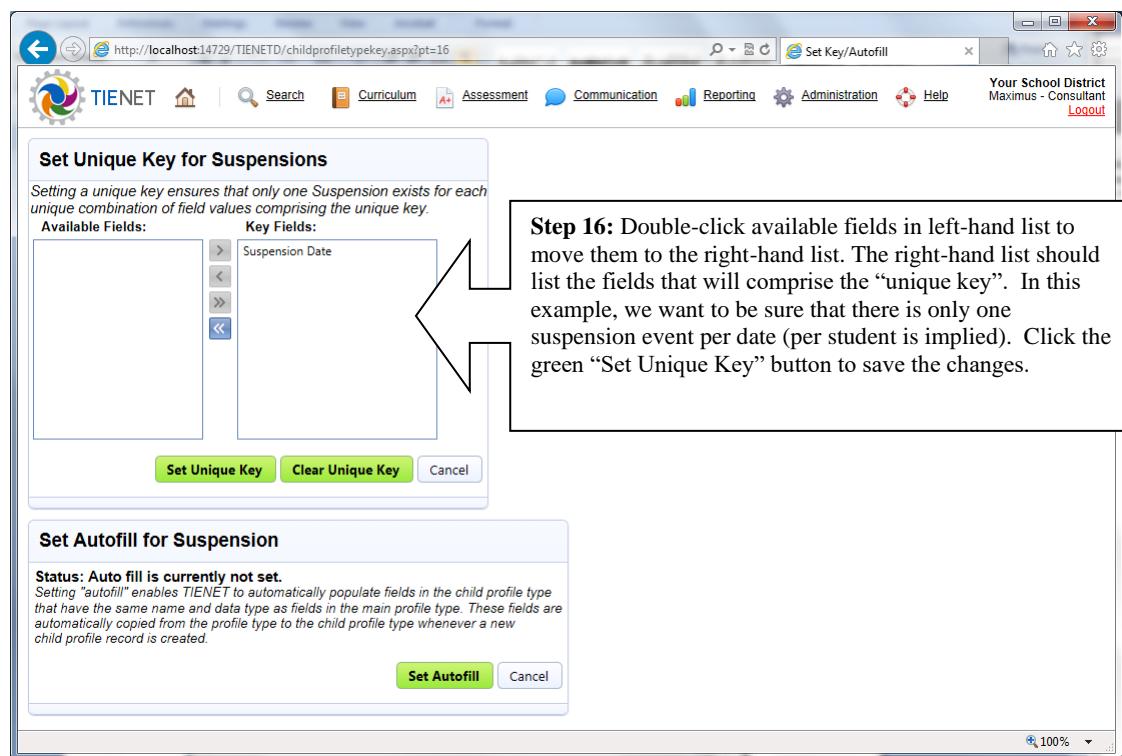
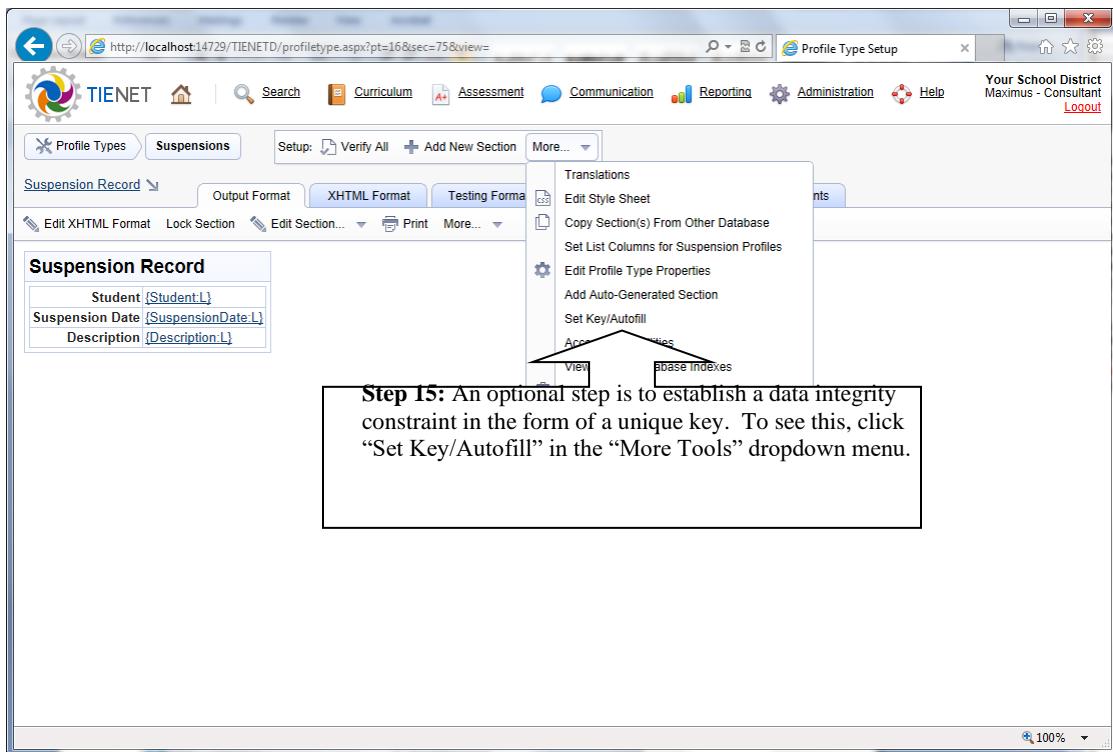
The screenshot shows a web-based application interface for 'Profile Type Setup'. At the top, there's a navigation bar with links like 'Search', 'Curriculum', 'Assessment', 'Communication', 'Reporting', 'Administration', and 'Help'. A user session is shown as 'Your School District Maximus - Consultant Logout'. Below the navigation, there are tabs for 'Profile Types' and 'Suspensions', with 'Suspensions' being active. A sub-menu under 'Suspensions' includes 'Setup', 'Verify All', and 'Add New Section'. A callout box with an arrow points to the 'Add New Section' link, containing the text: 'Step 10: Next, click Add New Section to add a form for the child profile type.' The main content area displays a table titled 'All Suspension Fields' with columns: Field Name, Data Type (Length), Properties, Attributes, Custom Field, Description, and Linked To Sections. The table contains four rows of data, with the last row ('Description') highlighted in green.

The screenshot shows a 'Profile Form Section Properties' dialog box for adding a new section for 'Suspensions'. The 'Section Name' is set to 'Suspension Record', 'Category' is 'Top-Level Section', and 'Purpose of section' is 'View and edit'. Under 'Content generation method', 'Presentation Options' include 'White Background', 'No Page Break When Multiple Forms Printed', 'Show Profile References With IDs', 'Mobile Device Accessible', 'Mobile Device Only', and 'Editable From Mobile Device'. The 'Insert section where?' dropdown is set to 'Between 'beginning' and end'. At the bottom, there are 'Accept' and 'Cancel' buttons. A callout box with an arrow points to the 'Accept' button, containing the text: 'Step 11: Define the new form section as described in the previous chapter, however, you will want to check/include every field in the section with the exceptions of Profile_Created_On and Profile_Modified_On.'

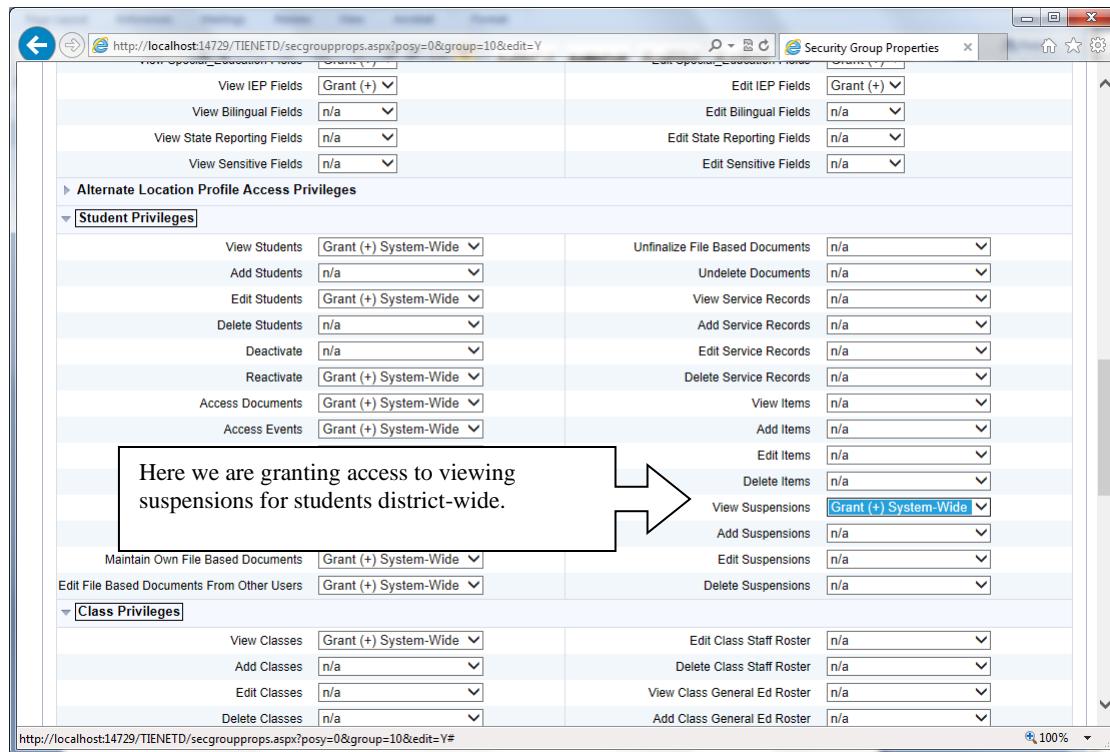
PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)



PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)



If you wish to allow any users (other than ADMIN or CONSULTANT) to access information in the child profile type, you will need to modify the security groups and grant them privileges as shown in the illustration below.



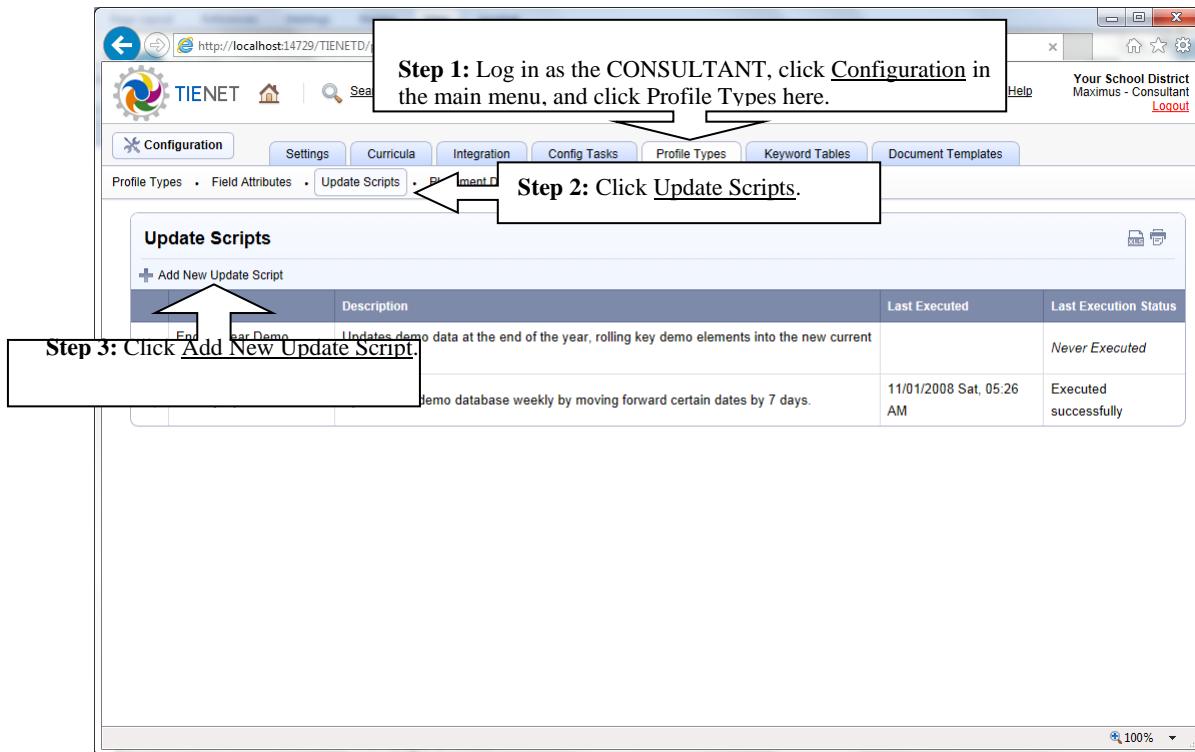
FYI - Providing Caseload or Class Access to Child Profiles of Student Profiles: Normally the "Access My Class" and "Access My Caseload" security privileges provides view access to student profiles in the caseload or class, but not by default the child profiles of students. But if you wish to extend this kind of access to child profiles of students, it is necessary to set the "Allow Access Via My Caseload" to "Allow Access Via My Classes" options in the configuration properties of the child profile type. In this case, it is not necessary to directly assign child profile view privileges in a security group that already has "Access My Class" or "Access My Caseload" security privileges.

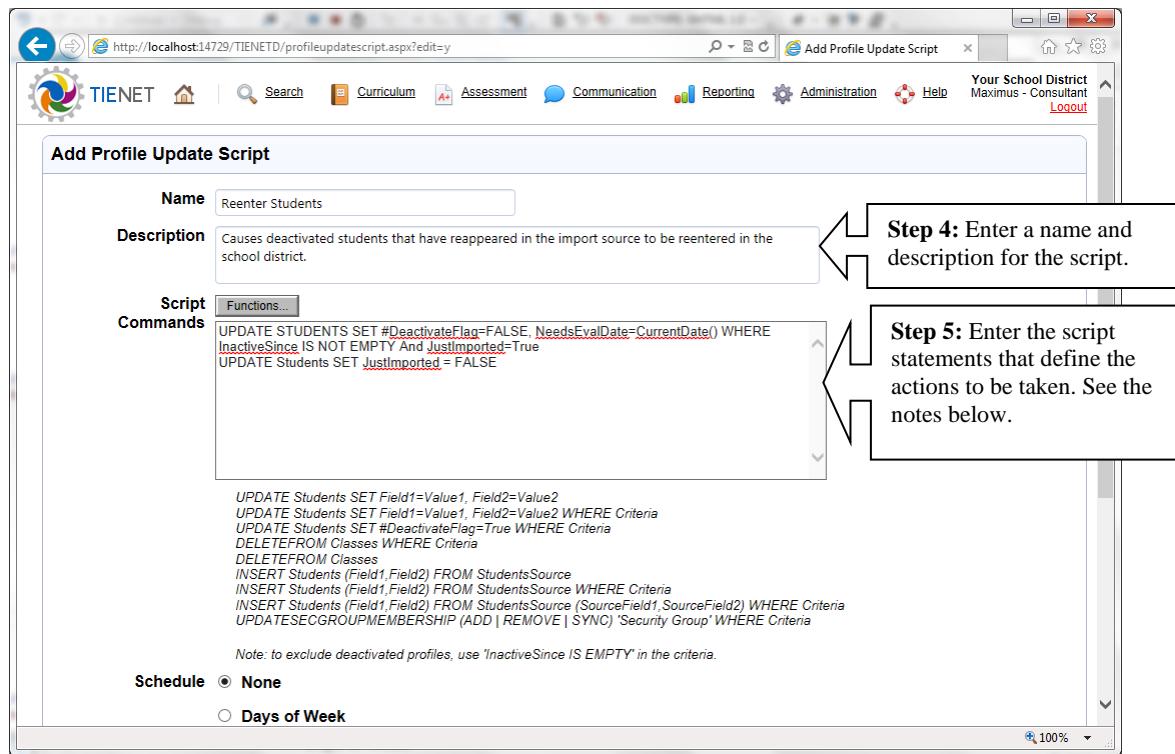
FYI – About “AllowEdit”, “AllowDelete” and “AllowAdd...” fields: You can optionally add “AllowEdit” and “AllowDelete” fields to a child profile type to control whether end users are allowed to edit or delete individual child profiles. “AllowEdit” and “AllowDelete” can be either data fields or calculated fields, although in most configurations where these are used, they will be calculated fields. Note that if these fields specify that a particular child profile cannot be edited and/or deleted, the user will be prevented from editing/deleting the child profile even if the user is otherwise authorized to do so. Similarly adding child profiles can be controlled by a field in the parent profile type named “AllowAdd<childprofiletypename>” where <childprofiletypename> is a placeholder for the plural form of the child profile type name. This field in the parent profile then controls whether it is valid to add child profiles of that type based on characteristics of the parent profile

Configuring Database Update Scripts

Database update scripts can be executed manually or can be scheduled to run nightly, on a certain day of the month, or before or after the end-of-year close procedure. Database update scripts can modify one or more fields of profiles selected by criteria, and can also be used to activate, deactivate or even delete profiles.

To set up a database update procedure, follow the steps below:





FYI – About Script Statements: A script can have one or more statements of three general types given below:

UPDATE Students SET Field1=Value1, Field2=Value2

UPDATE Students SET Field1=Value1, Field2=Value2 WHERE *Criteria*

UPDATE Students SET #DeactivateFlag=True WHERE *Criteria*

DELETEFROM Classes WHERE *Criteria*

DELETEFROM Classes

INSERT Students (Field1,Field2) FROM StudentsSource

INSERT Students (Field1,Field2) FROM StudentsSource WHERE *Criteria*

INSERT Students (Field1,Field2) FROM StudentsSource (SourceField1,SourceField2) WHERE *Criteria*

INSERT Students (Field1,Field2) FROM StudentsSource DISTINCT (SourceField1,SourceField2)
WHERE *Criteria*

UPDATESECGRPMEMBERSHIP (ADD | REMOVE | SYNC) 'Security Group' WHERE *Criteria*

The UPDATE command modifies profiles, the DELETEFROM command deletes profiles and the INSERT command inserts profiles. In all cases, the “WHERE clause” can be omitted in which case all profiles are affected. A single UPDATE command can set one or more field values, and the replacement values can be calculated. UPDATE also supports setting the value of a “pseudo-field” named **#Deactivated** (with a pound sign in front) to either true or false, and this has the effect of deactivating or reactivating the selected profiles.

Profile update scripts can also be used on child profile types including many-to-many associations. The example below writes all students that meet the specified criteria into the caseload of the staff with ID= *staffid*:

```
INSERT CASELOAD (Student, Staff) FROM Students (THIS, staffid) WHERE Criteria and  
not(EXISTS(Caseload, staff = staffid))
```

When inserting data into a profile table (parent or child), the selected fields can be further limited by using the DISTINCT keyword. This selects only DISTINCT combinations of source values to insert into the destination.

[New in 21.11.1.0] When inserting target profiles from source profiles in an update script, a new “CAPTURE” syntax can now be used to record in each source profile the target profile that was inserted from it. The “Capture” syntax, shown below, requires the source profile type to have a profile reference field to the target profile type, and the name of this profile reference field must be specified immediate after “CAPTURE” as shown below. The pattern below includes “WHERE *profile_reference_field IS EMPTY*” to ensure that if the update script is executed multiple times, that only one target profile is ever inserted from each source profile.

```
INSERT target_profile_type_name (target_columns) FROM source_profiletype_name  
CAPTURE profile_reference_field (source_columns) WHERE profile_reference_field IS EMPTY AND other_conditions
```

Once the script has been created, if you go back into its properties, you can set some advanced timeout and notification properties as shown below:

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

<http://localhost:14729/TIENETD/profileupdatescript.aspx?edit=y>

```

UPDATE Students SET Field1=Value1, Field2=Value2
UPDATE Students SET Field1=Value1, Field2=Value2 WHERE Criteria
UPDATE Students SET #DeactivateFlag=True WHERE Criteria
DELETEFROM Classes WHERE Criteria
DELETEFROM Classes
INSERT Students (Field1,Field2) FROM StudentsSource
INSERT Students (Field1,Field2) FROM StudentsSource WHERE Criteria
INSERT Students (Field1,Field2) FROM StudentsSource (SourceField1,SourceField2) WHERE Criteria
UPDATESECGROUPMEMBERSHIP (ADD | REMOVE | SYNC) 'Security Group' WHERE Criteria

```

Note: to exclude deactivated profiles, use 'InactiveSince IS EMPTY' in the criteria.

Schedule None

- Days of Week
 Sunday Monday Tuesday Wednesday Thursday Friday Saturday
- Day(s) of Month Retry nightly on failure
 1 2 3 4 5 6 7
 8 9 10 11 12 13 14
 15 16 17 18 19 20 21
 22 23 24 25 26 27 28
 29* 30* 31*
- *If any date you check off is after the last day in a month, the script will execute on the last day of that month. To always execute the script on the last day of the month, check day 31.
- End of Year Rollover
 Execute before normal EOY processing

Step 6: Schedule the execution of the script, or select the "None" option if you wish to manually execute the script.

Step 7: Click "Accept".

<http://localhost:14729/TIENETD/profileupdatescript.aspx?s=5>

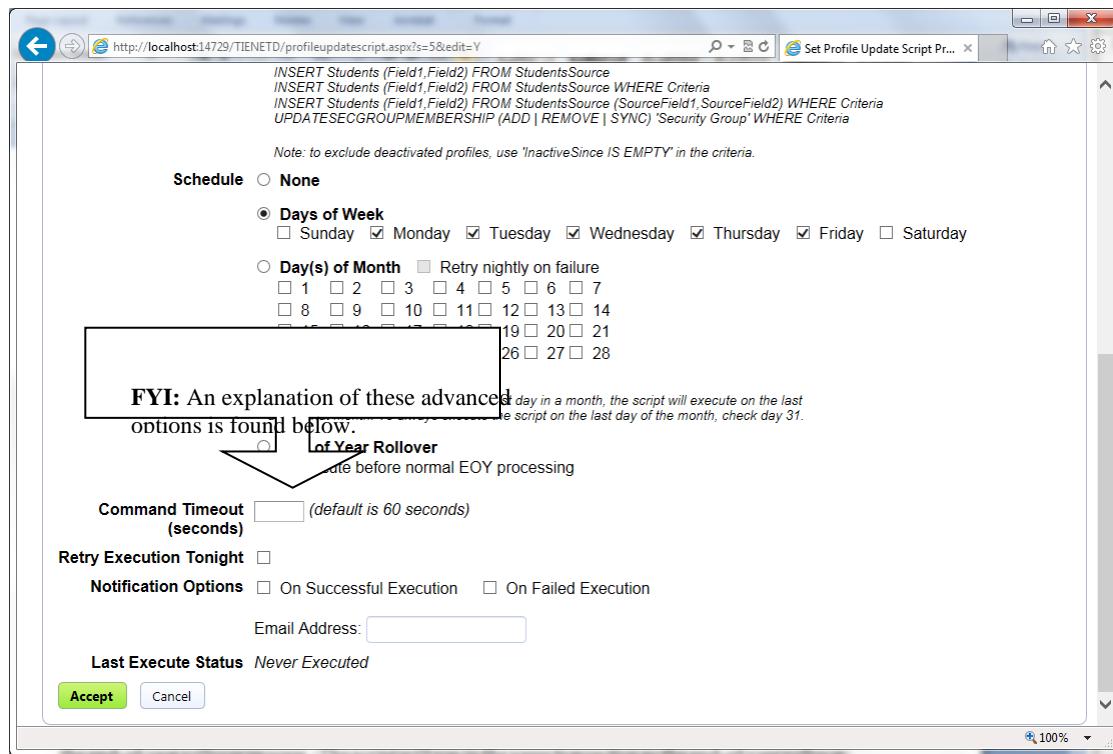
TIENET | Search | Curriculum | Assessment | Communication | Reporting | Administration | Help | Your School District | Maximus - Consultant | Logout

Profile Update Scripts

Step 8: Once the script is created, click Edit to see additional advanced options only available when editing an existing script.

Profile Update Script: Reenter Students

Name	Reenter Students
Description	Causes deactivated students that have reappeared in the import source to be reentered in the school district.
Script	UPDATE STUDENTS SET #DeactivateFlag=False, NeedsEvalDate=CurrentDate() WHERE InactiveSince IS NOT EMPTY
Statements	And JustImported=True UPDATE Students SET JustImported = False
Schedule	On the following days of the week: Monday, Tuesday, Wednesday, Thursday, Friday
Last Execute	Never Executed
Status	FYI: Note that there is a button that allows you to execute the script manually and immediately.
<input type="button" value="Execute Script"/>	



FYI – About Scheduling Scripts for End of Year Rollover: You can schedule a script to run during the end-of-year rollover process. The script will run in the same transaction as the end-of-year rollover so that if there is an unexpected failure, everything will roll back to the original state including the effects of any and all such scripts. There is a sub-option that allows one to determine whether the script runs either before or after the end-of-year rollover. If the script runs before, the current school year will still be the old year. If the script runs after, the current school year will already have advanced to the next year.

FYI – About Advanced Script Options:

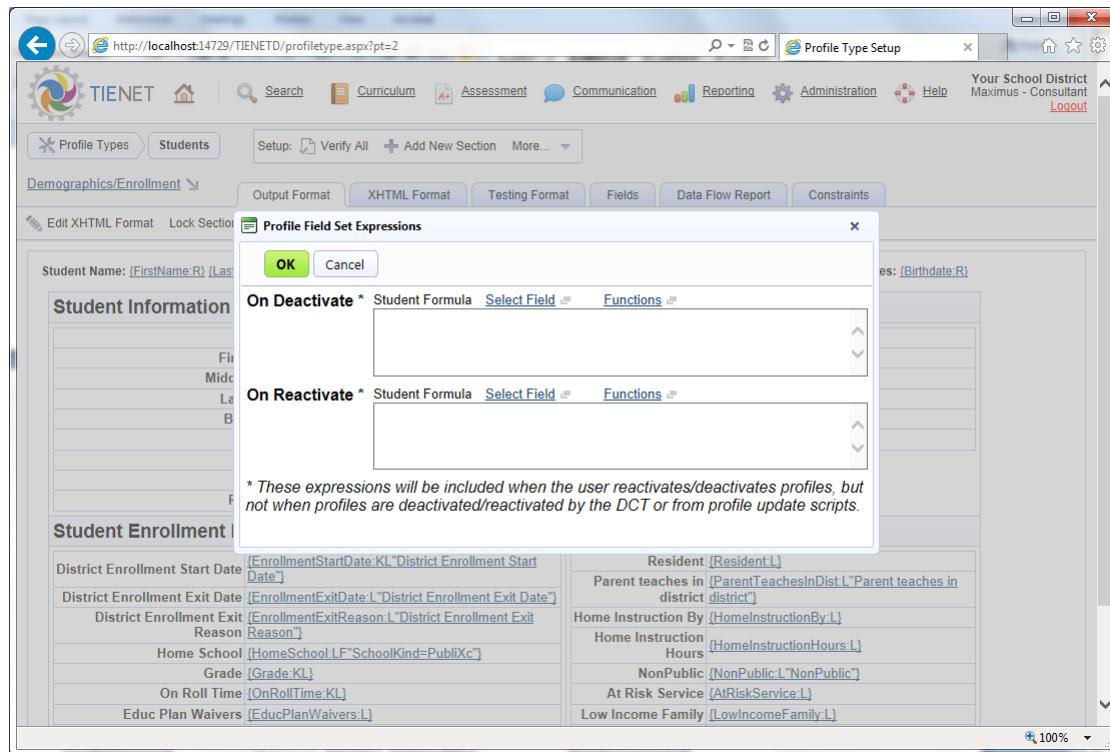
Command Timeout (seconds): Long-running scripts may require a longer command timeout than the default of 60 seconds. It is best to leave this entry blank unless timeout errors actually occur.

Retry Execution Tonight? If a script scheduled to run during the day fails (e.g. times out), the script will automatically be run again that night if this option is checked.

Notification Options: You can have an email sent to a particular email if a scheduled script execution successfully runs or fails (or both).

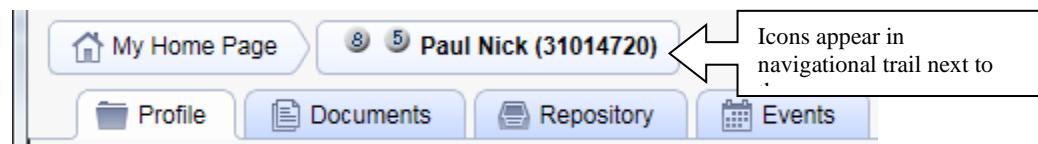
Configuring Deactivation/Reactivation of Profiles

It is a common scenario that it is desirable to set the values of some additional fields with a profile is deactivated or reactivated. To configure this, go to the relevant profile type configuration and select Set Profile Field Set Expressions from the “More Tools” dropdown menu. Then you can set the values of one or more fields using the configuration screen below. At this time, the expressions will only be used when an end-user deactivates or reactivates a profile, and not when the Data Connectivity Tool or a profile script does so. Support for the later could be added in the future if there is demand.



Configuring Profile Tags

Profile fields can be configured as special “profile tags” that appear to the user as status indicators or alerts for profiles of that type. Profile tags show in the user interface as icons in the navigational trail in the various tabs for the profile (e.g. Profile, Documents, Events) as shown below.



Only fields of the following data types can be configured as profile tags: logical, keyword, character or date. The field may be a calculated field; however, you will find that there are limitations on the complexity of the formula (for performance reasons). If a complex calculation is

needed, create the profile tag as a data field and then update its value nightly with a profile update script.

Setting a logical, character or date field as a profile tag: In the properties of the field, set the “Is Profile Tag?” property to “Yes”. Then specify a 16x16 pixel icon in one of three ways: 1) upload a 16x16 graphic (jpeg, gif or png), 2) paste in a 16x16 graphic, or 3) select from the built-in icon library. For logical fields, the icon will appear to the user if the field value is true. For the other field types, the icon will appear if the field has a non-empty value, and the actual field value will appear in the tool tip of the icon.

Setting a keyword field as a profile tag: In the associated keyword table, add a column with a data type of image and then establish 16x16 icons for each row. Then, in the properties of the keyword field, set the “Is Profile Tag?” property to “Yes”. You do not set an icon in the field properties, because the icons should be in the keyword table.

[New in 20.6.3.0] A new “Alert Type” attribute is now available for logical fields with the ‘profile tag’ attribute set and an associated icon. The ‘Alert Type’ property provides the icon with a behavior similar to Special Programs integrated alerts in PowerSchool SIS and Unified Classroom. When the end-user clicks the icon in the Special Programs user interface, a popup appears with a hyperlink to the document or student list of documents.

Edit Student Data Field Properties

Unique Field Name: GiftedTalented

Caption: Gifted Talented

Prior Field Name(s):

(used by CMT to rename from prior field name or comma-separated field names)

Data Type: Logical Value

Allow Empty Values:

Default Value: False

Quick Search Index?: No

Multiple-Value Field?: No

Is Profile Tag?: Yes

Icon Image:

To set a logical field as a profile tag, set the “Is Profile Tag?” property to “Yes”. Then specify a icon via uploading an icon image, pasting an icon image or selecting an icon from a library.

Alert Type: IS_GT

Field Properties:

- Internal Value Only
- Retired

Configuring General Ed Students for On-Demand Importing

The “Students” profile type generally only contains students where services are being tracked through the PowerSchool Special Programs system, and licensing costs are generally based on this number. In this case, it is helpful if an authorized user can instantly import any desired students from the entire

population of students. A popular approach to accomplishing this is to create a special profile type named “General Ed Students” that only contains the critical demographic fields from the Student Information System. As such, the “General Ed Students” profile type will typically contain a much smaller number of fields than the “Students” profile type. This profile type is then synchronized with data from the Student Information System nightly or according to some other schedule. Then an authorized user can instantly import a student from the General Ed Students profile type to the working Students profile type. The high-level steps to accomplish this are as follows:

1. Create the “General Ed Students” profile type, which is a special profile type that can be directly selected from the dropdown menu on the configuration screen for adding a new profile type. Add the critical fields to “General Ed Students” that you want to synchronize with the SIS. These should be the fields needed to get a student started when the student is referred or otherwise enters a PowerSchool Special Programs process.
2. Use the Data Connectivity Tool to set up a mapping and a regularly scheduled import from the Student Information System. The import should be set up to both add and update student profiles.
3. You will also want to set up a separate Data Connectivity Tool scheduled import directly into the “Students” profile type to keep it up to date. If student profiles will only be added to the “Students” profile type by user on-demand import from General Ed Students, it follows that the scheduled import for Students only needs to update student profiles, not add student profiles, and this is an option that can be set in the import layout.
4. You will need to authorize the appropriate users to import student profiles from the “General Ed Students” profile type to the “Students” profile type. In addition to view access to student profiles (“Students” profile type), the user also needs:
 - a. Appropriate view access to “General Ed Students” allowing the user to see those students the user should be allowed to import.
 - b. The “On Demand Import” special access privilege.
5. With the above configured, an authorized user can search “General Ed Students” via quick search, find a particular student, click Utilities and then “On Demand Import”.

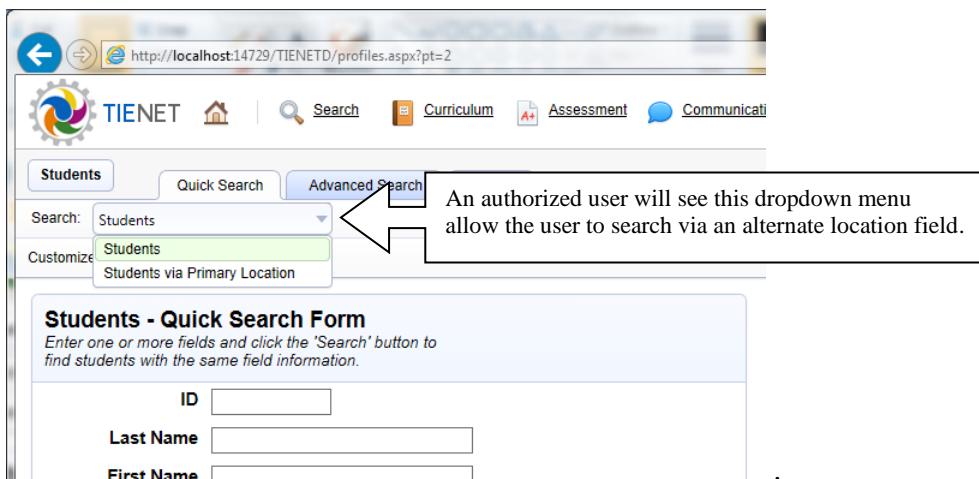
Class General Ed Roster: You can also take this a step further by creating a special “Class General Ed Roster” profile type, which like “General Ed Students”, is a special profile type that can be directly selected from the dropdown menu on the configuration screen for adding a new profile type. The problem this solves is that when a student is imported from “General Ed Students” to “Students”, those students will not be enrolled in classes right away. But by creating “Class General Ed Roster” and keeping it populated, PowerSchool Special Programs will automatically import the classes the student is enrolled in as well. No additional configuration is needed.

Additionally, when “Class General Ed Roster” is created, class room teachers can switch to a view of all the general ed students in the class (again, no additional configuration needed). And if a class room teacher is granted the “On Demand Import” special access privilege, the class room teacher can then import the student directly from the class view and work with the student right away.

Configuring Profile Access via Alternate Location Fields

If location-wide, system-wide or some other level of student access is granted to a group of users, then the list of students those users can access will be driven by the student profile location field that is participating in the organizational hierarchy. Typically this field is named “Home School”, “School”, “Building” or “Primary Location”. There may also be situations where it is desirable to allow users to access students based on one or more additional location fields in the student profile (e.g. “Alternate Location”, “Secondary Location”, “Summer Location”). The capabilities described in this section for such authorization, are subject to certain limitations. The next section following this section describes yet another model, supported only for student access, which allows a many-to-many relationship from students to locations all as part of the standard organizational hierarchy.

If “Home School” is the main location field participating in the organizational hierarchy, but a user is also authorized to access students via the “Summer Location” field (using the capabilities described here), then the student quick search screen will have the dropdown menu shown in the screen below. By default, the search is the normal one via the organizational hierarchy, but the user can use the dropdown menu to search for students via other authorized location fields. This can allow the user to access students who would be inaccessible via the normal organizational hierarchy.



If a user is authorized to access students via other location field(s), the behavior of the search (and user interface) will vary depending on what scope of access the user has:

- If the user only has location-wide access to students, the user can simply change the drop down shown above to search for students via other authorized location fields.

- If the user has system-wide access, the dropdown does not appear at all since it is not relevant. Such a user can search all students anyway.
- The situation is somewhat more complicated when a user has access that is broader than location-wide (e.g. unit-wide, area-wide) but narrower than system-wide. When this user does a student search the default way via the organizational hierarchy, the entire scope of access the user has within the organizational hierarchy is searched (e.g. unit-wide, area-wide). However, when this user searches via another authorized location field, since that field is not participating in the organizational hierarchy, the user can only do a location-wide search although for any location within user's scope of access (e.g. unit-wide, area-wide). Therefore the user will need to select a particular location to search (as shown below) with the default being the user's current location.

The screenshot shows the TIENET web application interface. At the top, there is a navigation bar with links for Home, Search, Curriculum, Assessment, and Communication. Below the navigation bar, there is a header with tabs for Students, Quick Search, Advanced Search, and Utilities. The Quick Search tab is active. In the search area, there is a dropdown labeled 'Search:' with options 'Students via Primary Location' and 'Primary Location'. The 'Primary Location' dropdown is open, showing a list with 'Sample Elementary (SAMPLE)' selected. A callout box with an arrow points to this dropdown. The callout box contains the text: 'A user with access broader than location-wide will be able to select a particular location to be searched with the default being the user's'. Below the search area, there is a section titled 'Students - Quick' with a sub-instruction 'Enter one or more fields find students with the same...' and two search fields: 'ID' and 'Last Name'.

The new capabilities also work with list reports. A report option can be set for individual list reports to provide a dropdown similar to one shown above but within the context of the list report.

To set this up, follow these high-level steps:

1. Make sure that each relevant location reference field is marked as having a quick search index. If it is desirable to omit such field(s) from the quick search form, there is an option in the field properties to allow a field to have a quick search index but not appear on the quick search form. Note that the location fields cannot be calculated or multiple value fields.
2. Associations between security groups and these location fields are not made directly. Rather one or more field attributes in the category "Alternate Location Access Attributes" (see below) are defined and used to "tag" the location fields in whatever way makes sense for authorization. Each field attribute defined in this category generates a new security privilege that can be granted to security groups.

Profile Field Attributes

Weak Field Security Attributes: * Asterisked entries are reserved for custom attributes.

Basic_Skills	Special_Education	IEP	Bilingual
+*	+	+ *	State Reporting *

A profile field with weak attributes is accessible to users with access privileges to ANY of those weak attributes.

Strong Field Security Attributes:

Define one or more field attributes in this category in ALL of those strong attributes. whatever way makes sense for authorization and then tag the location fields appropriately.

Reporting Fields	Sensitive
+*	+ *

A profile field with reporting fields is singled out for easy selection into common reports.

Alternate Location Access Attributes:

Primary Location	+*	+*	+*
------------------	----	----	----

A profile field with alternate location access attributes are singled out for easy selection into common reports.

- Once one or more location fields are tagged with the attribute(s) defined in the previous step, you can then assign security. As shown below, you can grant basic view access which allows viewing profiles and documents via the alternate location (further subject to document template security rights). You can also grant view access and document editing access, in which case the user can also edit documents subject to template security rights. Finally there is an additional option to grant editing access to both profiles and documents.

Alternate Location Profile Access Privileges

Primary Location

- Deny (-)
- n/a
- Grant View Access (+)**
- Grant View Profiles / Edit Documents (+)
- Grant View/Edit Profiles/Documents (+)

Many-to-Many Students/Locations Security Model

A limitation of the alternate location field security mechanism is that, to take advantage of it, the end user must select which alternate location field the user is searching through and then perform the search. An alternative is to set up a many-to-many relationship between students and locations and configure that as the basis for security access to students. Once that is done, a user does not have to select a particular location field to search with, but simply does a quick search. If a user has access to students at one location, then a quick search produces all students associated with that location regardless of what other locations such students may be associated with.

Setting up this model is relatively simple, although there may be data migration and reporting implications to retiring any location field(s) that may become unnecessary as a result. It is best to experiment with this model in a test database set up for that purpose.

To set up the model, follow these steps:

- 1) Create the students-locations many-to-many relationship by adding a child profile type similar to that shown below.

- 2) Once the new child profile type is created, select More -> Attach to Organizational Hierarchy.
- 3) Choose “Location” from the dropdown menu and click Accept.

In the last step above, the system copies all student profile organizational location field values into the many-to-many relationship, and then switches organizational security to look at the new many-to-many relationship. Because the values are copied, there is no change in access or apparent change in behavior right away. The original organizational location field in the student profile is no longer used for security at this point, but is otherwise left intact since there will at least initially be other dependencies on it (forms, reports, imports, exports, etc). If there are alternate location fields, migrating those to the many-to-many relationship can now be considered to simplify the experience for end users. An immediate concern will be to ensure that the new many-to-many relationship continues to get populated from an import source.

Configuring Student Data Transfer Envelopes

4. If a student is being transferred between two school districts using PowerSchool Special Programs that are hosted in different databases, a utility is available that captures the student profile, certain child profiles and documents in the form of PDFs and packages this data into a “student data transfer envelope” that can be sent from the source to the receiving school district where it can be uploaded to the receiving PowerSchool Special Programs database. This will enable the receiving school district to access a historical record of documents in a timelier manner. The student transfer envelope can also carry the data that underlies documents such that a “live document” can be launched in the receiving database.

For this utility to work as expected, the student profile must be configured to support it as follows:

Configuring Student Profile for Student Transfer Envelopes: Enable the “Include Fields in Transfer Envelope” property in the section properties of one or more student profile form sections that should be included in student transfer envelopes. The student data transfer envelope will only include student profile data fields from those marked sections. Note that the student transfer envelope will transfer fields of all data types except for profile reference fields (e.g., staff references) since those will not be valid in the destination database.

To enable child profiles of student profiles to be transferred in the same way, enable the “Eligible for Transfer Envelopes?” property of the child profile type noting that many-to-many child profile types (e.g., Caseload) and do not support the property, and child profiles with a secondary parent (e.g., Service Records under students and staff) support it only in certain cases to be described shortly. Form sections associated with the child profile do not have a “Include Fields in Transfer Envelope” since it is assumed that all data fields in the child profile are included.

Conditional Support for Transferring Service Capture Records: Service capture records have a primary parent profile type of students and a secondary parent profile type of staff. Child profiles that have a secondary parent do not generally support the “Eligible for Transfer Envelopes?” property because it is unlikely that the secondary parent profile type will exist in the receiving district. An exception to this is conditional support for service capture records. The impediment to transferring service capture records is that the receiving district tenant will not normally have the same staff profiles as the sending district. However, if the model is intended for use with a state/regional controller, it is possible that the same staff profiles may exist in more than one tenant in which case it is possible to transfer service records while preserving their staff associations. If this is a valid scenario for the model, the “Eligible for Transfer Envelopes?” property can be set for service capture enabled records with the caveat that this option will only be available to end users in district tenants that are part of a state and regional controller group. When service records are transferred this way, the system will successfully transfer service records whenever the relevant staff profile exists in both source and target district tenant (staff profiles are matched by ID, FirstName and LastName in sending and receiving district tenants). If there are additional staff fields in the service capture record profile, those values will also be transferred if a staff profile match can be made. Normally, service records for which the main staff association is not

found in the target district will not be transferred; however, it is possible to enable even unmatched service records to transfer for historical purposes by implementing the following:

1. Add a character field named HistoricalStaffNameID (length=100) to the service capture child profile.
2. Make sure each district tenant has a nominal staff profile with ID="RecordsTransfer". The name for the staff profile can be something like "Historical Records Owner (DO NOT DELETE!)", but only the ID matters to the functioning of the student transfer envelope.

If the above is in place, upon transferring service capture records, if the main staff association cannot be found in the receiving district tenant, then the service capture records will be transferred anyway under staff profile with ID="RecordsTransfer". Additionally, the HistoricalStaffNameID field will be populated with the staff ID and name from the sending district tenant for historical purposes.

Configuring Document Templates for Student Transfer Envelopes: No configuration is required to allow any documents to be included in a student transfer envelope and received by the receiving district as a PDF. There are several optional document template properties to modify or enhance the default behavior:

- **Exclude From Transfer Envelopes:** Prevents the document template from being included in any new student transfer envelopes. Note that the ADMIN can override the value of this property as a behavior override.
- **Include Live Documents in Transfer Envelopes:** If this is enabled, the receiving school district will still receive the PDF, but assuming source and receiving district are on the same model, the receiving district will have the option to recreate the original live document in draft form with all sections incomplete and any profile reference fields having an empty value as described above. Note that the ADMIN can override the value of this property as a behavior override.
 - **Support Transferring Final Live Documents:** This option can be used to support cases where it is necessary to launch the live document as a final document. Using this option may require additional supporting configuration as described below. Note that the ADMIN cannot override the value of this property since it typically requires supporting configuration.

Enabling Student Transfer Envelopes to Transfer Final Live Documents (new in 22.6.4.0):

Normally, documents transferred in a student transfer envelope as live documents are recreated in draft mode in the receiving database because it is typically not possible to repopulate all fields (particularly staff fields) leaving any number of sections incomplete (i.e., required fields may be empty). However, scenarios such as allowing the receiving school district to complete a progress report for an IEP from the sending district require a solution for this limitation. Note that progress reports are typically configured to be editable when the document is final. The "Support Transferring Final Live Documents" document template property can be enabled to allow final documents to be recreated in the receiving database such

that the document status is set to final even while ignoring the fact that required staff and other profile reference fields may be empty. However, the progress report scenario will typically depend on certain staff fields being populated which will be empty after the transfer. To resolve this, one can: 1) add a logical field named “FromStudentTransferEnvelope”, 3) add a document action with a trigger type of “Modify Data For New Transfer Envelope Live Document” (new in 22.6.4.0) that sets the “FromStudentTransferPackage” field to true, 4) have additional content in the progress report (assumed to be editable in final) that allows a user to edit the key staff fields if “FromStudentTransferEnvelope” is true.

Configuring End-Of-Year Rollover

At the end of the year, following end-of-year reporting procedures, it is necessary for the system administrator to complete the end-of-year rollover process.

This end-of-year process first performs the following functions inside of an “all or nothing” transaction:

Executes any profile update scripts configured to run immediately before the end-of-year process. Profile update scripts can perform mass insertions, updates, or deletions of profile data in the PowerSchool Special Programs database. See the previous section for details on profile update scripts.

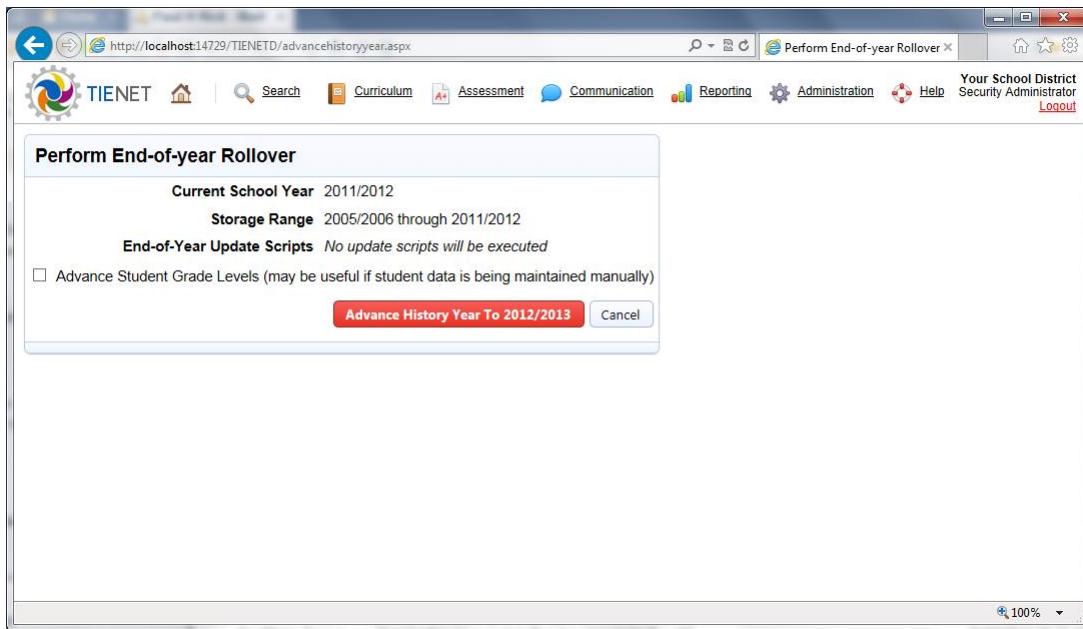
Advances the current school year of the PowerSchool Special Programs database (affecting any reports that refer to the current school year), extends calendar storage, and resets the marking period to the first one.

Executes any profile update scripts configured to run immediately after the end-of-year process. Profile update scripts can perform mass insertions, updates, or deletions of profile data in the PowerSchool Special Programs database.

Additionally, there are some additional cleanup options that are selectable by the system administration.

Advance student grade levels. Note that this option is used only if there is no SIS integration, since it is more accurate to import current student grade levels from the SIS.

Options to clear out class profiles or class rosters. These options will not be available and visible if class rosters are configured to be retained across multiple years.



Service Capture Configuration

PowerSchool Special Programs provides an integrated service capture by recognizing and integrating a specially configured child profile type named 'ServiceRecords' with the calendaring module. However, the child profile type needs to be configured according to certain rules. To add and configure the 'ServiceRecords' child profile type in a database, follow these steps:

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)

Add the ‘ServiceRecords’ profile type as illustrated below.

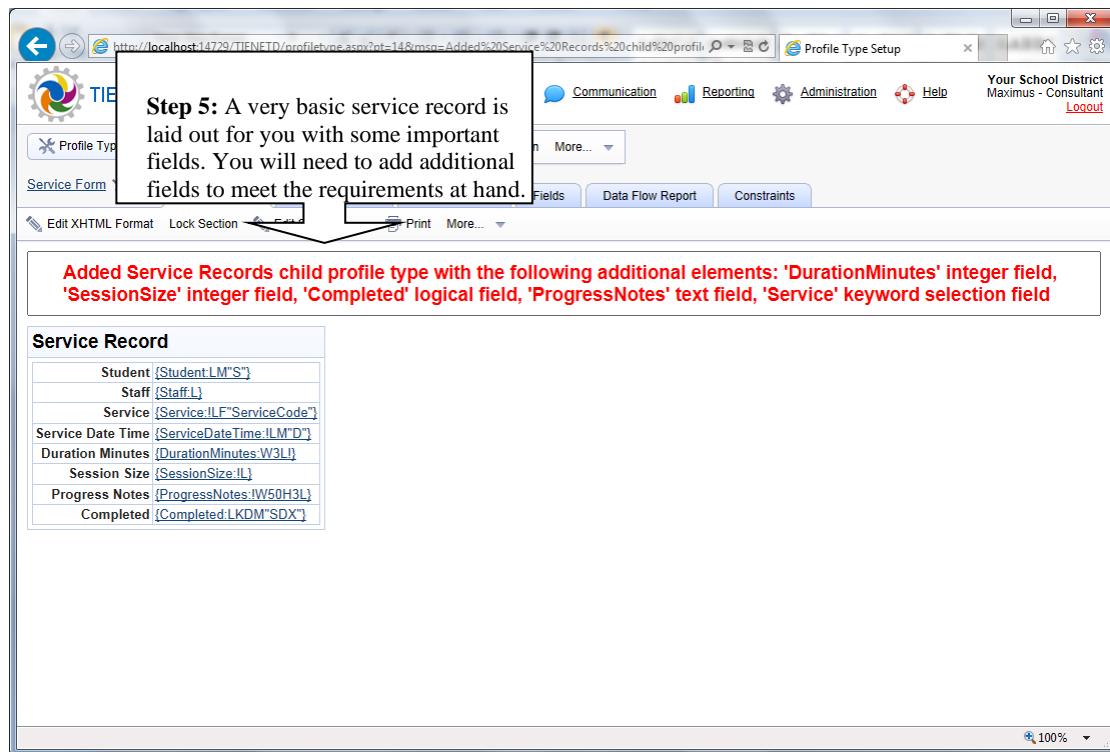
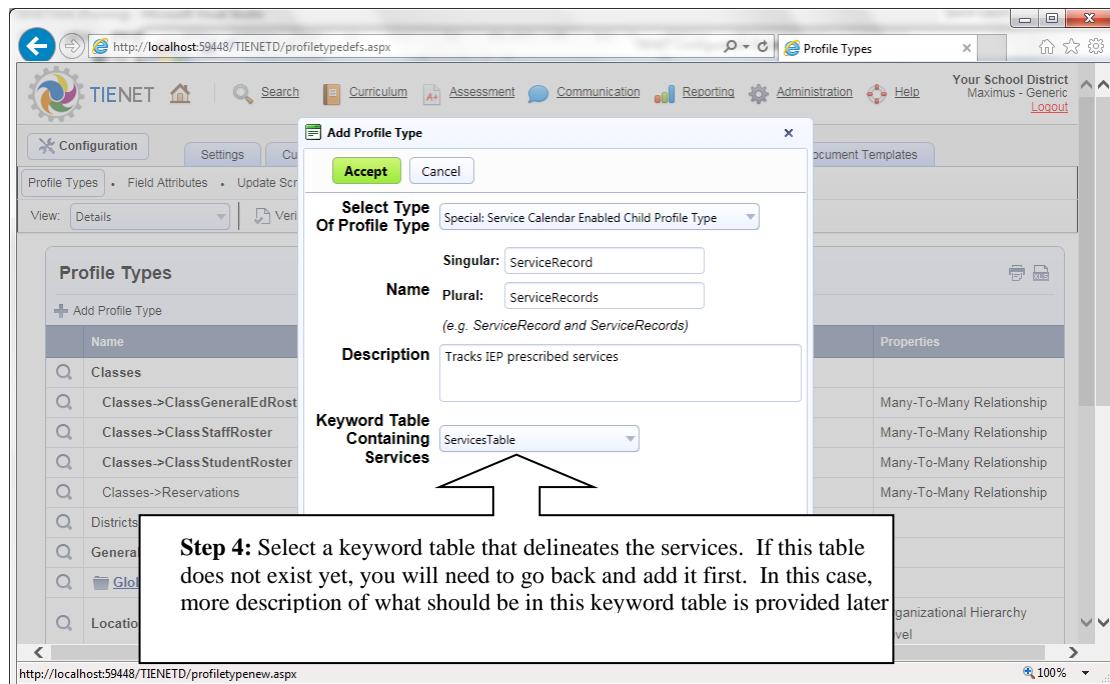
Step 1: Log in as the CONSULTANT, click Configuration in the main menu, and then click Profile Types here.

The screenshot shows the TIENET application interface. The main menu bar includes Configuration, Settings, Curricula, Integration, Config Tasks, Profile Types, Keyword Tables, and Document Templates. The Profile Types option is highlighted. The sub-menu under Profile Types lists: Profile Types, Field Attributes, Update Scripts, Placement Definitions, and Custom UDFs. The View dropdown is set to Details. A search bar is present. The main content area displays a table titled 'Profile Types' with columns: Name, Caption, Description, and Properties. The table contains several entries, including 'Classes', 'Class General Ed Roster', 'Class Staff Roster', 'Class Student Roster', 'Reservations', 'Districts', 'GeneralEdStudents', 'Globals', and 'Locations'. The 'Properties' column for 'Locations' indicates it is an 'Organizational Hierarchy Level'.

Step 3: Select “Special: Service Capture Enabled Child Profile Type”. Enter a singular and plural name for the child profile type to contain the service data, and then a description

The screenshot shows the 'Add Profile Type' dialog box. The 'Select Type Of Profile Type' dropdown is set to 'Special: Service Calendar Enabled Child Profile Type'. The 'Name' field is set to 'ServiceRecord' and the 'Plural' field is set to 'ServiceRecords'. The 'Description' field contains the text 'Tracks IEP prescribed services'. The dialog has 'Accept' and 'Cancel' buttons. The background shows the TIENET Profile Types page with the same list of profile types as the previous screenshot.

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)



FYI – About the default fields in ‘ServiceRecords’:

Student: References the student receiving the service. This field is required.

Staff: References the service provider giving the service. This field is required.

Service (keyword selection): Identifies the type of service, which will be a selection from the keyword table you selected when creating ‘ServiceRecords.’ This keyword table should have a character field named ‘ServiceCode’ containing the abbreviated service codes that will appear on the various calendar views. Only keywords with a service code will be selectable and will appear on the calendar; therefore, if there are types of services that should not be selectable into service records, you can leave the ‘ServiceCode’ field blank for that service. This field is required.

ServiceDateTime (date/time): Indicates the date and optional time when the student received service. This field determines where the service record appears on the calendar. This field is required.

DurationMinutes (integer): Contains the number of minutes for which the student received service. Normally this field is a user-enterable data field, but it is also possible to have this be a calculated field based on other enterable data fields. This field determines the length of the item on the calendar.

SessionSize (integer): This field is used to track session size for group service records. PowerSchool Special Programs will automatically recognize this field and populate it when service records are being added for a group. This field can be deleted if it is not wanted. It can also be renamed to GroupService and PowerSchool Special Programs will still automatically populate it.

ProgressNotes (character): A field for the provider to enter progress notes. This field can be deleted if it is not wanted.

Completed (logical): PowerSchool Special Programs will automatically recognize this field and use it to determine whether service records have been completed or not. Incomplete service records are italicized and shown in an orange/red color on the calendar to distinguish them. Note that you may alternatively wish to use profile constraints and required fields to ensure that the user can ONLY add completed records, with no possibility of incomplete service records. In this case, you can delete the ‘Completed’ field. Another alternative is to configure Completed as a calculated field based on other enterable data fields

Add any additional fields that are required to meet the requirements. Note that there are some optional additional fields that you can add that the service capture module will recognize and automatically provide additional useful behavior for, as follows:

If you will be tracking number of units per service record, you can manually add an integer field named ‘Units’. PowerSchool Special Programs will recognize this field and show its value on various calendar and instant report views next to ‘DurationMinutes’.

If you wish to have a date time range rather than a duration in minutes, add a field named “ServiceEndDateTime”. PowerSchool Special Programs will automatically recognize this field and constrain it to the same day as ServiceDateTime and will make sure the end time is not earlier than the start time. If using this approach, configure a calculated field named “DurationMinutes” that calculates the minutes.

As stated earlier, you can rename the SessionSize field to GroupSize. You can also replace the field with either name with a keyword field of the same name. The keyword table should contain “Group” and “Individual” keywords. PowerSchool Special Programs will automatically recognize this situation and automatically populate the field correctly.

You can optionally add logical fields named “AllowEdit” or “AllowDelete” to the “ServiceRecords” child profile type for the purpose of controlling whether the end user can edit or delete service records through the user interface. These fields can be either data fields or calculated fields, but typically are calculated fields in most configurations. As an example, there could be a data field called “ExportedForBilling” that is set to true by a backend process, and then “AllowEdit” and “AllowDelete” could be calculated as “NOT ExportedForBilling”.

PowerSchool Special Programs also offers legacy support for a logical field named “Finalized” which behaves similarly to “AllowDelete”. The ‘Finalized’ field can be either a data field or a calculated field. Service providers will be prevented from deleting service records for which “Finalized” is true. Unlike “AllowDelete”, “Finalized” is recognized in the service capture module but only if it is explicitly associated with the service capture form. “AllowDelete” is a generic feature available for any child profile type, and should be used instead of “Finalized” in the future.

A basic data entry form for ‘ServiceRecords’ has automatically been created for you. However, you can edit and enhance the form using the following guidelines:

- Add field directives for the “Student” and “Staff” fields, which will automatically be read only reflecting the student(s) for which the user is entering service record(s) and the staff user that is doing it. Add the rest of the field directives for the fields that should be on the form.
- Add the exclamation point (!) field directive modifier to any required fields on the form.
- Note that the field directive for ‘Service’ field has an F"ServiceCode" modifier to indicate that only services with a service code should be selectable. Without this directive, services with no service code could be entered but they would not appear on the calendar views!
- Note that the field directive for ‘Student’ has an M”S” field directive modifier so that this field is hidden when the user is entering common information for multiple students at once. Add this directive to any other fields that should also be hidden when the user is entering common information for multiple students at once.
- Note that the field directive for ‘ServiceDateTime’ has an M”D” field directive modifier so that this field is hidden when the user is entering common information for multiple dates at once. Add this directive to any other fields that should also be hidden when the user is entering common information for multiple dates at once.
- The ‘Completed’ field has an M"SDX" so that it is hidden when the user is entering common information for multiple students (S) and/or multiple dates (D) at once, or when the user is

scheduling services in the future (X). Generally, the completed field should only be checked for individual service records to help ensure that the end user has checked over each record.

- The ‘Completed’ field also has a K field directive modifier to indicate that it is ultimately required to have a complete service record. You could optionally add the D field directive to make the “Completed” field a dropdown instead of a checkbox.
- Consider using the F and O modifiers to create dynamically filtered dropdowns where one dropdown is filtered by another.
- You may wish to make certain fields required fields. The user will then be required to complete those fields even before saving a service record. There is an exception to this however. Required fields are not enforced for service records scheduled in the future. But the user will be required to complete the fields once the service record moves into the present and the user attempts to finish the service record.
- You can add profile constraints to enforce data requirements and validations for service records. Please note that profile constraints are not executed when new future scheduled service records are added, since it is assumed that such records are initially incomplete. However, if the user edits such records later, the constraints will be applied. However, there is a profile constraint property labeled “Enable Evaluation When Adding Future Scheduled Records” that overrides this default behavior.

FYI – Advanced Use of M field modifier: The M field modifier supports a number of options which either show or hide fields in different situations, as follows:

S – Suppresses field if entering common information for multiple students.

s – Suppresses field if not entering common information for multiple students.

D – Suppresses field if entering common information for multiple days.

d – Suppresses field if not entering common information for multiple days.

X – Suppresses field if scheduling services in the future.

x – Suppresses field if not scheduling services in the future.

I – Suppresses field if entering individual service records after having entered common information for multiple students.

G – Suppresses field if entering common information for group service (versus individual service). Note that prior to Version 11.0, selecting multiple students always implied group service, however, it is now possible to enable an option in the service record profile form section that supports a new scenario where

the user selects multiple students and then creates individual (not group) service records at different times on the same day(s).

g – Suppresses field if not entering common information for group service. See the notes for the G option above.

M – Suppresses field if entering common information for multiple students, but for individual student service, not group service. Note that prior to Version 11.0, selecting multiple students always implied group service, however, it is now possible to enable an option in the service record profile form section that supports a new scenario where the user selects multiple students and then creates individual (not group) service records at different times on the same day(s).

m – Suppresses field if not entering common information for multiple students, but for individual student service, not group service. See the notes for the M option.

FYI – You can control the inclusion of arbitrary form content using the `{#IF_SERVICECAPTURECONTEXT "options"}content{#ENDIF}` directive. The options here are the same options that are available for the M field modifier.

Customizing Service Capture Instant Reports: If you wish to customize the look of the services listing in the “instant reports” available from the service calendar screen, you can create a second profile form section for service records as follows:

Name the form “Listing” and configure it to have “No Page Breaks When Multiple Forms Printed” as shown below.

Add New Form Section For Service Records

Section Name: Listing

Category: Top-Level Section

Purpose of section: View only

Content generation method: HTML Format

Presentation Options:

- White Background
- No Page Break When Multiple Forms Printed
- Show Profile References With IDs
- Schedule Multiple Individual Service Records At Same Time
- Individualize Service Records When Scheduling in Future
- Mobile Device Accessible
- Mobile Device Only
- Editable From Mobile Device

Insert section where? Between 'Service Form' and end

Accept **Cancel**

Include which profile fields?

<input checked="" type="checkbox"/> Student (Profile Reference)	<input type="checkbox"/> Profile_Modified_On (Date Time)	<input checked="" type="checkbox"/> ProgressNotes (Short Text)
<input checked="" type="checkbox"/> Staff (Profile Reference)	<input checked="" type="checkbox"/> Completed (Logical Value)	<input checked="" type="checkbox"/> Service (Keyword Selection)
<input checked="" type="checkbox"/> ServiceDateTime (Date Time)	<input checked="" type="checkbox"/> DurationMinutes (Integer)	<input checked="" type="checkbox"/> SessionSize (Integer)
<input type="checkbox"/> Profile_Created_On (Date Time)		

Accept **Cancel**

Use the {#TABLE_HEADER} and {#TABLE_FOOTER} directives to define the header and footer regions as described in Appendix A, and be sure to follow the best practices described there (e.g. for the <thead> tag).

In the header region, you can use the pseudo-field directives {_ReportTitle} and {_DateRange} to incorporate the report title and/or the date range of services being reported respectively. Both of these directives will generate labels for the pseudo-field. You can use the T-modifier to override this behavior by using, for example, {_ReportTitle:T""} specifies that the report title field will have no label.

For the HTML format between the header and footer, PowerSchool Special Programs will inject CSS style information into the first tag in certain circumstances, for example, to highlight a service record that the user has drilled into. It is best to not specify an explicit CSS style attribute in the first tag as this will prevent PowerSchool Special Programs from injecting that CSS information for highlighting purposes.

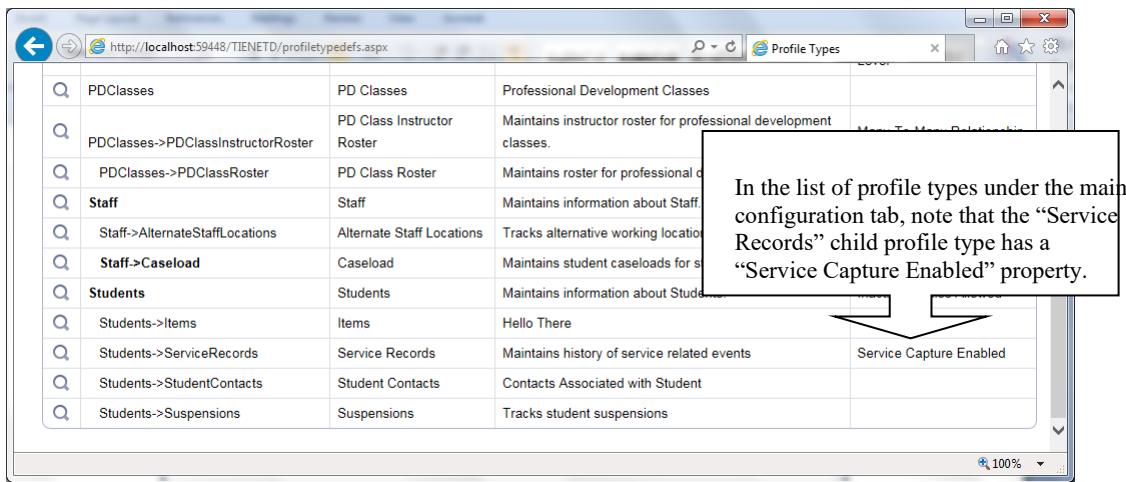
If you decide that a different header is needed for the services listing for a student versus the services listing for a staff member (provider), PowerSchool Special Programs will also recognize two separate sections named “Staff Services Listing” and “Student Services Listing”.

Finally, there is a standard caption area that will always appear above the custom header unless one specifically takes steps to integrate the caption text into the custom header. To do that, configure two pseudo-tags into the custom header, namely <caption /> and <information />, to serve as placeholders for the text that is normally in the standard caption area.

Optional Special Service Capture Fields: The service capture module can also recognize certain student profile fields as a source of authorized service information. Generally, this service information in the student profile would be supplied by flow-back from the IEP document. The following fields are recognized, all of which currently must be multiple value with the same keyword table delineating the values for all these fields. Note that they can be calculated fields, which may be useful in certain specialized circumstances.

- **Service or SpEdService:** Can be a logical multiple-value field or a keyword selection multiple value field. If this is a logical field, then it is assumed that each of the multiple values corresponds to a specific service and the logical value indicates whether or not the student has the service. If this is a keyword field, then it is assumed that the multiple values represent potential slots and the keyword field values identify the specific services.
- **DurationIndividual, ServiceDuration or ServiceMinutesWeekly:** Can be of type integer or numeric (integer is preferred) and is assumed to specify the authorized number of minutes. By default, weekly minutes is assumed; however, a separate frequency field can also be recognized.
- **ServiceFrequency or FrequencySpEdSvc:** This field must be a multiple-value field of type keyword and is assumed to specify a frequency (weekly, monthly, etc.). The keyword table this field references would delineates the different frequency possibilities (weekly, monthly, etc). If a keyword starts with “W”, the service capture module takes that to mean weekly. If a keyword starts with “M”, the service capture module takes that to mean monthly. At the present time, the service capture module makes use of weekly and monthly only in the instant report showing authorized services. On this instant report, services for which the student is authorized with a weekly frequency will show weekly totals, and services for which the student is authorized with a monthly frequency will show monthly totals.

“Service Capture Enabled” property: If you look at the list of profile types under the main configuration tab, you will notice that the Service Records child profile type has a “Service Capture Enabled” property as shown below. This property is set when you create the child profile type using the “Special: Service Capture Enabled Child Profile Type” option and cannot be set in another way.



The screenshot shows a web-based application interface for managing profile types. A callout box highlights the 'Service Capture Enabled' checkbox for the 'Service Records' profile type in the list.

Q	PDClasses	PD Classes	Professional Development Classes
Q	PDClasses->PDClassInstructorRoster	PD Class Instructor Roster	Maintains instructor roster for professional development classes.
Q	PDClasses->PDClassRoster	PD Class Roster	Maintains roster for professional development classes.
Q	Staff	Staff	Maintains information about Staff.
Q	Staff->AlternateStaffLocations	Alternate Staff Locations	Tracks alternative working locations for staff.
Q	Staff->Caseload	Caseload	Maintains student caseloads for staff.
Q	Students	Students	Maintains information about Students.
Q	Students->Items	Items	Hello There
Q	Students->ServiceRecords	Service Records	Maintains history of service related events
Q	Students->StudentContacts	Student Contacts	Contacts Associated with Student
Q	Students->Suspensions	Suspensions	Tracks student suspensions

Coloring Service Records: Service records in the service calendar can be colorized by adding a calculated “ServiceColor” character field that calculates the color based on criteria. It is recommended that the calculated field return any of the following values: red, orange, blue, green, yellow, black, white, pink, brown, gray, cyan or purple.

Preventing Duplicate Service Records: Duplicated service records can be prevented by imposing a unique key on the service records profile type using the fields that you don’t want duplicates of, but keep in mind that any fields in the unique key are required to have a value in every single record (even those scheduled in the future). A unique key may be problematic to set if it is not set from the very beginning. If a unique key is not set, the system will automatically check for duplicates based on the following fields: Student, Staff, ServiceStartTime and Service.

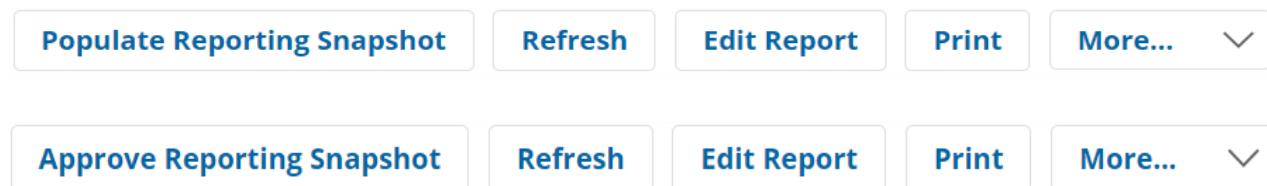
Service Capture Privileges: Please note that privileges such as View Service Records, Edit Service Records, and Delete Service Records will authorize users to interact with service record data directly instead of through the service capture module. Therefore, such privileges should only be assigned to system administrators. Instead, the following privileges are specifically for authorizing users to use the service capture module:

- **Student Privileges:** Note that if the child template is named something other than “Service Records”, the privileges below will be named accordingly.
 - **Caseload – Capture My Service Records:** A service provider with this privilege will be able to access the Service Capture module and record services for students on the staff member’s caseload. Note that such a user should also be assigned relevant caseload access privileges described in the previous section. A user with this service capture privilege must also have the “Access My Caseload” privilege.

- **Non Caseload – Capture My Service Records:** A service provider with this privilege will be able to access the Service Capture module and record services for any students the user is authorized to view (i.e. through the student search).
- **Schedule Future Services:** This privilege will only appear if your school district is using the service capture module. A service provider with this privilege will be able to schedule future services on the calendar and when the scheduled time arrives, the service provider can complete the record of those scheduled services.
- **Special Access Privileges:**
 - **View Staff Service Calendars:** This privilege falls under the “Special Access Privileges” category. It can be assigned to supervisors to enable them to see the service calendars (and associated reports) for other staff members. The privilege can be assigned at different levels in the organization, e.g. location-wide, district-wide, system-wide, etc.

Reporting Snapshot Approval Process

This section describes functionality that supports a simple reporting snapshot process where certain student data is captured at a particular point in time in a child profile, can be edited if necessary, and then is approved and locked down for submission to a receiving authority. At a high level, this involves configuring a child profile type of the Students profile type that will be populated with snapshot data. The user interface to the functionality is a list report in which a “Populate Reporting Snapshot” button appears in the toolbar, and once populated, the button changes to an “Approve Reporting Snapshot” button.



To enable this kind of functionality, the following elements must be configured:

1. **Child Profile Type:** A child profile type with all the columns necessary to store the snapshot as required for the reporting submission. The child profile type will be populated using a profile update script to be described later. The child profile type must have the “Use for Report Snapshots?” property set to yes in order to support the process described here.

Use For Report Snapshots?

Yes No

Additionally, the child profile type must have an “AllowEdit” logical field with a default value of true. This field will play a critical role in the approval process. By defaulting its value to true, snapshot rows will be editable initially. But when the “Approve Reporting Snapshot” button mentioned above is clicked, the AllowEdit column will be set to false for all rows. An

alternative is to add an ApprovalDate (date/time) field to the child profile that is set when the snapshot is approved, and the AllowEdit logical field can be calculated based on “ApprovalDate IS EMPTY”. It is also recommended to add an AllowDelete logical field to the child profile prevent the user from deleting child profiles after approval. Finally, you may want to add a “AllowAdd<childprofiletypename>” logical field to the top level profile (e.g. Students) if you want to prevent manual addition of child profiles to the snapshot. Note that at this time, AllowEdit and AllowDelete are only enforced for staff users.

2. **List Report:** Implement a list report based on the snapshot child profile type that presents the state reporting data. Be sure to set the “Optimize for Editing Profiles” property in report properties to allow the user to edit report rows prior to approval. To support report administrators previewing data long before the snapshot is created, consider having two reports: a preview report that reports directly on the live data that will be put into the snapshot that can be reviewed at any time, and the actual report that reports on the snapshot itself.
3. Profile update scripts that support the various operations needed to support the approval process.
 - **Populate Snapshot Script:** Script to populate the snapshot data which must be named according to the convention “Populate Snapshot: <child profile type plural name>”. This should clear any data currently in the child profile and then repopulate it. Failure to include the line to clear data could result in duplicate records

```
DELETEFROM StateReportingRecords
INSERT StateReportingRecords (Student, Field1, Field2) from Students (This,
Value1, Value2) where ???
```

- **Approve Snapshot Script:** Script to approve the snapshot data which must be named according to the convention “Approve Snapshot: <child profile type plural name>”. Normally this simply sets AllowEdit and potentially AllowDelete to false for all records, but another option is to set an ApprovalDate field and let AllowEdit and AllowDelete be based on that.

```
UPDATE StateReportingRecords SET AllowEdit=FALSE
```

- **Clear Snapshot Script (optional):** Optional script to delete all snapshot data which must be named according to the convention “Clear Snapshot: <child profile type plural name>”. This is more of an edge case that allows the administrator to select “Clear Snapshot Data” from the more dropdown of the list report. Note that this option becomes unavailable after the snapshot is approved. To handle clearing of the approved snapshot at the end of the school year to prepare for the next school year, this same script is typically scheduled to execute after the end of year rollover (thereby serving a double purpose).

```
DELETEFROM StateReportingRecords
```

- **Unapprove Script (optional):** Support for an unapproved script is currently targeted towards a state level setup with multiple tenants. You can create an unapproved script according to the convention “Unapprove Snapshot: <child profile type plural name>”. This will allow a state level administrator to select “Unapprove Snapshot” from the more dropdown of the list report where a state level administrator is defined as a “named” ADMIN or CONSULTANT user (i.e. a user who can log into multiple tenants).

```
UPDATE StateReportingRecords SET AllowEdit=True
```

4. **Due Date (optional):** If you want to establish a due date for the snapshot to be taken, configure a date field in Globals named according to the convention “<child profile type plural name>Due”. Typically, this would be a calculated date field and marked as internal. When a date is configured and specified this way, the due date will be displayed on the “Populate Reporting Snapshot” button. The “Approve Reporting Snapshot” button will be disabled and include the due date prior to the due date, and then become enabled on or after the due date. **Note:** if you plan to use this feature, make sure the plural name of the child profile is three characters less than the limit of 25 to account for the “Due” suffix. Alternatively, the system will also recognize the field if it uses with the “<child profile type singular name>Due” pattern.

Security: Normally, only CONSULTANT/ADMIN users can populate the snapshot data and click the “Approve Reporting Snapshot” button. But it is also possible to authorize a staff security group to do this by making sure the security group has the following privileges: 1) the “Administrate Snapshot Reporting” reports security privilege, and 2) the system-wide editing privilege for the child profile type used for the snapshot data.

Troubleshooting: If the buttons are not appearing in the expected list report, check the following: 1) child profile type has “Use for Report Snapshots?” property set to Yes, and has an AllowEdit column defaulted to true (assuming it is not a calculated field), 2) the list report is pointing to the correct child profile type, and 3) the profile update scripts exist and have the correct names.

State/Regional Module Related Configuration

Configuration settings are available that allow a model to behave optimally in the context of the state/regional controller module. These options are currently all related to “merged” list reports (list report results merged from multiple district tenants).

One such setting is in the configuration screen for list report sort values. Normally, when a list report is merged across multiple tenants, the report rows appear grouped by district tenant as if district tenant is the main sort value, and within each district tenant, the configured sort values apply. But for certain list reports, you may wish the configured sort values to be applied across district tenants without grouping by

district tenant. In this case, the checkbox option in the red rectangle below can be checked:

Sort Values
Note: A name sort is automatically added. It is not necessary to specify a name sort.

Sort Value 1 Student Formula
 (field name or calculation)
 Descending Sort Page Break

Sort Value 2 Student Formula
 (field name or calculation)
 Descending Sort Page Break

Sort Value 3 Student Formula
 (field name or calculation)
 Descending Sort Page Break

State/Regional Module Options Do Not Group By District Tenant When Merging District Tenants

Accept **Cancel**

Duplicate Inactive Profile Elimination: If a list report includes inactive profiles and more than one district tenant has the same student in an inactive state, a merged report can show more than one profile for the same student (i.e. coming from different district tenants). To eliminate this and show only one profile per student, you need to first set up a duplicate inactive profile elimination expression by following the steps below:

1. Log into the model and go to the configuration for the relevant profile type being reported on (typically Student Profiles).
2. Select More -> Set State/Regional Module Settings
3. Check the “Eliminate Duplicate Inactive Profiles when Merging Tenants for State/Regional Module” checkbox.
4. Define an expression that the PSSP application can use to determine which of two inactive profiles from different tenants should be included in the merged report versus omitted. The expression will only be applied to inactive profiles and is generally used to determine which profile has a higher priority than the other. For example, the expression can compute the date the student was last enrolled in the respective tenant. The radio buttons allow you to take the inactive profile with either the larger value of the expression or the lower value of the expression. These settings are intended to provide a significant level of flexibility in determining which of two duplicate profiles will be included or omitted from the merged report in order to show only one profile per student.

The settings described above are shipped out as part of a model release. Once a duplicate inactive profile elimination expression is set, a new checkbox option appears in the report selection criteria dialog box (see red rectangle below). This checkbox, which is enabled only when “Include Inactive Profiles” is enabled, allows you to turn on duplicate profile elimination for a list report easily both in model reports and customer reports synced out via the state/regional controller sync.

Selection Formula for Report: My Date Time Example

Student Selection Formula Quick Formula ▾

Report Selection Formula

Report Selection Description

Selection Options

- Include Only 'Not Exited' Students
- Include Inactive Students (Only) (Eliminate Duplicate Inactive Profiles During Merged Reporting)
- Allow User Filtering Based on Organizational Location

Implementing Custom SQL User Defined Functions (UDFs)

In this context, User Defined Function or UDF is a Microsoft technical term for a custom SQL function that is added to SQL Server at the backend. It is possible to create a UDF and “register” it with PowerSchool Special Programs so that it can be used as a function in the PowerSchool Special Programs formula language. This effectively creates a new formula function that can take zero or more parameters and returns a single value.

Although custom UDFs can add considerable power to PowerSchool Special Programs and can address complex specialized scenarios, they should only be used as a last resort and after considering the following:

Implementing custom UDFs should only be done by a programmer with expertise in SQL Server, the SQL language, and maximizing SQL performance. Improperly programmed UDFs can cause bugs, errors and/or performance problems in PowerSchool Special Programs.

Much attention should be paid to implementing the UDF in the most efficient way including utilizing available indexes, avoiding use of temporary tables within the UDF, avoiding full table scans, etc.

Custom UDFs require specialized maintenance in the future. Although PowerSchool Special Programs has some support in the CMT for handling registered UDFs, there is very limited ability to automatically test them, verify them, etc.

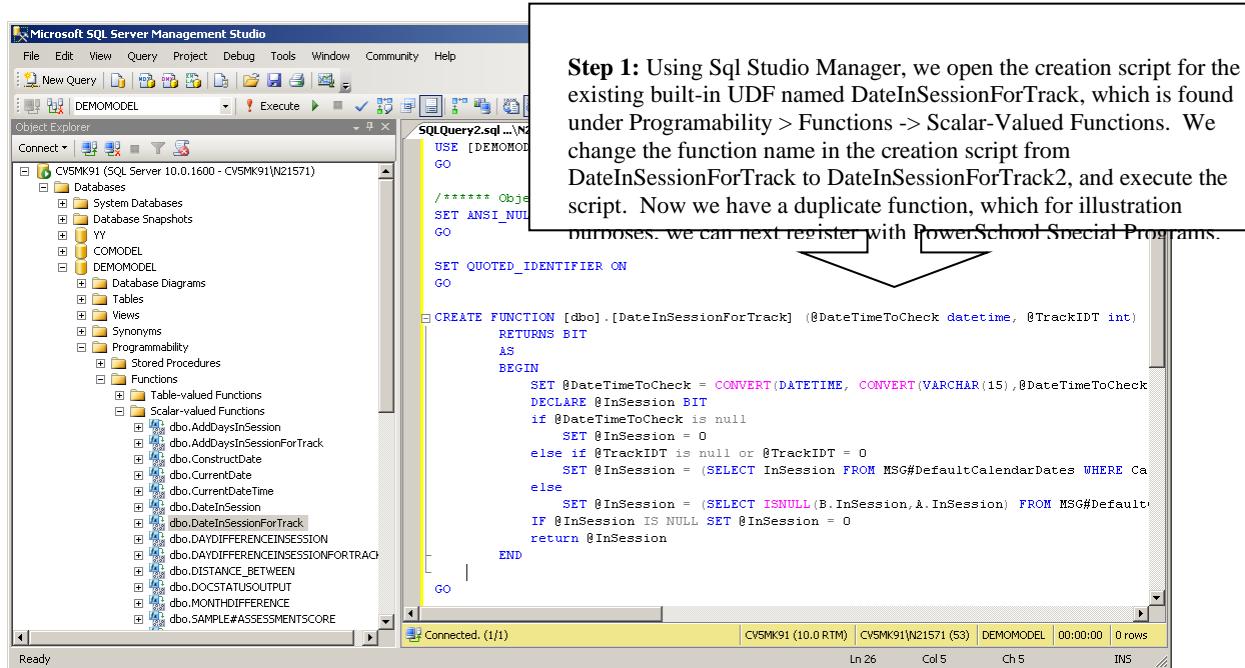
In many cases, the same results can be achieved through normal calculated fields using aggregate functions. Custom UDFs should only be used as a last resort. If used, they should be reviewed with

future versions of PowerSchool Special Programs in case new standard features can achieve the same result.

Use of custom UDFs can create schema dependencies in the database that may cause unexpected exceptions when configuring other schema objects that are referenced by the custom UDFs.

The first step in creating a UDF is to identify the name and data type of each parameter, and the data type of the return value. It is helpful to define the data types as PowerSchool Special Programs data types (character, logical, etc) which can then be translated to SQL data types using translation tables provided later in this section. Additionally, if the first parameter is a profile reference, the function can be used in a self-referential way from the context of that profile type. For example, if there is a function *integer* CaseloadCapacity(*StaffReference*), then the function can be used from the staff context using CaseloadCapacity() or from another context using CaseloadCapacity(*StaffField*).

The process is best documented using an example. In this example, we will duplicate the functionality of a built-in PowerSchool Special Programs function named DateInSessionForTrack(*datetime*, *track*).



Below is a typical looking UDF creation script. Note the names and data types of the parameters and the data type of the return value. More explanation follows the script. **Important:** In the SQL script that creates a UDF, there should be no other statements other than the function creation statement. Other statements, if not eliminated, may accidentally become part of the function.

```

CREATE FUNCTION [dbo].[DateInSessionForTrack2] (@DateTimeToCheck datetime, @TrackIDT
int)
RETURNS BIT
AS
BEGIN

```

```

SET @DateTimeToCheck = CONVERT(DATETIME,
CONVERT(VARCHAR(15),@DateTimeToCheck,101))
DECLARE @InSession BIT
if @DateTimeToCheck is null
    SET @InSession = 0
else if @TrackIDT is null or @TrackIDT = 0
    SET @InSession = (SELECT InSession FROM
MSG#DefaultCalendarDates WHERE CalendarDate=@DateTimeToCheck)
else
    SET @InSession = (SELECT ISNULL(B.InSession,A.InSession) FROM
MSG#DefaultCalendarDates AS A LEFT JOIN MSG#Locations#CalendarDates AS B ON
A.CalendarDate=B.CalendarDate AND B.TrackIDT=@TrackIDT WHERE A.CalendarDate =
@DateTimeToCheck)
IF @InSession IS NULL SET @InSession = 0
return @InSession
END

```

The table below shows the correct SQL data types to use for equivalent PowerSchool Special Programs data types. It is also essential that the UDF properly handles the values that PowerSchool Special Programs may pass into it for a given data type) and also to only return valid values for the PowerSchool Special Programs data type of the return value. The chart below should be helpful in understanding what the valid values are.

PowerSchool Special Programs Data Types	SQL Data Type	Valid Values
Character (or ID, ShortText)	VARCHAR(<i>maxlength</i>)	Any string value. However NULL is not used for this data type and should not be returned.
Profile Reference	INT	Foreign key reference to profile IDT#. May be NULL.
Keyword Selection	INT	Foreign key reference to keyword IDT#. May be NULL.
Logical	BIT	0, 1, NULL
Integer	INT	Any integer or NULL
Numeric (or Score)	DECIMAL(9,2)	Any decimal number or NULL.
Date	DATETIME	Any date with no time component, or NULL.
DateTime	DATETIME	Any datetime value or NULL.
SchoolYear	INT	First part of school year value pair (e.g. 2005-06 is 2005) or NULL.

DocStatus	INT	1=Draft, 2=Review, 3=Final
-----------	-----	----------------------------

Next, you must register the UDF by inserting a row into the PRM#CustomUDFs table. You must also register each parameter by inserting rows in proper sequence into PRM#CustomUDFParms. Below is an example script for our DateInSessionForTrack2 example. More explanation follows the script.

```

DECLARE @idtCustomUDF INT, @idtKeywordTable INT

--Register the UDF
INSERT PRM#CustomUDFs (Name, Description, DataType, ProfileTypeIDTRef,
KeywordTableIDTSel) VALUES
    ('DateInSessionForTrack2', 'Determines if specified date is in session in
specified track', 3, NULL, NULL)
SET @idtCustomUDF=@@IDENTITY

--Register parameter #0
INSERT PRM#CustomUDFParms (CustomUDFIDT, ParmSequence, ParmName, DataType,
ProfileTypeIDTRef, KeywordTableIDTSel) VALUES
    (@idtCustomUDF, 0, 'DateTimeToCheck', 5, NULL, NULL)

--Register parameter #1
SET @idtKeywordTable = (SELECT KeywordTableIDT FROM PRM#KeywordTableDefs WHERE
Name='CalendarTracksTable')
if @idtKeywordTable IS NULL
    PRINT 'Incorrect keyword table name'
ELSE
    INSERT PRM#CustomUDFParms (CustomUDFIDT, ParmSequence, ParmName, DataType,
ProfileTypeIDTRef, KeywordTableIDTSel) VALUES
        (@idtCustomUDF, 1, 'TrackIDT', 11, NULL, @idtKeywordTable)

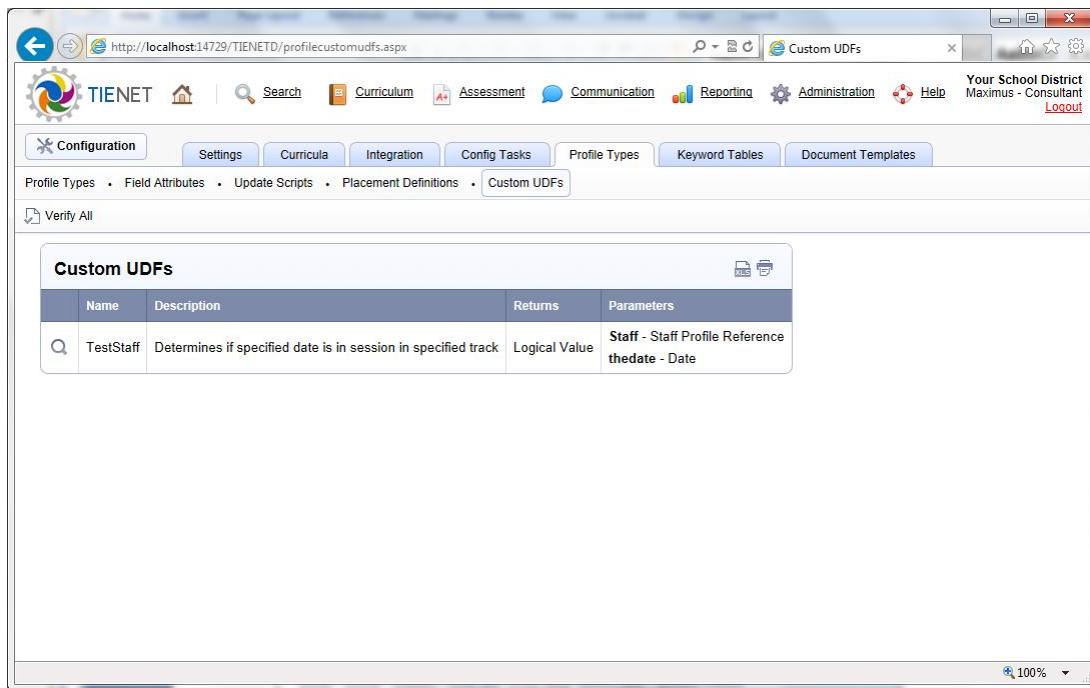
```

The PRM#CustomUDFs table identifies the PowerSchool Special Programs data type in the DataType, ProfileTypeIDTRef, and KeywordTableIDTSel columns. In the same way, the PRM#CustomUDFParms table identifies the PowerSchool Special Programs data type of each parameter. It is essentially to set the values properly in these tables. The chart below delineates the details. Note that KeywordTableIDTSel references a keyword table for the “keyword selection” data type, but is NULL for anything else. Similarly ProfileTypeIDTRef references a top level profile type for the “profile reference” data type, but is otherwise NULL

PowerSchool Special Programs Data Types	DataType Value	ProfileTypeIDTRef, KeywordTableIDTSel
Character	2	Both NULL
Profile Reference	13	ProfileTypeIDTRef =profile type identifier KeywordTableIDTSel = NULL
Keyword Selection	11	ProfileTypeIDTRef =NULL KeywordTableIDTSel = keyword table identifier
Logical	3	Both NULL
Integer	8	Both NULL
Numeric	6	Both NULL
Date	4	Both NULL
DateTime	5	Both NULL
SchoolYear	15	Both NULL
DocStatus	16	Both NULL

Once the UDF is created and properly registered, it can be viewed and inspected from the PowerSchool Special Programs web UI, as shown below, and the UDF can be used in formulas. Additionally, at the time of writing, work was being done on the Configuration Management Tool to allow it to fully compare and synchronize registered custom UDFs.

PROFILE DATA SCHEMA CUSTOMIZATION (ADVANCED)



Custom UDFs can be viewed from the PowerSchool Special Programs configuration UI

Assessment Repository Customization

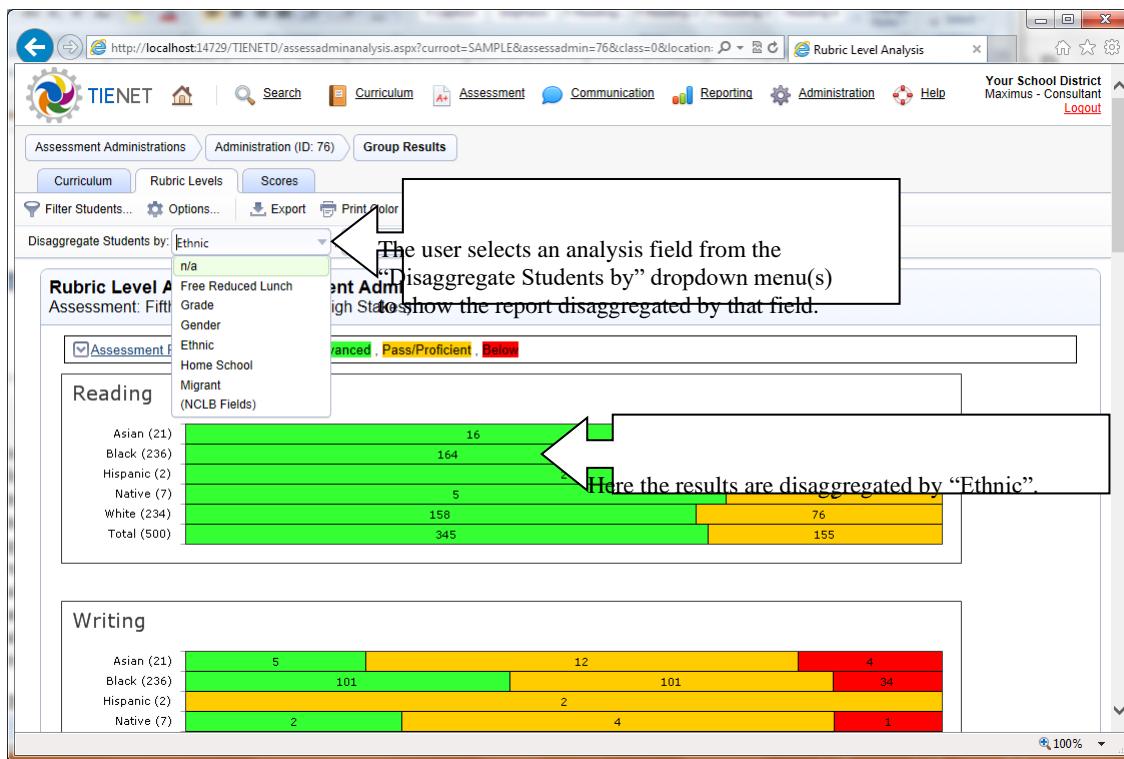
The procedures described in this chapter only apply if a PowerSchool Special Programs database has been configured to allow student assessment.

Chapter

5

Configuring Analysis Fields in the Assessment Repository

To meet student assessment reporting requirements of school districts, such as those related to “No Child Left Behind”, analysis fields (e.g. Ethnic, Gender, Free/Reduced Lunch, etc) can be set up in the assessment repository to enable filtering and disaggregating of results. An example of an assessment reporting screen is shown below. In this example, the user selects one of the analysis fields to disaggregate the results.

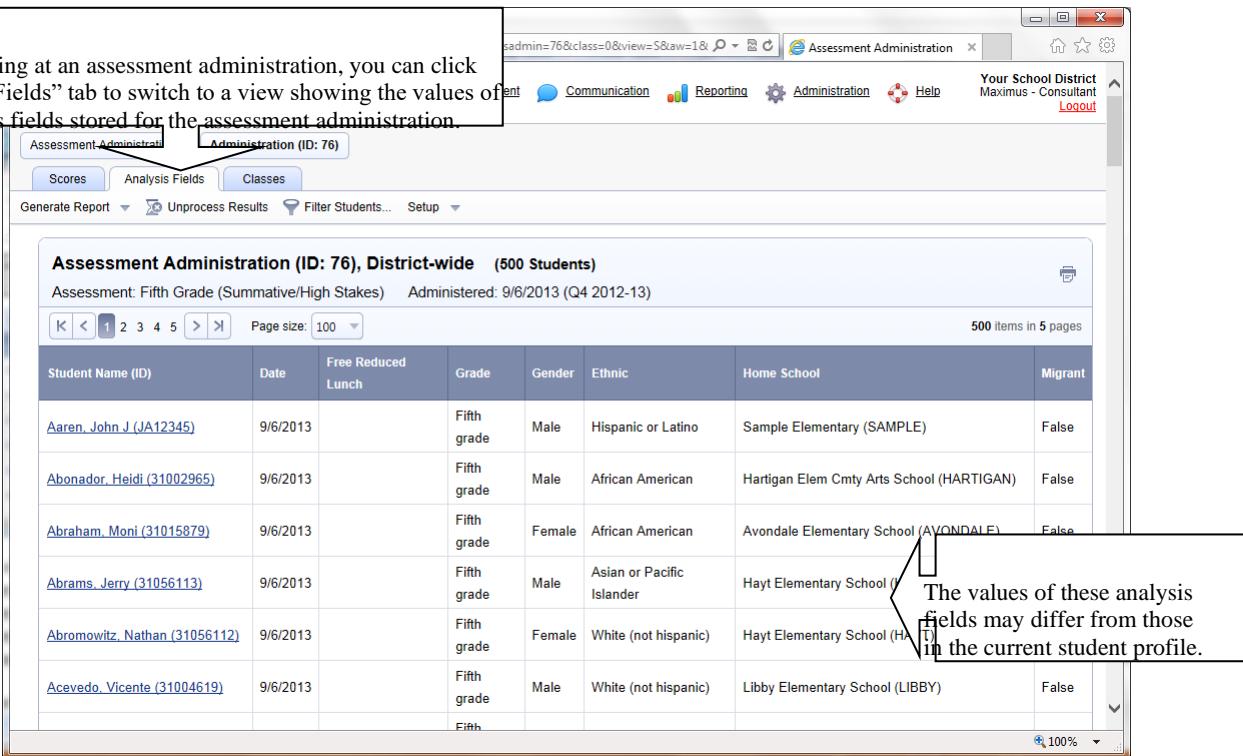


It is critical to understand that whenever a student result is added to an assessment administration, the values for each of the analysis fields are copied from the current student profile into the assessment administration. In effect, a “snapshot” of the field values are copied at the time the student was administered the test and preserved historically. For fields that are volatile and change over time, this is important in maintaining historical reporting accuracy. Three years down the line, longitudinal reports may

be needed that show yearly progress for three years. Such a report should be based on what location students were in when they took the assessments, not the location they are currently in (which has no bearing on their result three years ago). Some students may leave the district and eventually have their student profiles purged from the database, but nevertheless, the assessment repository retains a snapshot of their ID, name, analysis field values and score information in each assessment administration.

FYI – About Home School or Primary Location Field: It is critical that the field containing the student's home school or primary location be configured as one of the analysis fields. The assessment repository will not be fully functional otherwise.

When looking at an assessment administration, you can click "Analysis Fields" tab to switch to a view showing the values of the analysis fields stored for the assessment administration.



The screenshot shows the 'Assessment Administration' interface with 'Administration (ID: 76)' selected. A callout points to the 'Analysis Fields' tab in the top navigation bar. Another callout points to a box around the 'Home School' column in the main table, which contains values like 'Sample Elementary (SAMPLE)', 'Hartigan Elem Cnty Arts School (HARTIGAN)', 'Avondale Elementary School (AVONDALE)', 'Hayt Elementary School (HAYT)', and 'Libby Elementary School (LIBBY)'. A note to the right of this table states: 'The values of these analysis fields may differ from those in the current student profile.'

Student Name (ID)	Date	Free Reduced Lunch	Grade	Gender	Ethnic	Home School	Migrant
Aaren, John J (JA12345)	9/6/2013		Fifth grade	Male	Hispanic or Latino	Sample Elementary (SAMPLE)	False
Abonador, Heidi (31002965)	9/6/2013		Fifth grade	Male	African American	Hartigan Elem Cnty Arts School (HARTIGAN)	False
Abraham, Moni (31015879)	9/6/2013		Fifth grade	Female	African American	Avondale Elementary School (AVONDALE)	False
Abrams, Jerry (31056113)	9/6/2013		Fifth grade	Male	Asian or Pacific Islander	Hayt Elementary School (HAYT)	False
Abromowitz, Nathan (31056112)	9/6/2013		Fifth grade	Female	White (not hispanic)	Hayt Elementary School (HAYT)	False
Acevedo, Vicente (31004619)	9/6/2013		Fifth grade	Male	White (not hispanic)	Libby Elementary School (LIBBY)	False
			Fifth				

To drive these points home with a specific example, if the “Grade” field is set up as an analysis field (and it normally is), a student will have a current “Grade” in the student profile and also a distinct “Grade” stored in each assessment administration having results for that student. Generally, some extra care must be taken to ensure all these values are accurate. For this reason, it is helpful to know exactly when values are copied from the current student profile into an assessment administration. There are three cases:

The values are always copied when the student is initially added to an assessment administration.

The values are updated (copied) again when the assessment administration is processed, but only if the assessment administration is in the current school year.

Lastly, an administrative tool (described later in this section) can be used to explicitly synchronize analysis field values to values in the current student profile.

The following illustrates how to set up analysis fields or synchronize them to current values in the student profile:

Step 1: Log in as a system administrator level user, access Configuration from the main menu, and then click Curricula here.

Step 2: Click Set up analysis fields for the curriculum you are working with. If this option does not appear, then student assessment is not enabled for this curriculum.

Curricula

SAMPLE Curriculum [SAMPLE] (Active)

Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Enabled ([Disable Now](#)) • Legacy "One-Click" Progress Reports: Enabled ([Disable Now](#)) - Screening Groups: [Columns](#)

Progress Monitoring Groups: Enabled ([Disable Now](#)) - [Settings](#), [Periods](#), [Reports](#)

Assessment Analysis Fields: Free Reduced Lunch, Grade, Gender, Ethnic, Home School, Migrant ([Set Up Analysis Fields](#))

Student Portfolio Definitions: Graduation Portfolio ([Set Up Portfolio Definitions](#))

Curriculum outline editable by: *Users with access privileges*

Resource library editable by: *Users with access privileges* ([Change Security Mode](#))

Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)

AnyState Core Curriculum Standards [ASCCS] (In Progress)

Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Not Enabled ([Enable Now](#)) • Legacy "One-Click" Progress Reports: Not Enabled ([Enable Now](#))

Progress Monitoring Groups: Not Enabled ([Enable Now](#))

Assessment Analysis Fields: Ethnic, Grade, Gender, Migrant, Home

Student Portfolio Definitions: None ([Set Up Portfolio Definitions](#))

Curriculum outline editable by: *Users with access privileges*

Resource library editable by: *Users with access privileges* ([Change Security Mode](#))

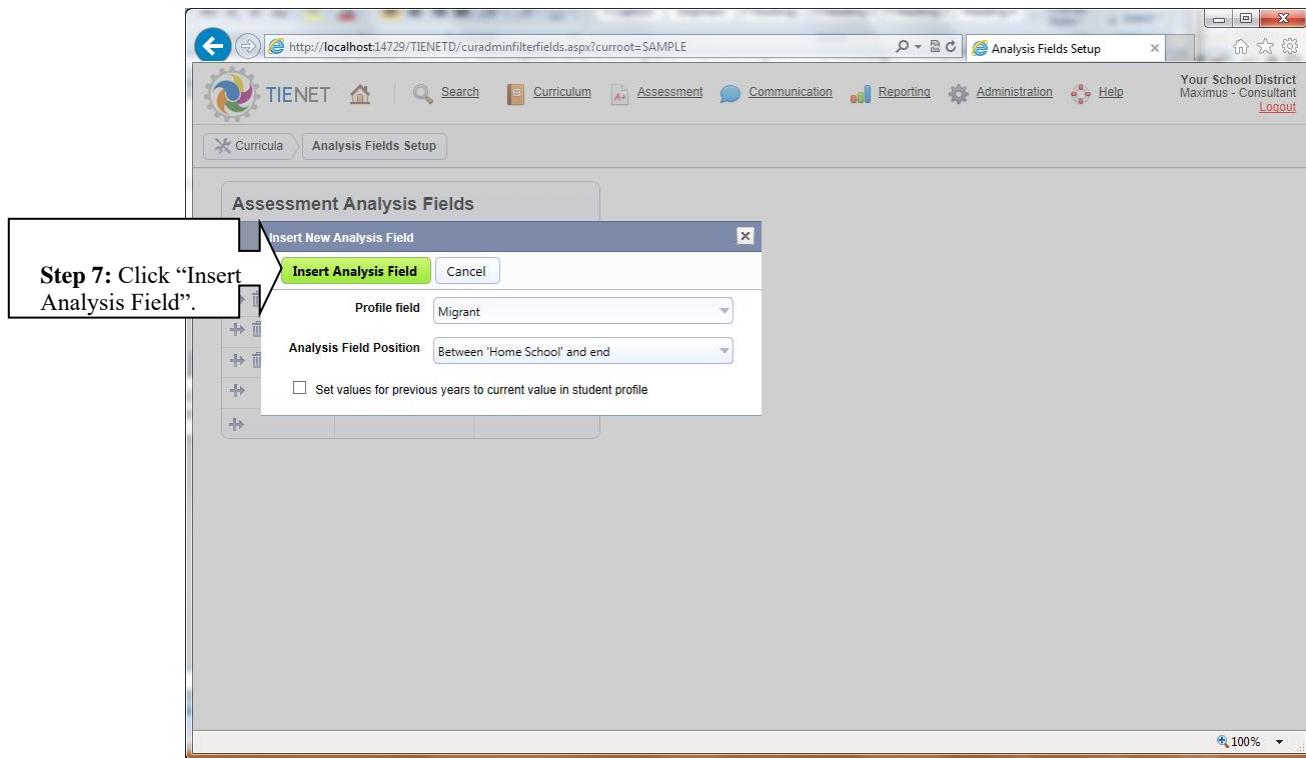
Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)

Step 3: These icons allow you to insert, delete, change the order of, and synchronize analysis fields. Analysis fields are always listed in a particular order which can be seen and changed here.

Step 4: To add a new analysis field, click the insert icon here.

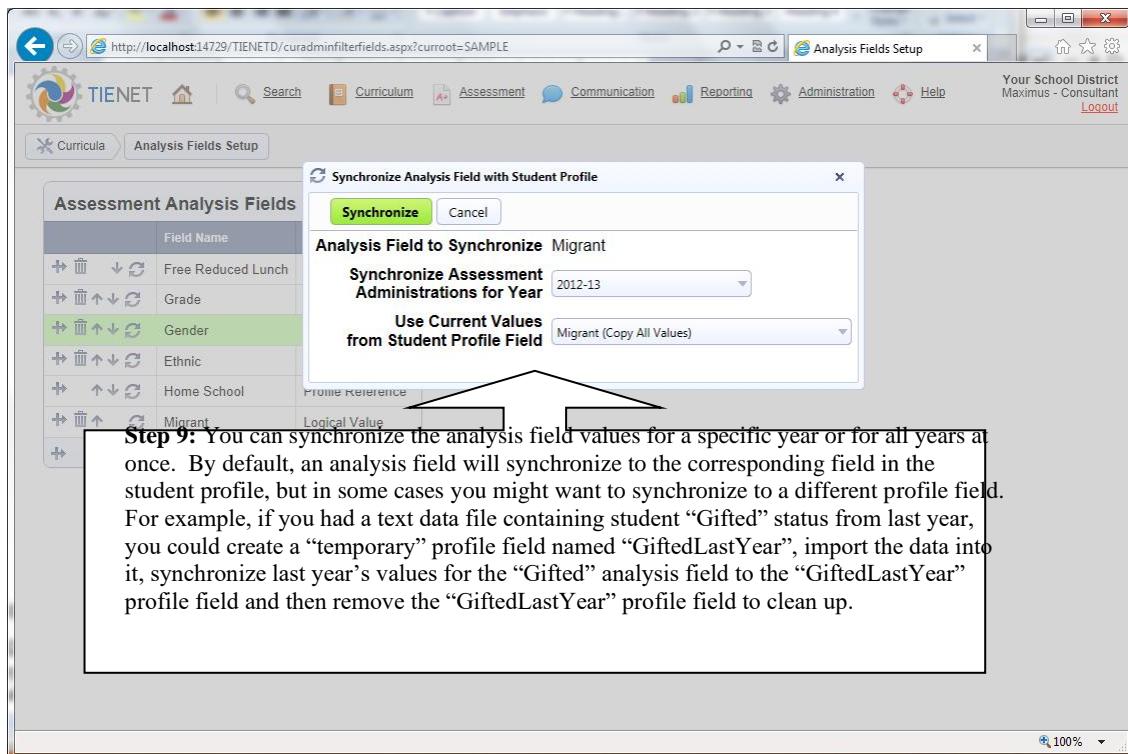
Step 5: Select an available field from the student profile. Only data fields that have a data type of keyword selection, profile reference, or logical (yes/no) can be selected. See Understanding the Data Dictionary on page 63 for an explanation of data types.

Step 6: If the field changes little over time (such as Gender), it may be appropriate to check this option (Set values for previous years to current value in student profile) so that analysis values for previous years get set to the current value in the student profile. Otherwise values for previous years are left blank and can only be set with the synchronization tool described next.



Field Name	Data Type
Free Reduced Lunch	Keyword Selection
Grade	Keyword Selection
Gender	Keyword Selection
Ethnic	Keyword Selection
Home School	Profile Reference
Migrant	Logical Value

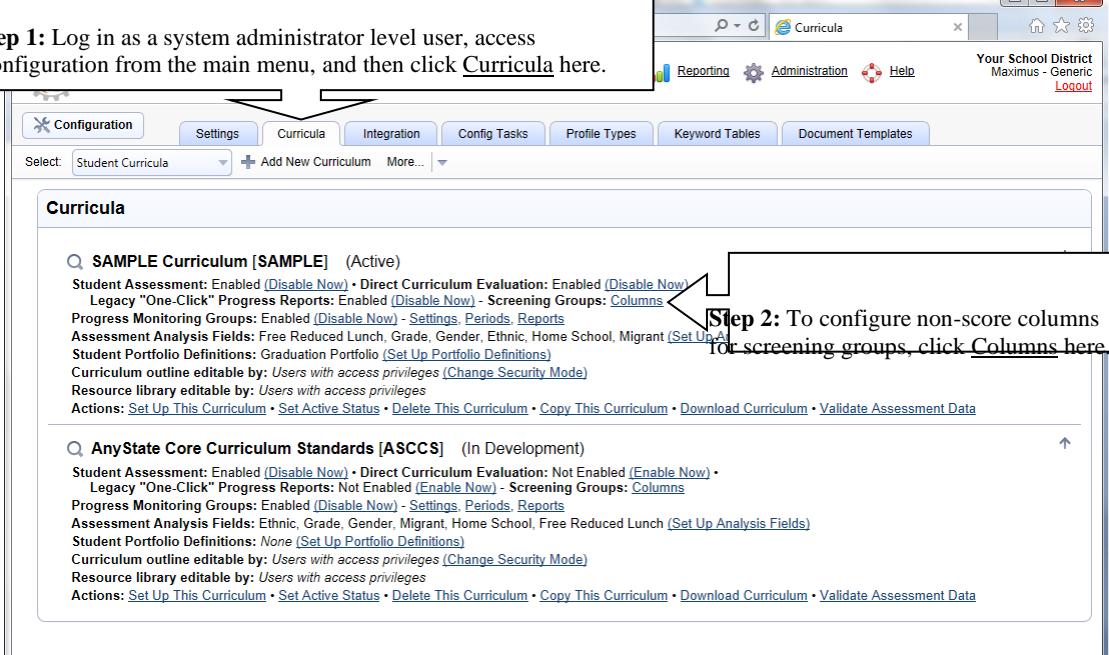
Step 8: The new analysis field is added. Now to see the synchronization capability, click the synchronize icon next to it.



Configuring Screening Groups

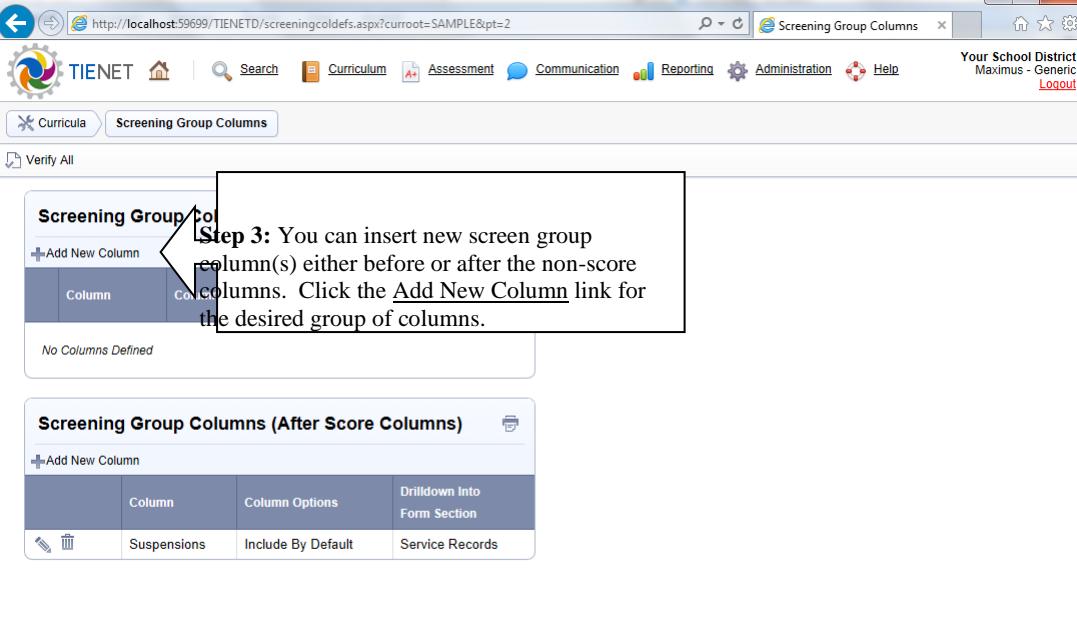
Screening groups do not require any mandatory configuration. Any user who has access to assessment administrations can work with screening groups. However, to allow users to include non-score columns (e.g. behavioral information) in screening groups, then non-score column definitions must be configured.

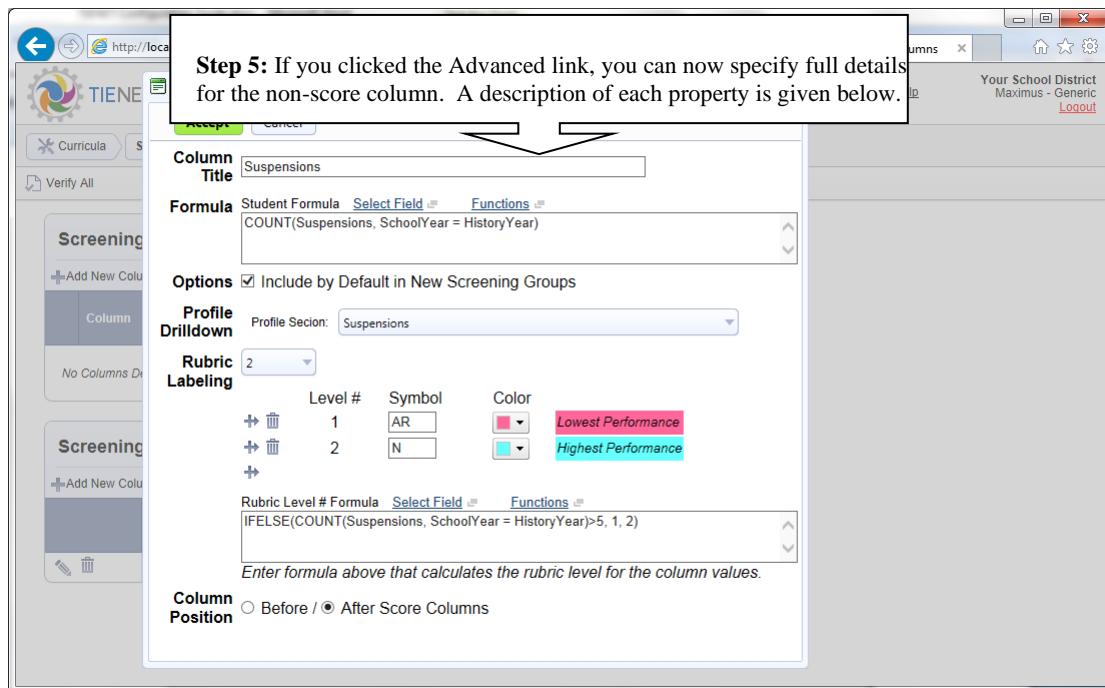
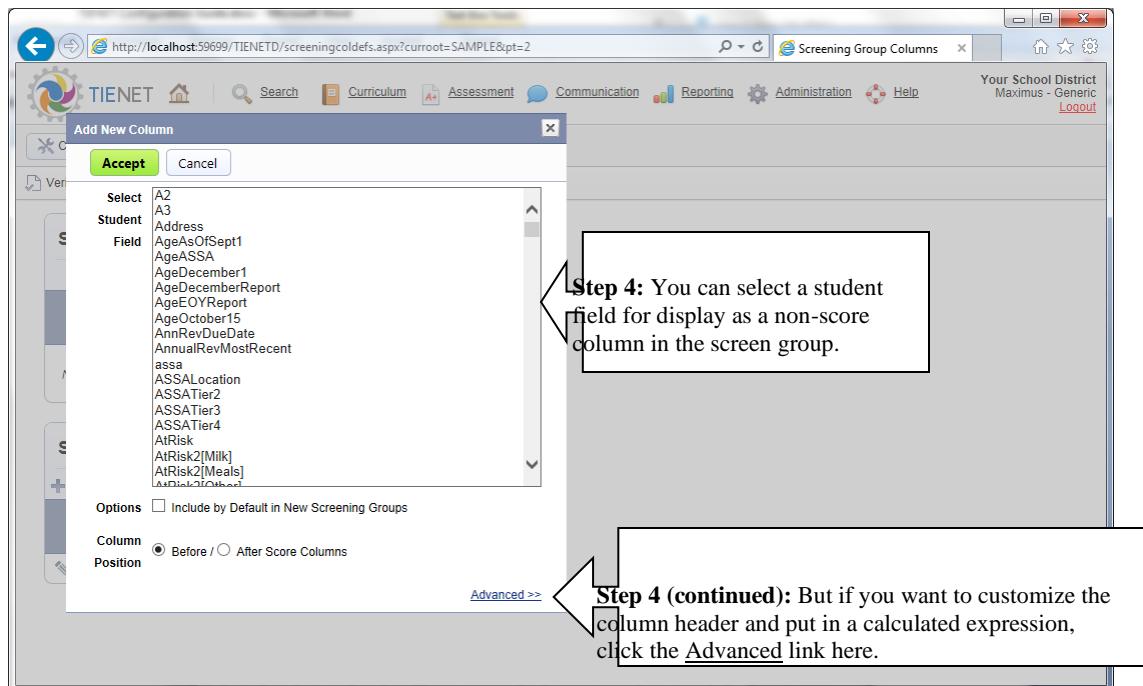
Step 1: Log in as a system administrator level user, access Configuration from the main menu, and then click Curricula here.



Step 2: To configure non-score columns for screening groups, click Columns here.

Step 3: You can insert new screen group column(s) either before or after the non-score columns. Click the Add New Column link for the desired group of columns.





Description of Non-Scoring Column Properties:

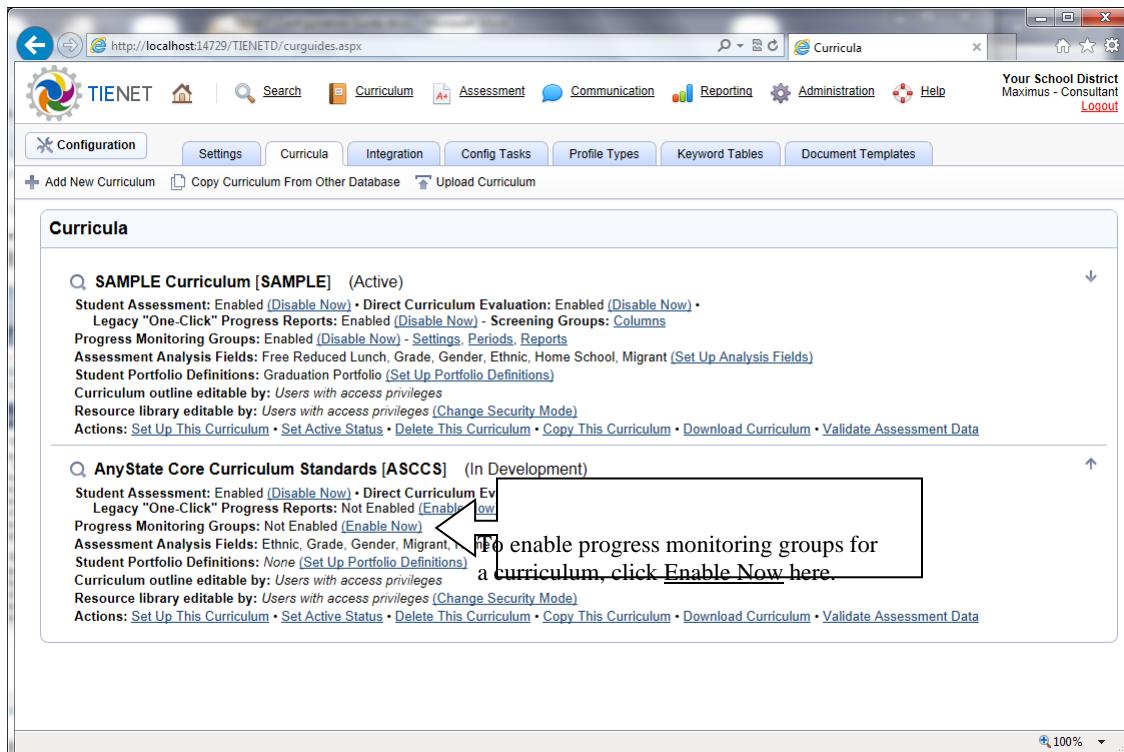
- **Column Title:** Appears as the header of the column in screening grids.
- **Formula:** This computes the value of the column.

- **Profile Drilldown:** This optional selection allows the user to drill down into a particular profile section to obtain more detail about the value represented in the non-score column.
- **Rubric Labeling:** Applicable to numeric non-score columns only. Allows you to define a rubric that interprets the numeric value of the non-score column and provides a label. This also allows for filtering the screening group based on the non-score column. Assign a label and column for each performance level. The “Rubric Level # Formula” calculates the level. For example, if there are three rubric levels defined, this formula should calculate a number from 1 to 3.
- **Column Position:** Non-score columns can appear either before (to the left of) or after (to the right of) the score columns. This property allows you to determine that for this non-score column.

Configuring Progress Monitoring Groups

To configure progress monitoring groups, follow the steps below:

Make sure progress monitoring groups are enabled for the curriculum you are using, as follows. Logged in as someone with administrative rights, click Configuration in the blue sidebar, then Curricula and then Enable Now next to ‘Progress Monitoring Groups’ as shown below.



Configure the settings for progress monitoring groups as shown below:

Curricula

- SAMPLE Curriculum [SAMPLE] (Active)**
 - Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Enabled ([Disable Now](#)) • Legacy "One-Click" Progress Reports: Enabled ([Disable Now](#)) - Screening Groups: [Columns](#)
 - Progress Monitoring Groups: Enabled ([Disable Now](#)) - [Settings](#), [Periods](#), [Reports](#)
 - Assessment Analysis Fields: Free Reduced Lunch, Grade, Gender, Ethnic, Home School, Migrant ([Set Up Analysis Fields](#))
 - Student Portfolio Definitions: Graduation Portfolio ([Set Up Portfolio Definitions](#))
 - Curriculum outline editable by: *Users with access privileges*
 - Resource library editable by: *Users with access privileges* ([Change Security Mode](#))
 - Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)
- AnyState Core Curriculum Standards [ASCCS] (In Development)**
 - Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Enabled ([Disable Now](#)) • Legacy "One-Click" Progress Reports: Not Enabled ([Enable Now](#))
 - Progress Monitoring Groups: Enabled ([Disable Now](#)) - [Settings](#)
 - Assessment Analysis Fields: Ethnic, Grade, Gender, Migrant, Hom
 - Student Portfolio Definitions: *None* ([Set Up Portfolio Definitions](#))
 - Curriculum outline editable by: *Users with access privileges*
 - Resource library editable by: *Users with access privileges* ([Change Security Mode](#))
 - Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)

Curricula

- SAMPLE Curriculum [SAMPLE] (Active)**
 - Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Enabled ([Disable Now](#)) • Legacy "One-Click" Progress Reports: Enabled ([Disable Now](#)) - Screening Groups: [Columns](#)
 - Progress Monitoring Groups: Enabled ([Disable Now](#)) - [Settings](#), [Periods](#), [Reports](#)
 - Assessment Analysis Fields: Ethnic, Grade, Gender, Migrant, Home School, Free Reduced Lunch ([Set Up Analysis Fields](#))
 - Student Portfolio Definitions: *None* ([Set Up Portfolio Definitions](#))
 - Curriculum outline editable by: *Users with access privileges*
 - Resource library editable by: *Users with access privileges* ([Change Security Mode](#))
 - Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)

Progress Monitoring Settings

Default Duration Once Every Week Once Every Two Weeks Once Every Four Weeks

Default Probe Frequency Once Every Week Once Every Two Weeks Once Every Four Weeks

Default Alert When?

Other Settings Always Chart Full Assessment Periods Allow CBM Rubric Overrides

FYI – About Progress Monitoring Settings:

Default Duration: This is the default duration for progress monitoring. The default can be changed for individual students.

Default Probe Frequency: This is the default for whether a probe will be administered weekly or bi-weekly. The default can be changed for individual students.

Default Alert When? This determines how many weeks the student must fall below the goal line before a “progress monitoring alert” will be placed on the student. Students with a progress monitoring alert are highlighted for the user. In addition, it is easy to produce a report that lists only students that have a progress monitoring alert.

Other Settings:

Always Chart Full School Year: If enabled, the progress monitoring chart will always show the full school year, so that progress will be shown in the context of the school year. By default, this option is off.

Allow CBM Rubric Overrides: This determines whether end users can override the cutoffs and goal lines established in the assessment definition. By default, this option is off.

Next you must establish the assessment periods for progress monitoring. Assessment periods are spans of dates at the end of which the school district wishes to establish target cutoff scores for progress monitoring. By default, the assessment periods initially correspond exactly to the marking period you have set up. But if your assessment periods do not correspond exactly to the marking periods, you can modify the assessment periods as shown below. Note that when you enter the assessment definitions for progress monitoring, you will be allowed to enter cutoff scores for each assessment period.

The screenshot shows the TIENET Assessment Repository Customization interface. The top navigation bar includes links for Configuration, Settings, Curricula, Integration, Config Tasks, Profile Types, Keyword Tables, Document Templates, and Help. The main content area is titled "Curricula". It displays two curriculum entries:

- SAMPLE Curriculum [SAMPLE] (Active)**: Descriptions include "Student Assessment: Enabled (Disable Now)" and "Legacy "One-Click" Progress Reports: Enabled (Disable Now) - Screening Groups: Columns". A callout box points to the "Groups: Periods" link with the text "Click Progress Monitoring Groups: Periods as shown here."
- AnyState Core Curriculum Standards [ASCCS] (In Development)**: Descriptions include "Student Assessment: Enabled (Disable Now)" and "Legacy "One-Click" Progress Reports: Not Enabled (Enable Now) - Screening Groups: Columns".

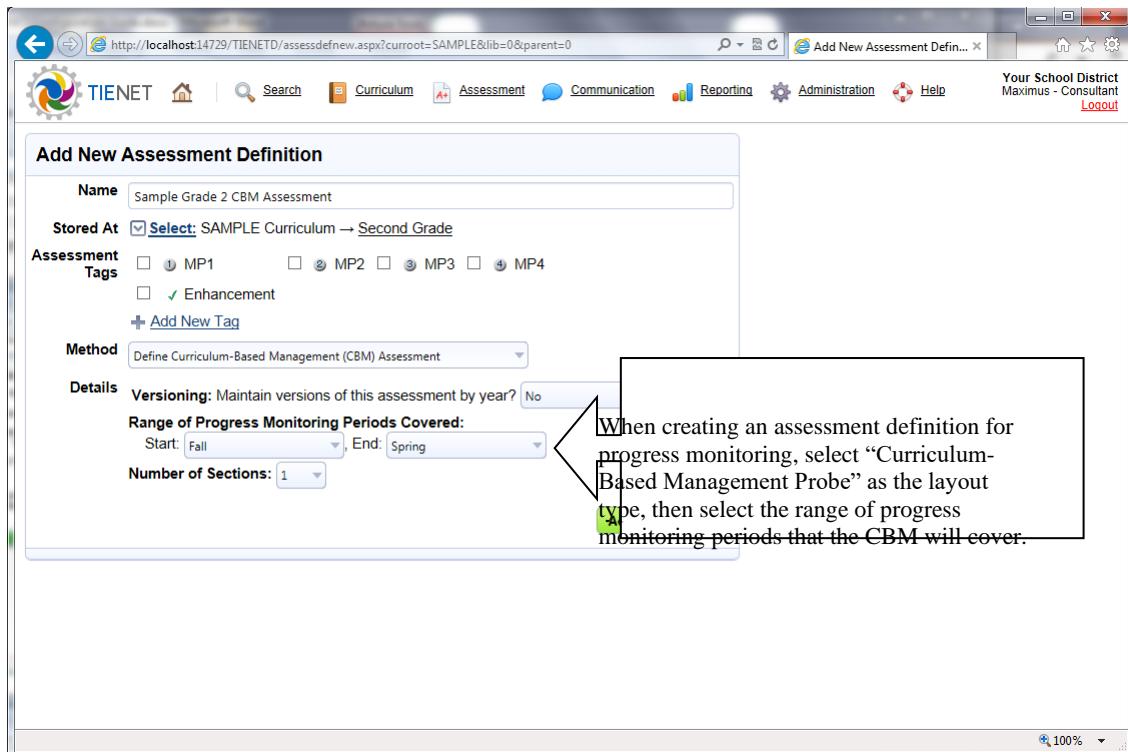
Actions for both entries include Set Up This Curriculum, Set Active Status, Delete This Curriculum, Copy This Curriculum, Download Curriculum, and Validate Assessment Data.

The screenshot shows the TIENET Assessment Repository Customization interface. The top navigation bar includes links for Configuration, Curricula, Assessment, Communication, Reporting, Administration, Help, and Help. The main content area is titled "Progress Monitoring Periods". It shows a table of progress monitoring periods for the "SAMPLE Curriculum [SAMPLE]" for the year 2012-13:

Progress Monitoring Periods (2012-13) for SAMPLE Curriculum [SAMPLE]			
	ID	Starts On...	Ends On...
	Fall	9/4/2012, Start of Q1	1/5/2013, End of Q2
	Winter	1/8/2013, Start of Q3	3/31/2013, End of Q3
	Spring	4/4/2013, Start of Q4	6/3/2013, End of Q4

A callout box points to the table with the text "Initially, the progress monitoring periods correspond exactly to the marking periods in use by the school district. However, you can use this screen to reconfigure the progress monitoring periods to span several marking periods or to begin and end on specific dates."

Create an assessment definition for each Curriculum Based Measurement (CBM) probe you will be using, differentiated by grade level. When creating an assessment definition for progress monitoring, choose “Define CBM Assessment” as the layout type, and then select the range of progress monitoring periods that the CBM will cover. CBM assessment definitions also allow you to enter cutoffs that increase over time to reflect the growth that students are expected to make during the school year.



The screenshot shows a Windows application window titled 'Edit Assessment Definition ...'. The URL in the address bar is <http://localhost:14729/TIENETD/assessdefpropsedit.aspx?assessment=4663&version=1&sec=21647&currout=SAM>. The window contains the following fields:

- Description:** A large text area.
- Data Collected:** CBM Scores
- Range of Assessment Periods Covered - Start:** Fall
- End:** Winter
- Stored at***: Select: SAMPLE Curriculum → Second Grade
- Ownership:** Public
- Current Status***: In Development Note: To administer this assessment, it must be 'Active'.
- Tags:**
 - MP1
 - MP3
 - Enhancement
 - MP2
 - MP4
- Language:** N/A
- Norming Info:**
 - Norm Year: N/A
 - Norm Period: N/A
 - Norm Type: N/A
- Cutoff Scores for Total Scores:** Specify minimum score needed to attain each level.

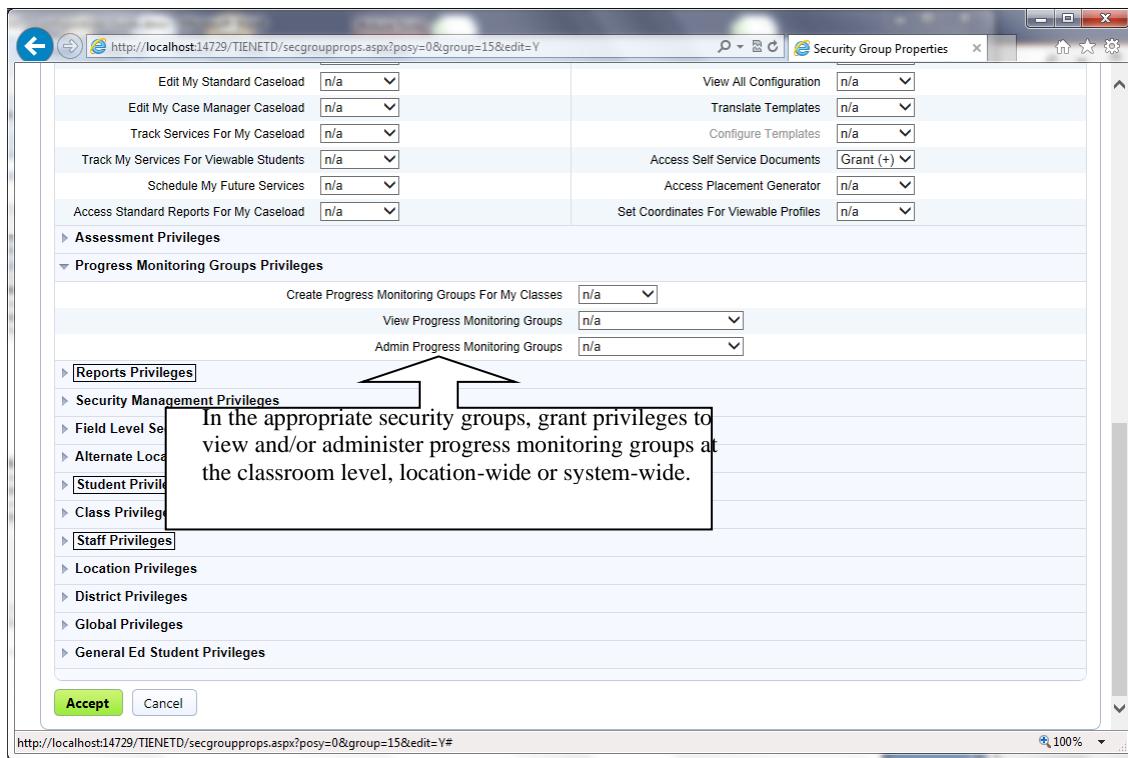
	Start of Fall	End of Fall	End of Winter
Pass/Proficient	55	65	75
Pass/Advanced	65	75	85
- Curriculum Level Assessed:** N/A

A callout box points to the 'Cutoff Scores for Total Scores' section with the following text:

CBM assessment definitions also allow you to enter cutoffs that increase over time to reflect the growth that students are expected to make during the school year.

Buttons at the bottom right: **Accept** (highlighted in green), **Cancel**, and *Required Fields.

For end users to be able to use progress monitoring groups, they must be assigned appropriate security privileges. You can grant privileges to view and/or administer progress monitoring groups at the classroom level, location-wide or system-wide as shown in the figure below.



If you would like to establish a library of interventions for easy selection in progress monitoring groups, then set up a resource type in the resource library named “Interventions” and then populate it with the interventions you want to make available.

The screenshot shows a web-based application window titled "Resource Library Setup". The URL in the address bar is <http://localhost:14729/TIENETD/curreslibrarysetup.aspx>. The top navigation bar includes links for Search, Curriculum, Assessment, Communication, Reporting, Administration, Help, and a "Your School District" dropdown. A "Logout" link is also present.

The main content area displays a table titled "Resource Types for SAMPLE Curriculum". The table has two columns: "Name" and "Properties".

Name	Properties
Teaching Resources	Maintain Publisher Information, Student Access Allowed, Parent Access Allowed 3 Tags 0 Formats
Interventions	n/a 0 Tags 0 Formats
Lesson	Student Access Allowed, Parent Access Allowed 0 Tags 0 Formats
	1 Tag 1 Format

A callout box on the left side of the table contains the following text:

If you would like to establish a library of interventions for easy selection in progress monitoring groups, then set up a resource type in the resource library named “Interventions.”

You can optionally configure one or more ad-hoc list reports that list students who have intervention alerts. Such an ad-hoc list report has to be created a special way that links it to the progress monitoring repository as shown in the screen below.

The screenshot shows the TIENET Assessment Repository Customization interface. The top navigation bar includes links for Configuration, Settings, Curriculum, Assessment, Communication, Reporting, Administration, Help, and Logout. The main content area is titled "Curricula". It displays two curriculum entries:

- SAMPLE Curriculum [SAMPLE] (Active)**
 - Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Enabled ([Disable Now](#)) • Legacy "One-Click" Progress Reports: Enabled ([Disable Now](#)) - Screening Groups: [Columns](#)
 - Progress Monitoring Groups: Enabled ([Disable Now](#)) - [Settings](#), [Periods](#), [Reports](#)
 - Assessment Analysis Fields: Free Reduced Lunch, Grade, Ethnic, Home School, Migrant
 - Student Portfolio Definitions: Graduation Portfolio ([Set Up Portfolio Definitions](#))
 - Curriculum outline editable by: *Users with access privileges*
 - Resource library editable by: *Users with access privileges (Change Security Mode)*
 - Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)
- AnyState Core Curriculum Standards [ASCCS] (In Development)**
 - Student Assessment: Enabled ([Disable Now](#)) • Direct Curriculum Evaluation: Not Enabled ([Enable Now](#)) • Legacy "One-Click" Progress Reports: Not Enabled ([Enable Now](#)) - Screening Groups: [Columns](#)
 - Progress Monitoring Groups: Enabled ([Disable Now](#)) - [Settings](#), [Periods](#), [Reports](#)
 - Assessment Analysis Fields: Ethnic, Grade, Gender, Migrant, Home School, Free Reduced Lunch ([Set Up Analysis Fields](#))
 - Student Portfolio Definitions: None ([Set Up Portfolio Definitions](#))
 - Curriculum outline editable by: *Users with access privileges*
 - Resource library editable by: *Users with access privileges (Change Security Mode)*
 - Actions: [Set Up This Curriculum](#) • [Set Active Status](#) • [Delete This Curriculum](#) • [Copy This Curriculum](#) • [Download Curriculum](#) • [Validate Assessment Data](#)

A callout box points to the "Columns" link under the SAMPLE Curriculum entry, with the following annotation:

To create an ad-hoc list report that reports on progress monitoring data, click Configuration in the blue sidebar, then Curricula, and then Reports as shown here.

The screenshot shows the TIENET Progress Monitoring Reports interface. The top navigation bar includes links for Curriculum, Assessment, Communication, Reporting, Administration, Help, and Logout. The main content area is titled "Progress Monitoring Reports". It displays two sections:

- Class-wide Progress Monitoring Group Reports**
 - New Student Alert Report (for class-wide progress monitoring groups)
 - New Student Listing Report (for class-wide progress monitoring groups)

Report Name
No such reports created
- Location-wide Progress Monitoring Reports**
 - New Student Alert Report (for location-wide progress monitoring groups)
 - New Student Listing Report (for location-wide progress monitoring groups)

Report Name
No such reports created

A callout box points to the "Alert" link under the Class-wide Progress Monitoring Group Reports section, with the following annotation:

Alert reports are used to list students who have fallen below the progress monitoring goal line for a configured length of time. Listing reports simply list students undergoing progress monitoring. These reports can be generated for classroom level progress monitoring or location-wide progress monitoring.

Document Template Customization (Basics)

Getting Started (Tour of Document Templates)

The best way to get started with creating and customizing document templates and their associated workflow is to look at some existing document templates and see how they were done.

As with all other chapters, this chapter assumes the following:

You are familiar with relational databases and HTML.

You are authorized to log in as the CONSULTANT and potentially make changes to the database.

You are thoroughly familiar with all information in the “Special Programs Users Guide” and the “System Administration Guide”.

You are familiar with “Profile Data Schema Customization” chapter of this guide including all the information on fields, data types and keyword tables.

To perform configuration on document templates, you must first create and select configuration tasks as described in the “Configuration Tasks” chapter. Configuration tasks allow the configuration work to be effectively logged and audited.

Follow the steps below to examine an existing document template:

Chapter
6

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

Step 1: Access Configuration from the main menu and then click Document Templates.

Template Name	Template ID	Status	# Sections
Individualized Education Program	IEP	Active	1 In Development, 19 Active, 2 Retired
Speech IEP	SpeechIEP	Active	4
Eligibility Determination	Eligible	Active	4
Full Individual Evaluation	FIE	Active	8
Referral	Referral	Active	1 In Development, 7 Active
Pre-referral	PreRef	Active	5

FYI: Generally, there is a document template for each major phase or step in the Special Education or other workflow process. The template specifies the layout, data and business rules for documents that collect information on the corresponding step of the workflow process. Each template has a name, a short unique identifier and a status as shown here. Normally the status is “active”. However, the status is “In Development” if the template is still being developed. The status can also be “Retired” if the template is no longer in use, but must be kept around to maintain access to historical documents.

Step 2: Click the magnifier icon next to the Individualized Education Program template or

Template Name	Template ID	Status	# Sections
Individualized Education Program	IEP	Active	1 In Development, 19 Active, 2 Retired
Speech IEP	SpeechIEP	Active	4
Eligibility Determination	Eligible	Active	4
Full Individual Evaluation	FIE	Active	8
Referral	Referral	Active	1 In Development, 7 Active
Pre-referral	PreRef	Active	5

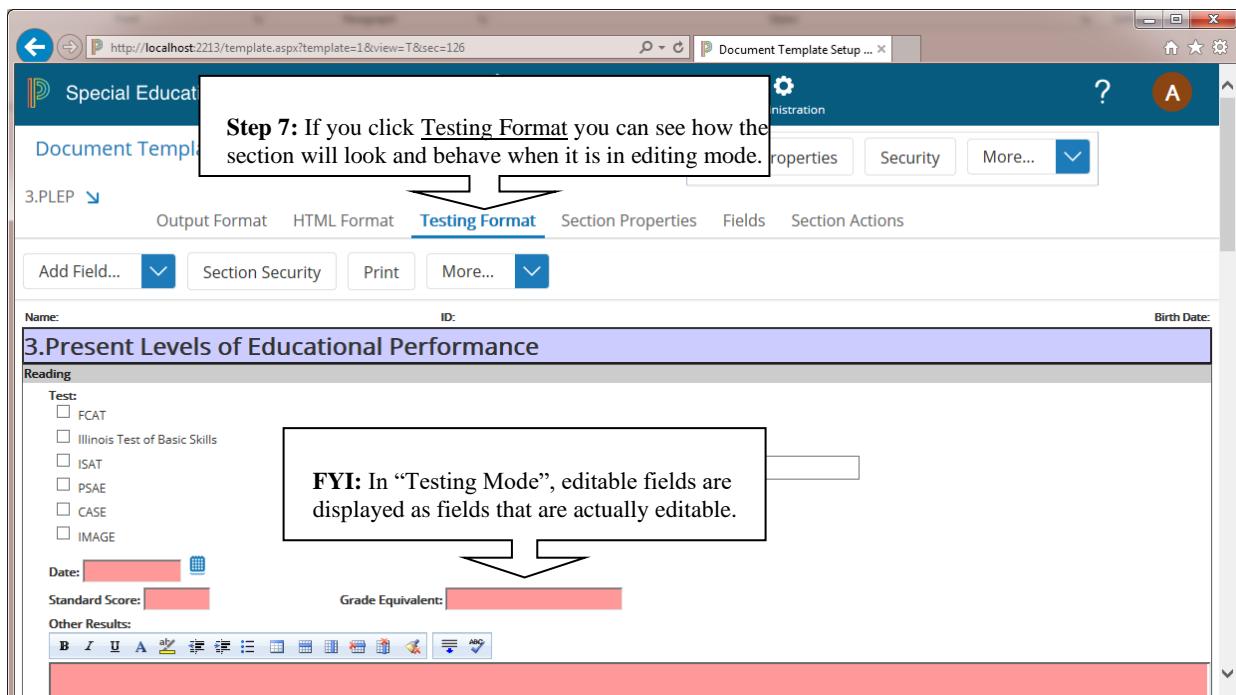
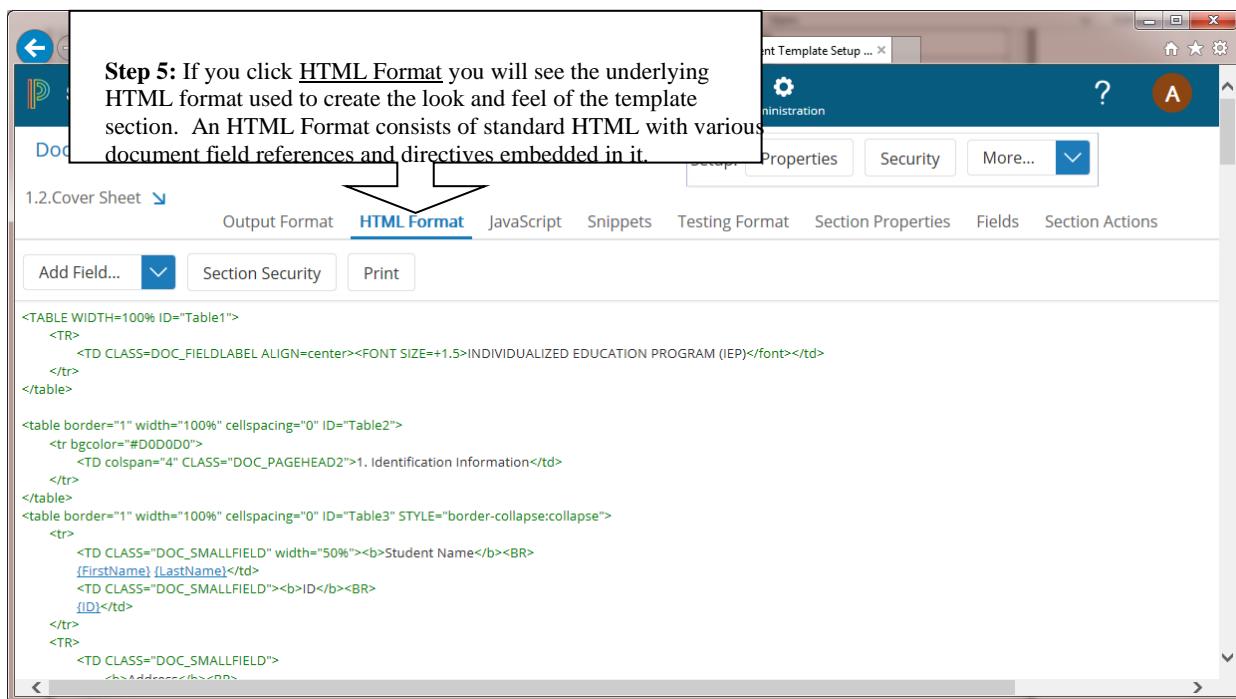
Step 3: The template is organized into sections to facilitate organization when viewing and editing documents and in some cases to approximate the school district's existing set of forms. You will view and work with one section at a time. Use the "select" section dropdown menu to view all the section names and to select one to work with.

The screenshot shows the 'Document Templates > Individualized Education Program [IEP]' screen. A callout points to the 'Sections:' dropdown menu, which lists various document sections such as '1.2.Cover Sheet', '12.Services', '3.PLEP', '13.14.15.Comp Svcs/Transpo', '4.5.Language/FABIP', '16.17.18.Safeguards/EVS', 'FA/BIP', 'IEP Summary', '6.Goals/Objectives', 'Transition Services Plan /in De...', '7.Modifications and Accommodations', 'Test123', '8.Assessment Accommodations', 'Goals and Objectives (Repeating: GOALS)', '9.10.Supplementary Aids/Services', 'IEP Snapshot', '11.Justification of Placement in LRE', 'Child Templates', 'Goals and Objectives [GOALS]', 'Other Sections', 'Conference Recommendations', 'SummaryExtract', 'Progress Report (Repeating: GOALS)', and 'Other'.

Step 4: By default, you will be looking at the "Output Format" tab which gives a sense of how the form will look when viewed (not edited). The form display below also includes the names and positions of the various document fields and directives.

The screenshot shows the 'Output Format' tab selected. A callout points to the 'Output Format' tab and the preview area. The preview area displays the 'INDIVIDUALIZED EDUCATION PROGRAM (IEP)' form with the '1. Identification Information' section highlighted. A callout points to a field name 'ID' (with the directive '{ID}') in the preview, which is linked back to the 'Sections' dropdown in the previous screenshot.

FYI: The document field names are shown as links. If you click such a link, you will access a list of fields (shown in a later illustration) that is automatically scrolled to the field you clicked.



DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

Step 8: If you click Fields, you will see a list of all profile (database) fields linked to this particular form section.

Document Field Name	Data Type (Length)	Data Flow / Default Text
BirthDate	Date	←BirthDate
CLEP_CB_1	Logical Value (prevent empty values)	
CLEP_CB_2	Logical Value (prevent empty values)	
CLEP_CB_3	Logical Value (prevent empty values)	
CLEP_CB_4	Logical Value (prevent empty values)	

FYI: Each field has a name, a data type and various other properties. In addition to the data types described in the “Understanding the Data Dictionary” section on page 63, you may also see “long text” fields. Long text fields only exist in document templates and allow the user to enter a virtually unlimited amount of text into them.

Step 9: If you click Section Actions, you will see a list of all section actions linked to this particular section.

Action Name	User Message	Condition	Follow-Up Action
Prevent Section Completion	You must indicate the student's spoken language.	Lang_LangUsedStudent IS EMPTY	
After Section Completed	You have answered yes to section 7. Click here to enable the behavioral intervention plan.	Behavior_BehaviorImpedeLearning	Prompt Add and Go to Section > FA/BIP

FYI – About Section Actions: A section action is a process-based business rule linked to a particular section that can check and verify data, inform the user of something, or trigger an action based on information the end user is entering into a document.

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

The screenshot shows the 'Document Templates > Individualized Education Program [IEP]' page. A callout box highlights the 'All Fields' dropdown menu, which lists various document fields like '1.2.Cover Sheet', '12.Services', '3.PLEP', etc. Another callout box points to a table titled 'Other Properties' which shows '1.2.Cover Sheet' linked to multiple sections.

Step 10: If you select “All Fields” in the dropdown menu, you will see a list of all document fields in the document template.

FYI: A document field can be used in (linked to) more than one section. In the “All Fields” display, you can see all the sections that a field is linked to.

The screenshot shows the 'Document Templates > Individualized Education Program [IEP]' page. A callout box highlights the 'Document Actions' dropdown menu, which lists actions like 'Print', 'Individualized Education Program Action', 'Modify Data With...', 'When Doc Triggered', etc. Another callout box points to a table titled 'FYI - About Document Actions' which describes the purpose of document actions.

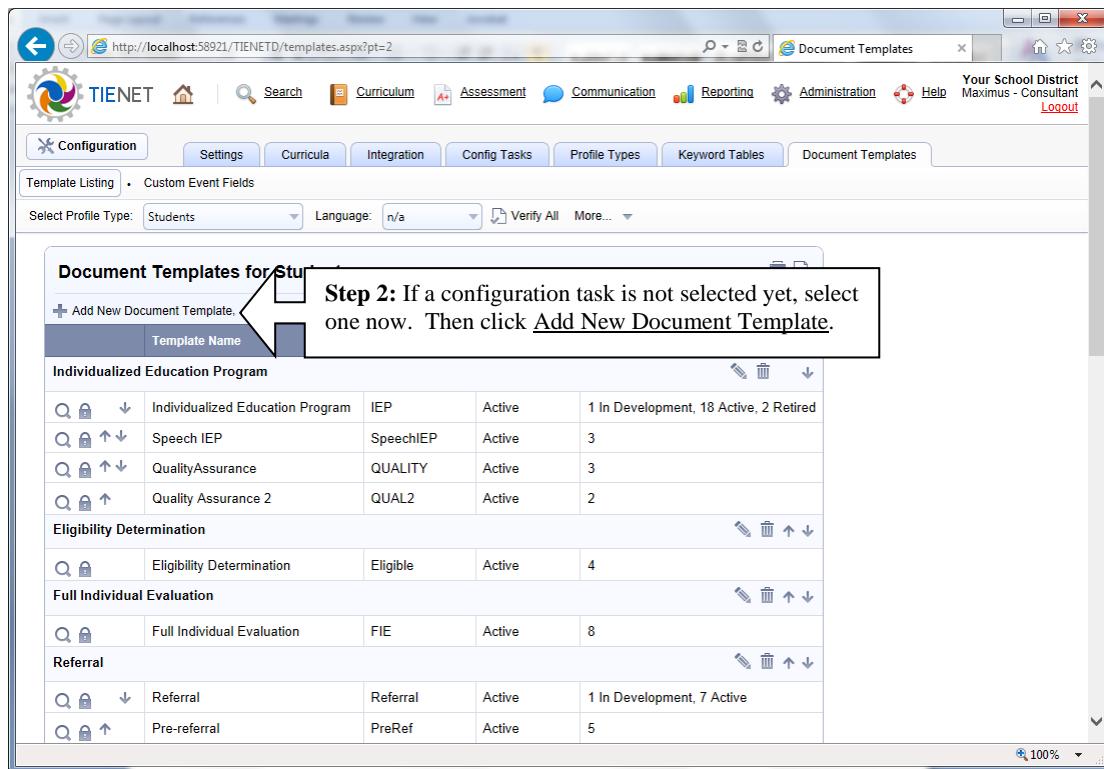
Step 11: If you click Document Actions, you will see a list of all document actions for the document template.

FYI – About Document Actions: A document action is a process-based business rule linked to the entire document that can check and verify data, inform the user of something, or trigger an action based on information the end user is entering into a document.

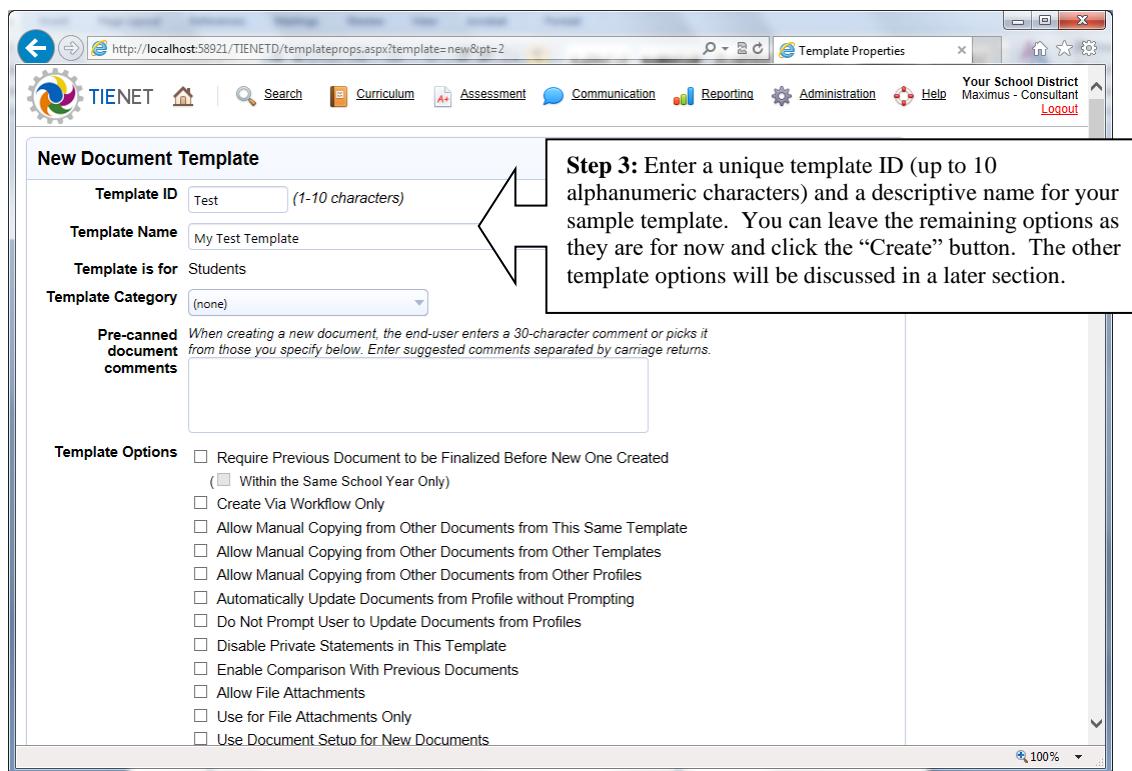
Creating a Document Template

The next step in familiarizing yourself with customizing document templates is to create a sample one of your own so that you can experiment. Follow the steps below:

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)



Step 2: If a configuration task is not selected yet, select one now. Then click Add New Document Template.



Step 3: Enter a unique template ID (up to 10 alphanumeric characters) and a descriptive name for your sample template. You can leave the remaining options as they are for now and click the “Create” button. The other template options will be discussed in a later section.

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

Step 4: Click Fields.

The screenshot shows a web-based application window titled "Document Template Setup ...". At the top, there's a navigation bar with links for Curriculum, Assessment, Communication, Reporting, Administration, Help, and Logout. Below the navigation is a toolbar with tabs: "Output Format", "XHTML Format", "Testing Format", "Section Properties", "Fields", and "Section Actions". A sub-menu for "Fields" is open, showing options like "Edit XHTML Format", "Lock Section", "Edit JavaScript", "Add Field...", "Section Security", "Print", and "More...". The main content area displays a table of fields with their corresponding database column names in parentheses:

ID:	(ID:L)
Last Name:	(LastName:L)
First Name:	(FirstName:L)
Middle Name:	(MiddleName:L)
Birth Date:	(BirthDate:L)
Grade:	(Grade:L)
Eligibility Category:	(EligibilityCategory:L)
Gender:	(Gender:L)
Case Manager:	(CaseManager:L)
Primary Location:	(PrimaryLocation:L)
Home School:	(HomeSchool:L)
Logical QS:	(LogicalQS:L)
Diagnosis Code1:	(DiagnosisCode1:L)

A callout box labeled "FYI:" provides information about the initial template setup:

FYI: The template, when initially created, will have a single section (named “Cover Sheet”). For convenience, the template automatically includes any quick search fields from the student profile and creates a simple form layout (that you will likely want to change later).

Step 5: A set of commonly useful fields is automatically created in the new template. You can review this list of fields and delete any that you know won't be needed in the template (by clicking the delete or trashcan icon next to those fields). Some fields are critical for proper functioning, such as the ID, name fields, and the gender field.

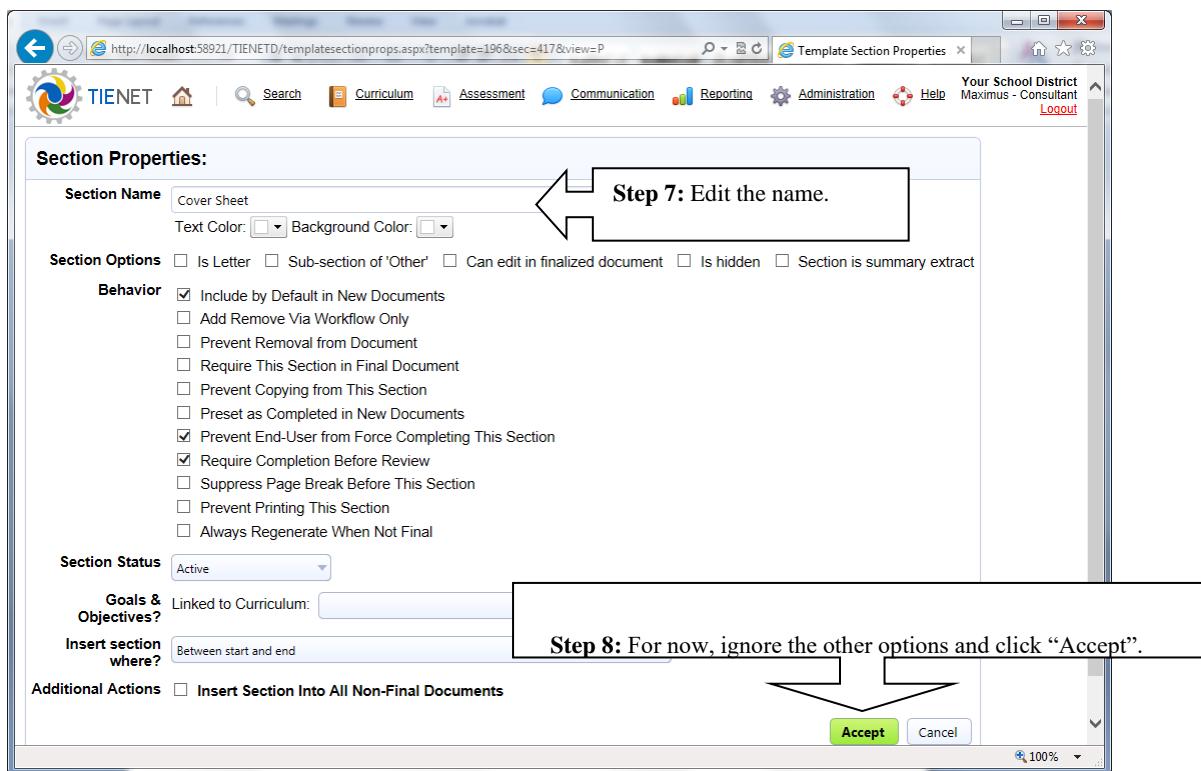
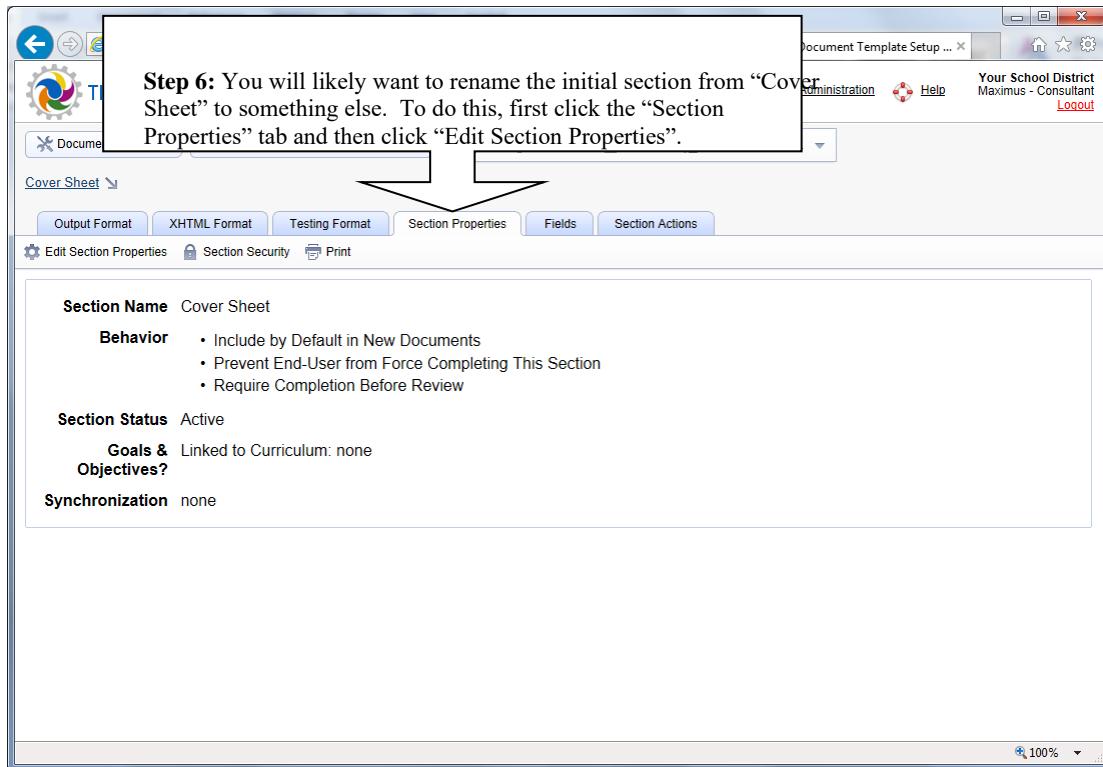
The screenshot shows the same application window with the "Fields" tab selected. A sub-menu for "Fields" is open, showing options like "Add Field...", "Verify Fields", "Set Data", "Data Gathering", "Section Security", and "Print". Below this is a table titled "Document Fields Linked To Cover Sheet".

	Document Field Name	Data Type (Length)	Data Flow / Default	Properties
	BirthDate	Date	← BirthDate	Read Only
	CaseManager	Staff Profile Reference	← CaseManager	Read Only
	DiagnosisCode1	Logical Value (prevent empty values)	← DiagnosisCode1	Read Only
	EligibilityCategory	EligibilityCategories Keyword Selection	← EligibilityCategory	Read Only applicable). Entry comes from the 'EligibilityCategories' Table.
	FirstName	Character(30)	← FirstName	Read Only Contains the student's name.
	Gender	GenderTable Keyword Selection	← Gender	Read Only Contains the student's gender.
	Grade	GradeTable Keyword Selection	← Grade	Read Only Contains the grade level of the student. Entry comes from the 'GradeTable'.
	HomeSchool	Location Profile	← HomeSchool	Read Only Contains the student's home school location (the school the student would attend if not classified). [Social education years - The student's]

A callout box labeled "FYI:" provides information about the data flow:

FYI: This column indicates that all these automatically added fields are “linked” to equivalent fields in the student profile, and will automatically be populated from the student profile when documents from this template are created (a process called data flow). Fields populated from the student profile are typically “read only” (as seen in the column to the right).

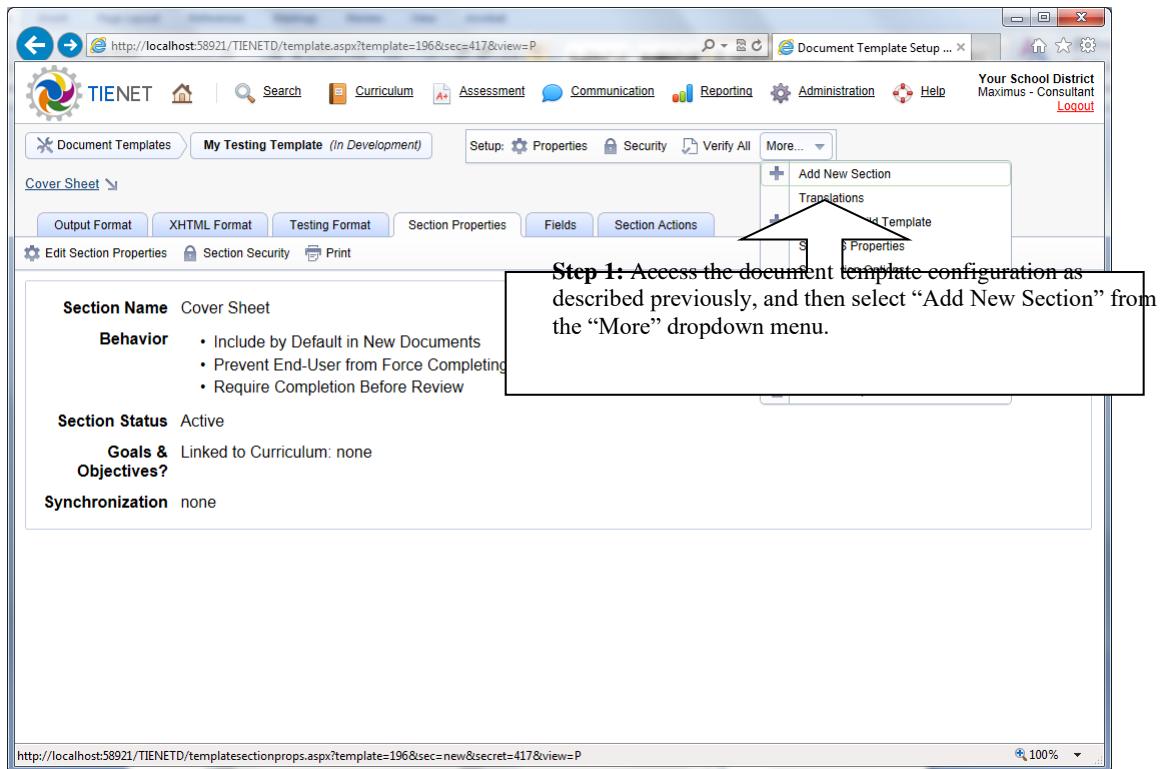
DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

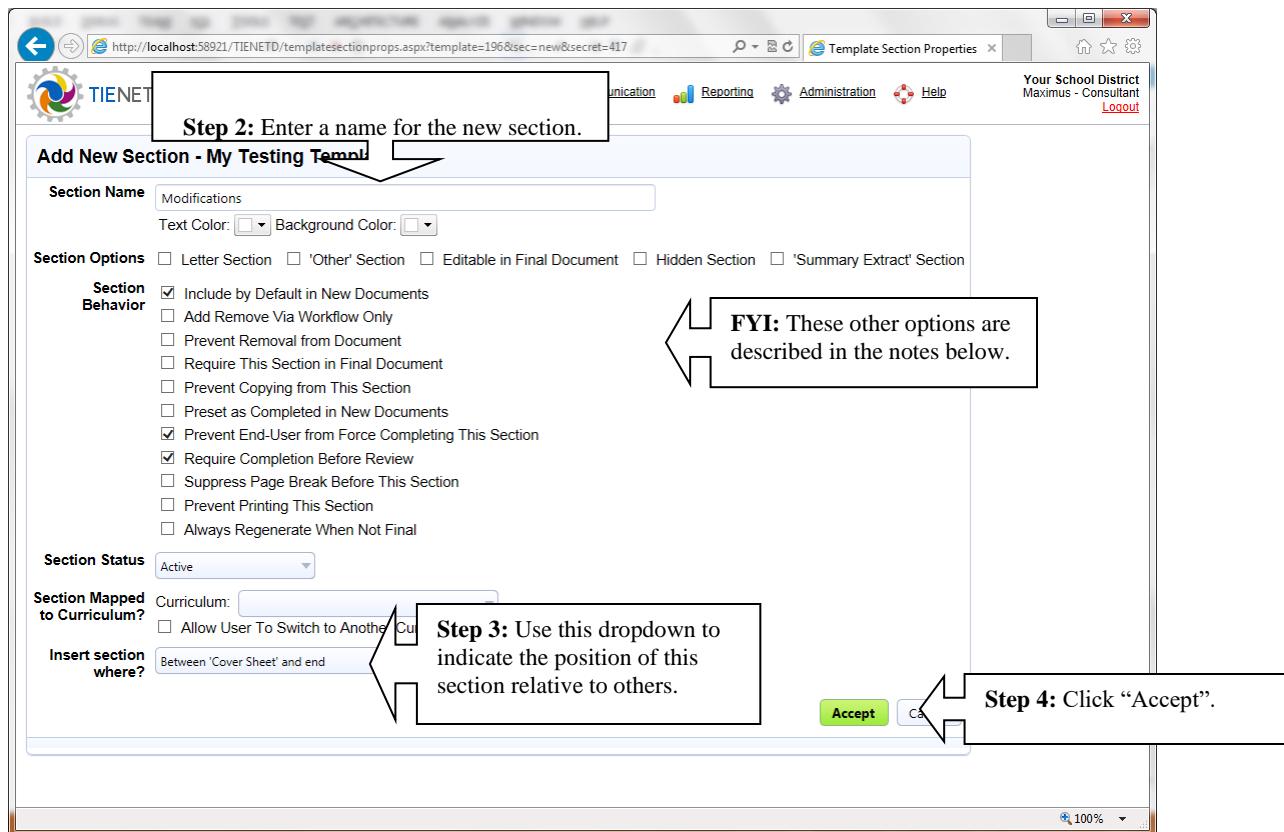


You are now ready to develop this first section fully. The following sections explain how to add and modify template sections.

Adding a New Section to a Document Template

To add a new section to a document template, follow these steps:





The new section is now added. Now you can develop it as described in subsequent sections.

FYI – About the Section Properties:

- **Section Name and Text/Background Color:** The section name is always used to identify the section to the end-user. You can also assign a text and background color to the section. These colors only affect the title/link of the section the end-user clicks to access the section. The colors do not affect the actual content of the section.
- **Section Options:** These are checkbox options as described below:
 - **Letter Section:** If you enable this option, the section will be marked as a “letter” which changes its behavior. Letter sections will not print with the entire document, but are printed individually generally as part of a linear workflow process. Printing a “letter” section will add an event to the student’s event chronology.
 - **‘Other’ Section:** This option is generally used to position ancillary or addendum sections in a separate category labeled ‘Other’ at the end of the document. When the user prints the entire document, ‘other’ sections are excluded by default although it is possible for the user to explicitly select them for printing.

- **Editable in Final Document:** If this option is enabled, the section will remain editable in final documents. This option should be used with care, because it means that the section will not be locked down and have its contents historically preserved if the format of this section changes in the future. Note that if you change this option in a section that is already in use, the change will only be applied to non-final documents. The change will not be retroactively applied to final documents. Also, note that sections with this option are not required to be completed before the document is finalized, whereas sections without this option are required to be completed.
 - **Hidden Section:** This is an advanced option and means that the section will always be hidden in the document. There is a feature that allows the content of a section of one document to be included in another document, and this option can be useful when content from the current document should only appear in the other document.
 - **‘Summary Extract’ Section:** This is an advanced option described in the “Customizing the Documents List View with Summary Extracts” section on page 287.
- **Behavior:**
 - **Include by Default in New Documents:** With this option enabled, the section is included by default in new documents.
 - **Add/Remove via Workflow Only:** This prevents the end user from choosing whether the section is included or not (with the assumption that it will be automatically determined).
 - **Prevent Removal from Document:** Similar to the previous option except that the user can still control when the section is added, but not when the section is removed. This option does not prevent ADMIN/CONSULTANT users from removing the section since there may be extenuating circumstances where this will need to be done.
 - **Require This Section in Final Document:** With this option enabled, a document cannot be finalized unless it includes this section.
 - **Prevent Copying from This Document:** Useful if you want to make sure that a user cannot copy information from this document into a new document. This only affects manual copying by users, and does not affect a “revision” document that utilizes the document revision feature of the system.
 - **Preset as Completed in New Documents:** This option can be enabled for a section that contains little or no editable information. With this option, there will be no requirement for any users to complete the section before the document is finalized.
 - **Prevent End-user from Force-Completing this Section:** With this option enabled, the document cannot be finalized unless the section is fully completed (required fields and compliance checks). If the option is not enabled, the user will receive a warning that the

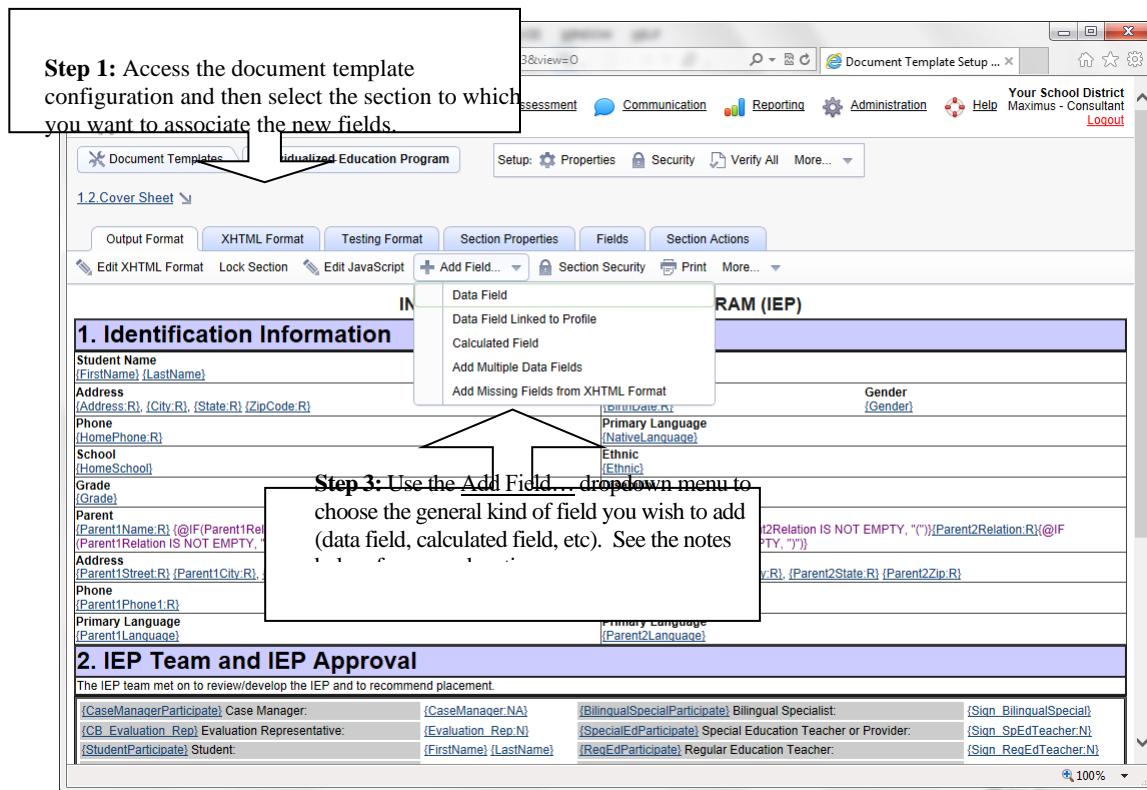
section is not complete, but will have an option to “force complete” it without necessarily meeting all its required fields and compliance checks.

- **Require Completion Before Review:** When this option is enabled, the document cannot be changed from draft to review until this section is completed.
 - **Suppress Page Break Before This Section:** Suppresses the page break that normally appears before this section.
 - **Suppress Water mark for This Section:** This option is available if water marks are enabled (for printing) in the document template. This option allows the water mark to be suppressed for this particular section
 - **Prevent Printing This Section:** The section will not print through normal printing mechanisms. However, keep in mind that a user can always take a screen shot and print that, or might take other extraordinary measures to print the information.
 - **Always Regenerate When Not Final:** Normally, when a user saves a section, the document content is saved in the database and simply redisplayed in the future for system performance reasons. There may be circumstances where the document section includes calculations referencing data outside of the current document that may change over time, and setting this option can prevent “stale” information from appearing to end users. But this option should be set only when needed since it reduces system performance.
 - **Set Incomplete In New Revision Document:** This option is available if revision documents are enabled in the document template. By default, sections in a new revision document are marked as complete since they would be complete in the original final document. Setting this option in a section causes it to be initially marked as incomplete in a new revision document.
 - **Always Omit When Generating PDF:** If set, the section will never be included in any PDF files generated in the web browser or via the Data Connectivity Tool.
 - **Append Signature Images When Printed:** This enables printing of any signatures images (captured with a signature pad) at the end of the section. Signatures made only with a password are not included. Note that multiple sections can be configured with this option, but when a document is printed, the signatures will be printed only once and at the end of the last printed section that has this behavior option. For example, if every section in the document template has this option, then the signature images will always be printed at the end of the document.
- **Section Status:** The template section “status” can be set to one of the following:
 - **In Development:** This means that the section is being developed and should remain hidden from end users other than the ADMIN or CONSULTANT.

- **Active:** This is the normal default status and means that the section should be available to end users as allowed by their access privileges.
- **Retired:** This means that the section is no longer being used and cannot be selected into documents that do not already include it. Note that a section can be deleted entirely without affecting finalized documents. However, retiring the section instead of deleting it can have several advantages. One is that a retired section can continue to exist even in non-final documents. It just cannot be selected into documents that do not already include it. It is especially useful for a goals/objectives section, because if a goals/objectives section is deleted (versus retired), progress can no longer be accessed for it even in final documents.
- **Mapped to Curriculum?** This option is used to enable mapping of data fields in the section to one or more curricula. This is described in detail in the “**Error! Reference source not found.**” section on page **Error! Bookmark not defined..**

Adding Fields to a Template Section

To add a field to a template section, follow the steps below:



FYI – General kinds of fields that you can add:

- **Data Field:** This is a data field that users will complete directly on the document (versus flowing from the profile).
- **Data Field Linked to Profile:** This is a data field that is linked to a field in the profile such that new documents are automatically populated with data from the profile.
- **Calculated Field:** This field is a calculation made using a formula. No actual data is stored in the database for this kind of field.

Step 4a: If you chose adding a data field, enter a name for the field (must be unique within the document template). You can optionally enter a description that will be used for user help text for the new field.

Use the “data type” dropdown menu to specify the type of information the field is to contain. Data types are explained below.

These checkbox properties are described later in this section.

This is an optional field that allows technical notes to be maintained for the field.

FYI – About Data Types: Each field in the data dictionary has a “data type” that identifies the kind of information it stores. The possible data types are described below:

- **Character (Length):** This type of field is used to store text. Each character field allows a single line of text with a maximum number of characters (specified as part of the field definition when the field was created or modified). A character field is typically used to store information such as names, addresses, phone numbers, and ZIP codes. When included in a form, a character field appears as a text box into which users can type.
- **Logical:** This type of field is used to store a yes or a no. You might use this field to show whether a student is low income, or receives a certain service. When included in a form, a logical field appears as a checkbox; however, it is also possible to display it as a dropdown menu.

- **Date:** A field used to store a date. A date field appears on forms as a text box into which users can type, but with a link next to it to bring up a popup calendar.
- **Date/Time:** Similar to date, but stores a time (hours and minutes) as well.
- **Integer:** Stores an integer or whole number value. A minimum and maximum value can optionally be specified in the field definition. By default, an integer field is presented as a text box in edit mode. However, if the integer field has both a minimum and maximum value, it can optionally be presented as a dropdown menu.
- **Numeric:** A field used to store a number that can include decimal places. Be careful not to assume that all numbers should go into Numeric fields. For example, ZIP codes beginning with a zero would be stored without the leading zero; after all 07701 as a number is the same as 7701. Also, if numbers in this field will not have decimal places, the integer data type should be used instead. A numeric field appears on forms as a box into which users can type. Of course, only valid numbers are accepted. A minimum and maximum value can optionally be specified in the field definition
- **Score:** A field used to store a numeric test score. Score fields are similar to Numeric fields, except that PowerSchool Special Programs recognizes them as fields containing test scores.
- **School Year:** School year data fields are designed to hold the identification of a particular school year. School year values are always displayed as a split year (e.g. 2011/2012). When an editable school year field is presented on a form, there will be a default range of years. But it is possible to override the range of years presented with a filter modifier to the field directive.
- **Short Text (Length):** Short text fields are similar to character fields except they allow more than one line of text. As with a character field, the maximum number of characters is specified as part of the field's definition. A short text field appears on forms as a multi-line text box into which users can type.
- **Long Text:** Long text fields allow the end user to enter a virtually unlimited amount of text. A long text field may include an associated bank of commonly-used public statements. In addition, individual users can create “private” statements and submit them for inclusion in the public bank of statements. Long text fields can be configured to allow “stylized” text which allows the user to apply styles such as bold, italic and underline. By default, long text fields are UTF-8 (8 bit) characters suitable for English, Spanish, French, and other European languages. However, individual long text fields can be set to UTF-16 by setting the “Unicode” document field option, and this provides support for non-European languages and phonetic characters.
- **Profile Reference:** This kind of field references a profile of another profile type. For example, the student profile might have a “Staff Reference” field to identify the student’s case manager and a “Location Reference” field to identify the student’s home school. A Profile Reference field appears on forms as a text box with a look-up link next to it. Users can use the look-up link to fill this field, or can type in the ID number if the referenced profile type has an ID field.

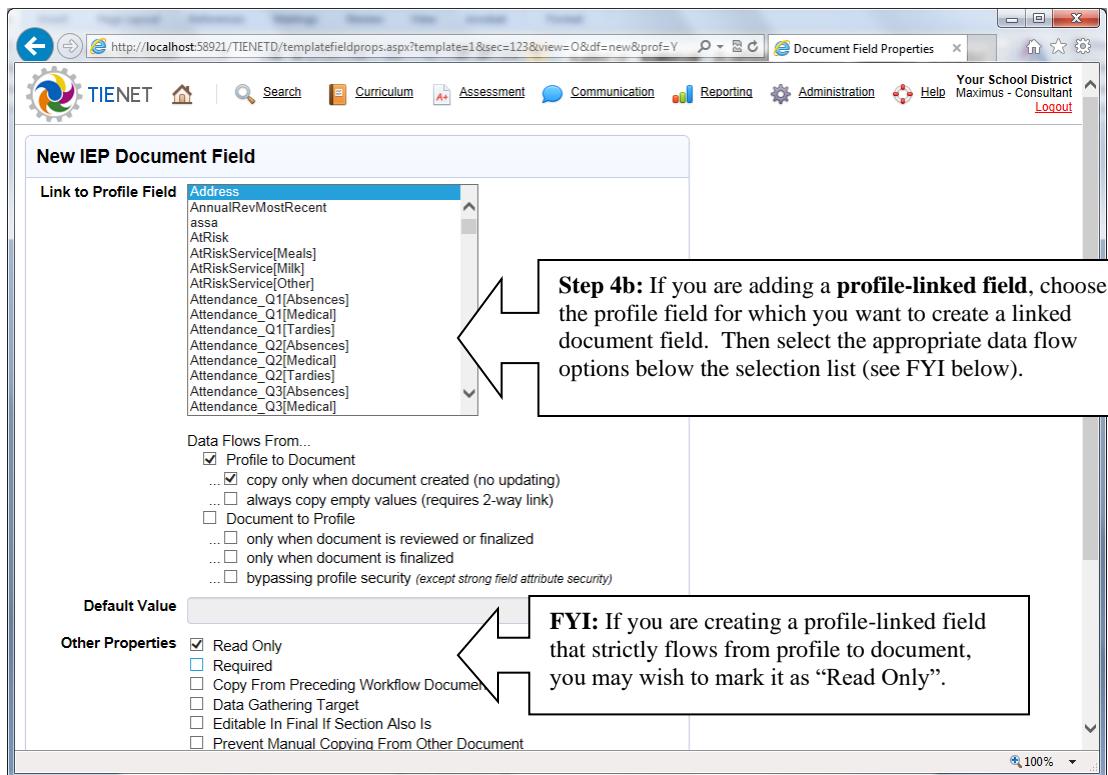
- **Keyword Selection:** A keyword selection field contains a single value selected from a pre-defined list of possible values (e.g. Gender, Ethnic, Grade Level, etc). Keyword selection fields are presented to the user as a dropdown menu, or alternatively as a set of “radio buttons”. The items in the dropdown menu are populated from a “keyword table”.
- **Image:** Stores a web-ready image type (GIF, JPG, or PNG), useful for storing a photo of a person or a logo. When this type of field is on a form in edit mode, the user can select an image file for uploading into the field. Pasting a graphic into the field is also supported in certain browsers: Chrome, Edge, and Firefox. Safari does not support pasting graphic images at the time of writing
- **File:** Stores a file, including the file name and contents. When this type of field is on a form in edit mode, the user can select a file for uploading into the field.

FYI – About Other Properties: On the various screens for adding new data fields, you will notice various yes/no checkboxes labeled under “Other Properties”. An explanation of each is given below:

- **Read-Only:** The document field will always be read-only in the document. This option is only useful if the document field is populated from the profile or from a previous document and should not be edited.
- **Required:** Indicates that the document field is a required field. All required fields on a section must be filled before the section can be marked as complete.
- **Translatable:** This option appears for character, short text and long text fields; and specifies that the field’s values should be translated when translating documents into other languages.
- **Copy from Preceding Workflow Document:** If enabled, the system checks to see if there is a document field with the same name and data type in the previous document in the workflow sequence, and if so automatically copies it into the new document. For more details, refer to the “Document Workflow” section on page 244.
- **Data Gathering Target:** “Data gathering” is a feature that allows a field in a new document to automatically be populated from values in one or more previously created documents. This is described in more detail in the section “Data Gathering” on page 247.
- **No Staff Notification:** Normally when an “event” occurs involving a document, staff referenced in the document are listed for notification. If this option is enabled, the staff member referenced from this field is not included in the list for notification.
- **Editable in Final If Section Also Is:** If the section that includes this field is marked as editable, it is still necessary to use this option to mark individual fields as editable in final as well.
- **Stylized Text:** This option is only available for long text fields. It allows end users to utilize stylized text (bold, italics, tables, etc) in the long text field. Note that the field starts out in plain text mode, and the user clicks an icon to switch the field to stylized text mode unless the next option is selected.

- **Default to Stylized Text:** Also for long text fields only, and requires the Styled Text to also be enabled. If this is selected, the field starts out in stylized text mode and is always in stylized text mode.
- **Allow Graphics in Stylized Text:** Also for long text fields only, and requires the Styled Text to also be enabled. If enabled, the field supports graphics.
- **Auto Save:** Supported for long text fields only. When enabled, as soon as the user leaves the field, its value is immediately saved, accompanied by a visual effect. Note that auto-save does not work in repeating rows, but does work in repeating sections.
- **Include With Quick Search Indexes:** Enables the field's value to be stored in quick search indexes without acting as the main index key. Used for database performance optimization.
- **Allow Signing Document:** If this option is enabled, then the staff person referenced by this field in a particular document can sign that particular document, regardless of whether that user has general signing rights at the document template level.
- **Auto Notify to Sign Document:** If this option is enabled, then when a document is finalized, the staff person referenced by this field will automatically receive a notification message (with a link back to the document) prompting the staff person to sign the document. The messaging will not check if the user actually has rights to sign the document, but enabling the "Allow Signing Document" as well will ensure that the staff person can in fact sign the document.
- **Auto Notify When Document Set To Review:** For staff reference fields only. Ensures that this staff receives a notification when the document is set to review.
- **Auto Notify When Document Set To Final:** For staff reference fields only. Ensures that this staff receives a notification when the document is set to final.
- **Always Available for Macros:** Ensures that the field is available for use in macros within document action user messages in document actions. It is not necessary to set this property for use with section actions since section action macros can always reference any field associated with the section. When applied to character fields, this option allows the field to be used as a macro anywhere in the document including in any statement banks. When editing statement banks, the field macro for the character field will appear at the bottom for easy insertion.
- **Behavior Option:** Setting this option allows the system administrator to preset the field's value in all newly created documents. This is used most often with logical fields that control certain behaviors in the document, and this allows the system administrator to control those behaviors in newly created documents without the need for configuration changes.
 - **Prevent Applying to Existing Documents:** Setting this option prevents the administrator from applying any changes to this field's value to non-finalized documents.

- **Apply When Revising:** Setting this option applies this field's value when revising a document instead of copying the value of the original document.
- **Reset Value If Section Not In Final Document:** If there is an optional section in a document, and you want to configure fields on that section to reset their values if the section is not included when the document is finalized, you can now simply check the "Reset Value If Section Not In Final Document" option in the field properties of those fields. This field option is also settable en masse the same way that required fields are (from the form testing view in configuration). Setting this option appropriately enables the reportable data in the document to align with what is visible in the document. Note that when the document is finalized, a field with this option will only be cleared if it is not associated with any section that is included in the final document.
- **Prevent Manual Copying From Other Document:** If end users are allowed to manually copy information into a document from previous documents, it may be a good idea to prevent copying values for specific fields (especially date fields). This option prevents the field from being copied when the end user manually copies information from other documents.
- **Internal Value Only:** This indicates that the field is used to hold a temporary value or is used in an internal calculation and therefore should not be offered as a field to display on reports. Additionally, fields used in internal calculations may not appear directly on any forms, and so this option also prevents the field from being identified as an unused candidate for deletion.
- **Retired:** Indicates that the field is no longer actively used for new documents, but is being kept in place to retain historical data.
- **No Past Dates in Draft:** Prevents the end-user from entering a date in the past. This is only enforced when the document is in draft mode.
- **No Future Dates:** Prevents the end-user from entering a date in the future.
- **Time Required:** This option is available only for date/time fields and indicate that the time is required (versus the time being optional).
- **Display as Hyperlink:** This option is only available for profile reference fields and causes the field to appear as a hyperlink. When the user clicks the hyperlink, a popup appears showing the profile referenced.
- **Audit Value Changes:** This option defaults to on for all fields but can be turned off for certain fields that are not of high interest to make the audit log less noisy.



FYI – About Data Flow Options: The data flow options control how and when data flows from profile to document and/or document to profile as explained below. Additionally, you will find additional technical details at the end of this section.

- **Profile to Document:** If this option is selected, the field value will flow from profile to document when the document is created or subsequently opened, unless that default behavior is modified via the sub-options below. When this option is selected, the sub-options below appear.
 - **Copy only when document created (no updating):** If this option is selected, the data for this field will flow from profile to document only when the document is first created, and not when the document is subsequently opened.
 - **Always Copy Empty Values:** Normally when the document is updated from the profile, an EMPTY value in the profile will never replace a non-EMPTY value in the document unless the field is ready-only in the document. Even a false value of a logical field that does not support EMPTY values will not override a true value in a document unless the field is read-only. Under certain circumstances this option will become visible for a field that is not read-only: 1) both "profile to document" and "document to profile" data flow is enabled, 2) flow from document to profile is not restricted to when the document is reviewed or finalized. When this option is selected, EMPTY values will flow and overwrite non-EMPTY values in the document even if the field is not read-only.

- **Document to Profile:** If this option is selected, the field value will flow from document to profile as soon as the section containing the field is completed (unless the default behavior is overridden by selecting one of the first two sub-options below). When this option is selected, the sub-options below appear.
 - **Only when document is reviewed or finalized:** If this option is selected, the data for this field will only flow back to the profile when the document is set to review or final mode. Additionally, the data for this field will only flow if the field is associated with one or more sections that are included in the document being set to review or final, or if the field is not associated with any sections at all. If this option and the next are off, the field value will flow as soon as the section containing the field is completed.
 - **Only when document is finalized:** If this option is selected, the data for this field will only flow back to the profile when the document is finalized. Additionally, the data for this field will only flow if the field is associated with one or more sections that are included in the document being set to final, or if the field is not associated with any sections at all. If this option and the previous option are off, the field value will flow as soon as the section containing the field is completed.
 - **Bypass profile security:** Without this option, the field value will only flow to the profile if the user has access privileges to edit the profile. With this option selected, security is bypassed and the value will flow (unless prevented by strong field attribute security, which cannot be bypassed).

New Calculated IEP Document Field

Document Field Name: Age

Description (Help Text): The student's age

Data Type: Integer

Formula: Student Formula [Select Field](#) [Functions](#) YearDifference(BirthDate,GetDate())

Other Properties: Internal Value Only

Integration Notes:

Buttons: Save, Save & Add Again, Save and Exit, Cancel

Step 4c: If you are adding a **calculated field**, select the data type from the dropdown menu and then specify the formula that calculates the values below.

Cover Sheet

Output Format

Edit HTML Format

ID: [ID_L]	Last Name: [LastName_L]
First Name: [FirstName_L]	Middle Name: [MiddleName_L]
Birth Date: [BirthDate_L]	Grade: [Grade_L]
Eligibility Category: [EligibilityCategory_L]	Gender: [Gender_L]
Case Manager: [CaseManager_L]	Primary Location: [PrimaryLocation_L]
Home School: [HomeSchool_L]	Logical QS: [LogicalQS_L]
Diagnosis Code1: [DiagnosisCode1_L]	

Step 5: Now that the field has been added, it can be added to the "HTML Format" for the section, which defines the appearance and layout for the section. Click the "Output Format" tab and then "Edit HTML Format".

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

The screenshot shows a web-based application window titled "Document Template Setup ...". The URL is <http://localhost:58921/TIENET/template.aspx?template=196&sec=417&view=0&edithtml=Y>. The top navigation bar includes links for Search, Curriculum, Assessment, Communication, Reporting, Administration, Help, and Logout. The main content area is titled "HTML For Cover Sheet" and contains a code editor with the following HTML:

```
<table class=DOC_FIELDGRID border=1 cellspacing=0>{ID:L}
[LastName:L]
[FirstName:L]
[MiddleName:L]
[BirthDate:L]
[Grade:L]
[EligibilityCategory:L]
[Gender:L]
[CaseManager:L]
[PrimaryLocation:L]
[HomeSchool:L]
[LogicalQS:L]
[DiagnosisCode1:L]
[Explanation:L]
</table>
```

A callout box labeled "Step 7" provides instructions for editing the HTML format to include a new long text field:

Step 7: Edit the HTML Format to include the new long text field and then click “Save” to save your changes. The specifics of editing an HTML format is covered in the “HTML Formatting Tutorial” chapter and is outside the scope of this section. However, it is useful to note here that in the directive for including a long text field, you can specify a width and height of the editing text box in characters. In this example, the directive {Explanation:LW50H5} is used. “Explanation” is the name of our example long text field. The “L” modifier specifies automatic rendering of a field label within an HTML table (if this is omitted, no label would be rendered). The W50 and H5 modifiers specify a width and height respectively, in characters, for the edit box. You can also refer to the “HTML Format Directives Reference” for more information on directives.

The screenshot shows the same application window, but the "Testing Format" tab is selected. A callout box labeled "Step 8" indicates that the form now looks in editing mode:

Step 8: Click the “Testing Format” tab to see how the form now looks in editing mode.

The testing format view displays the following fields:

ID:	
Last Name:	
First Name:	
Middle Name:	
Birth Date:	
Grade:	
Eligibility Category:	
Gender:	
Case Manager:	
Primary Location:	
Home School:	
Logical QS:	<input type="checkbox"/>
Diagnosis Code1:	<input type="checkbox"/>
Explanation:	<input type="text"/>

Below the form, there is a toolbar with buttons for Output Format, XHTML Format, Testing Format, Section Properties, Fields, and Section Actions. The "Testing Format" button is highlighted. At the bottom, there is a "Set Fields With Values As:" dropdown with options: Required, Not Required, Stylized Text, and Not Stylized Text.

FYI – Verify Fields: If you have made a lot of field changes to an existing document template or to the data dictionary in general, it is advised that you verify all fields in the affected templates to identify problems or obsolete configuration with calculated fields or with data flow. To do this, simply bring up the list of fields for the template (either by section or all fields). Then click Verify Fields to verify and update all calculated fields and data flow properties.

FYI - When does data flow from the profile to the document? For linked fields, it flows when the document is first created and also when the document is updated. If a document is not finalized and it is opened by a user, PowerSchool Special Programs checks for any discrepancy between the fields in the profile and the document, and if so, prompts the user to update the document. It is also possible to configure the template properties of a template so that non-final documents are always updated without prompting the user.

Data flows from a profile field to the document to which it is linked only if all of the following three conditions are met:

The “Profile to Document” option is enabled in document field properties.

The value in the profile is different than the value in the document. Note that if a date field is linked to a date/time field, only the date portion is considered when determining whether the values are different.

It is appropriate to replace the document field’s value, which is deemed to be true when any of the following conditions are met:

The document field is read only.

The “Always Copy Empty Values (requires 2-way link)” option is enable for the link, which also requires a 2-way link (document-to-profile and profile-to-document).

The value in the profile is not EMPTY. Note that for a logical field which does not explicitly allow EMPTY values, a FALSE value will not copy over replacing a TRUE value. Character and short text fields are considered empty if they do not have any characters entered in them.

FYI - When does data flow from the document back to the profile? It can flow when a document section is completed or when a document is set to review and/or final status depending on the options that have been set for the document field, such as the “Only when document is finalized” option described above.

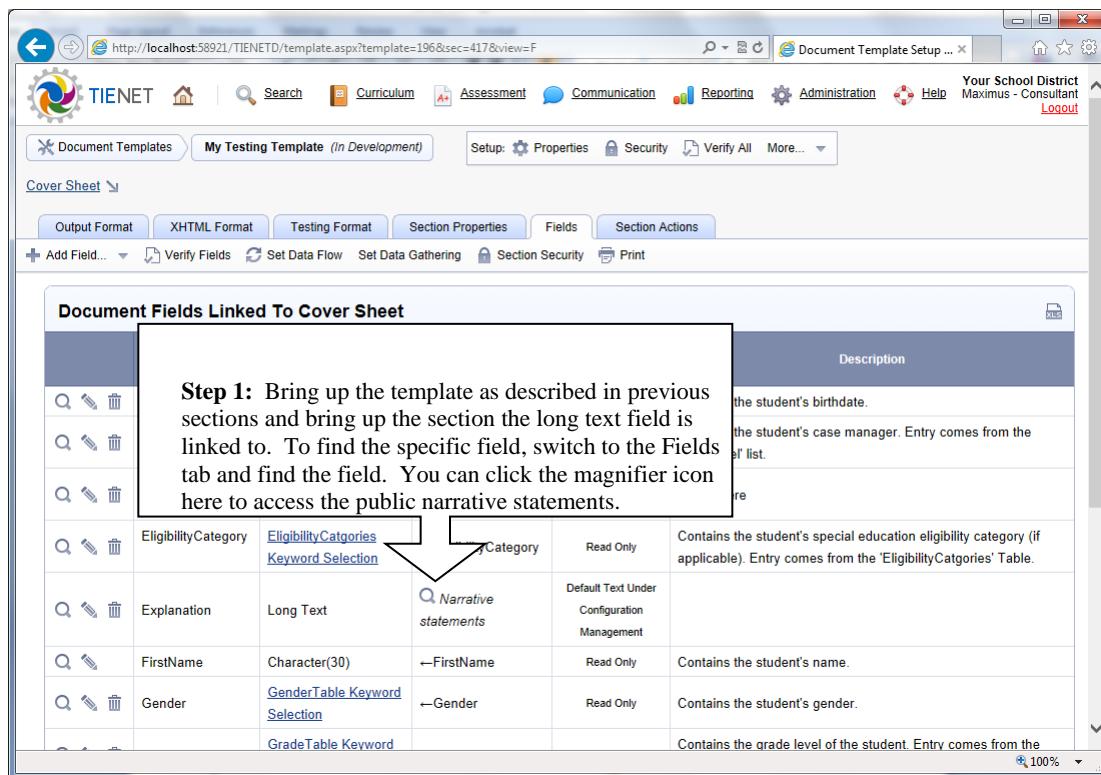
Data flows from the document field back to the profile field to which it is linked only if all of the following three conditions are met:

- The “Document to Profile” option is enabled in document field properties.
- Either the “bypassing profile security” option is enabled, or the user has field level privileges to write to the profile field.

- If the “Only when document is reviewed or finalized”, then data flow will only occur when the document status is being changed to review or final. If the “Only when document is finalized” option is enabled, then data flow will only occur if the document status is being set to final. If none of these options are set, then data flow will occur when the document section is completed and the document field actually changes value during the save.
- If the option is set to flow the data when the document status is set to review and/or final, the data for this field will only flow if the field is associated with one or more sections that are included in the document being set to review and/or final, or if the field is not associated with any sections at all.

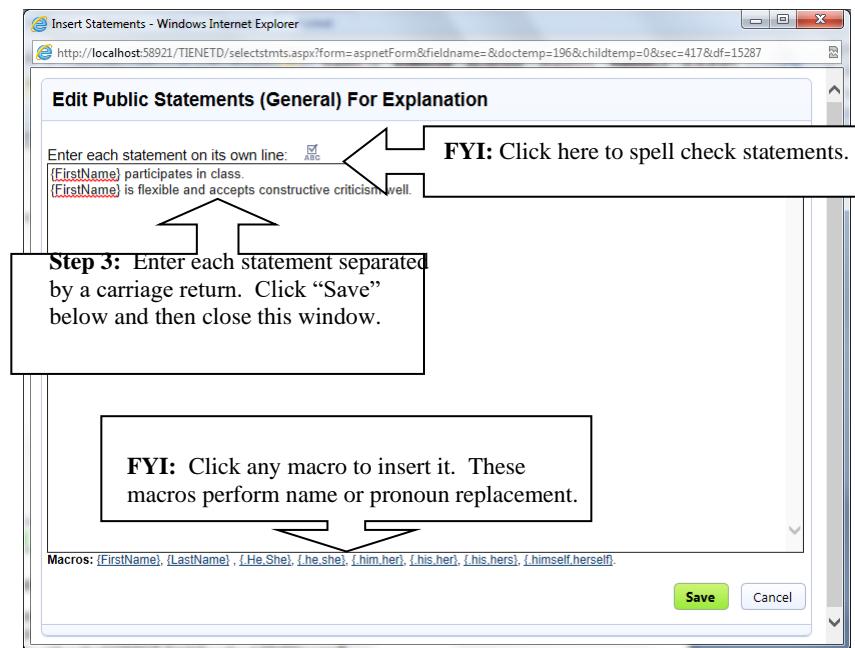
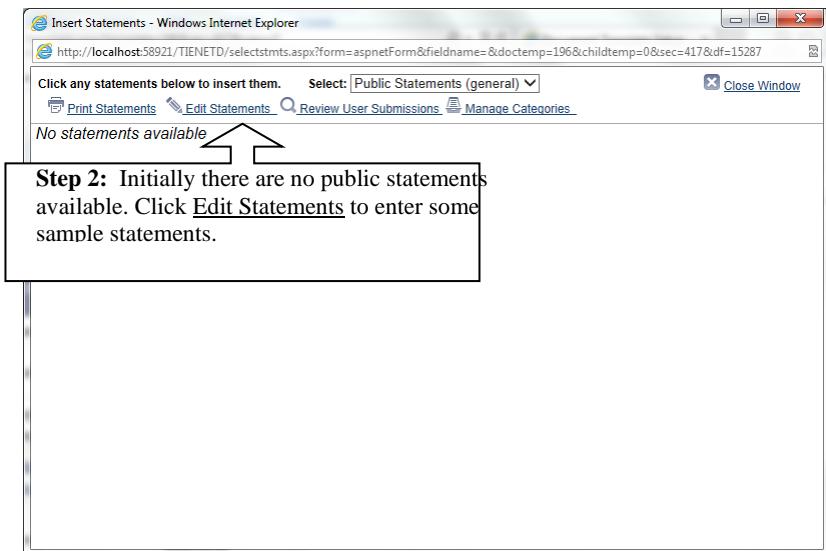
Setting up Public Statements for a Long Text Field

To set up narrative statements for a long text field that end-users can simply insert, follow the steps below:



Step 1: Bring up the template as described in previous sections and bring up the section the long text field is linked to. To find the specific field, switch to the Fields tab and find the field. You can click the magnifier icon here to access the public narrative statements.

	EligibilityCategory	Narrative statements	Description
<input type="button" value="Search"/>	EligibilityCategories Keyword Selection	<input type="button" value="Search"/>	the student's birthdate.
<input type="button" value="Edit"/>	Explanation	Long Text	the student's case manager. Entry comes from the list.
<input type="button" value="Delete"/>	FirstName	Character(30)	the student's first name.
<input type="button" value="Search"/>	Gender	GenderTable Keyword Selection	the student's gender.
<input type="button" value="Edit"/>		GradeTable Keyword	Contains the grade level of the student. Entry comes from the



FYI – The statements popup screen also allows you to create multiple categories of statements which is useful for managing situations where listing them all on one screen would be unwieldy.

Determining whether statements are “under configuration management”: In the field properties, there is an option that allows you to determine whether the narrative statements are “under configuration management” or not (see screen shot below). When under configuration management, the system administrator cannot edit the public statements in the production database since the presumption is that the narrative public statements will be managed in the configuration environment or model database and deployed into production via the CMT.

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

The screenshot shows a web-based application titled "Testing Document Field Properties". The URL is <http://localhost:58921/TIENET/templatefieldprops.aspx?template=196&sec=417&view=F&df=15287&edit=Y>. The page includes a navigation bar with links like Search, Curriculum, Assessment, Communication, Reporting, Administration, Help, and Logout. The main content area displays various configuration settings for a document field:

- Document Field Name:** Explanation
- Description (Help Text):** (Empty text area)
- Data Type:** Long Text
- Default Value:** (Text area containing "ABC")
 - Default Text Under Configuration Management
- Other Properties:**
 - Read Only
 - Required
 - Translatable
 - Copy From Preceding Workflow Document
 - Data Gathering Target
 - Editable In Final If Section Also Is
 - Stylized Text
 - Default To Stylized Text
 - Public Statements Under Configuration Management
 - Prevent Manual Copying From Other Document
 - Internal Value Only
 - Separate Insert Statements with Carriage Returns
- Quick Search Index?**: No
- Synchronization:** Prior fields.name(s) to be synchronized to this one.

A callout box points to the checkbox for "Public Statements Under Configuration Management" with the following text: "This option determined whether the public statements are under configuration management."

Document Template Security Configuration

To establish security access for document templates (IEP, Referral, etc), follow the steps below:

Step 1: Log in as the security administrator and select “Configuration” from the “Administration” menu.

Step 2: Click the “Document Templates” tab.

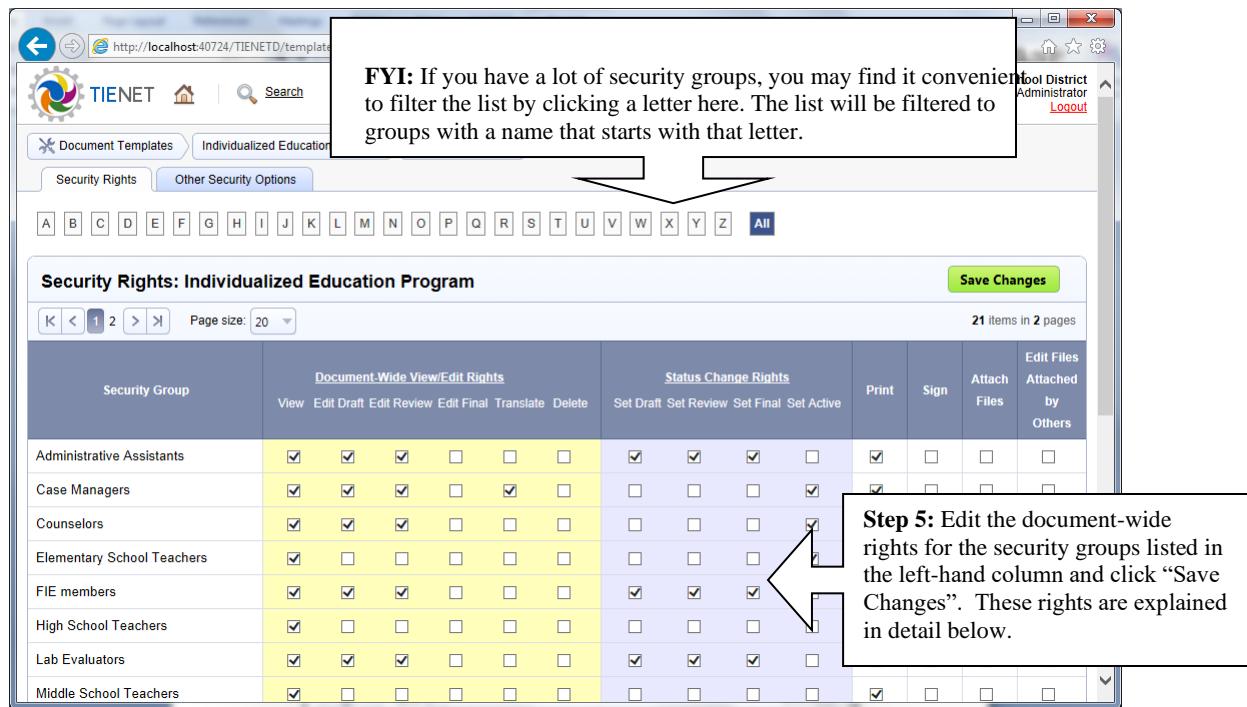
Step 3: Click the magnifier icon next to the document template for which you want to establish security.

	Template Name	Template ID	Status	# Sections
Individualized Education Program				
	Individualized Education Program	IEP	Active	1 In Development, 16 Active, 2 Retired
	Speech IEP	SpeechIEP	Active	3
Eligibility Determination				
	Eligibility Determination	Eligible	Active	4
	FIE	Active	8	
	Referral	Active	1 In Development, 7 Active	
Intervention Documents				
	Student Assistance Team Forms	SAT	Active	11
Other				
	504 Plan	504	Active	3

Step 4: Click “Security” here to establish which groups will have document-wide privileges for documents of this type. Assigning access to individual sections only will be covered later in this same section.

The IEP team met on to review/develop the IEP and to recommend placement.

(CaseManagerParticipate) Case Manager: (CaseManager NA) (BilingualSpecialParticipate) Bilingual Specialist: (Sign_BilingualSpecial)
 http://localhost:40724/TIENET/template.aspx?template=1&sec=123&view=F&hilite=4397#F4397 (edclEdParticipate) Special Education Teacher or Provider: (Sign_SpEdTeacher.N)



FYI: If you have a lot of security groups, you may find it convenient to filter the list by clicking a letter here. The list will be filtered to groups with a name that starts with that letter.

Security Rights: Individualized Education Program

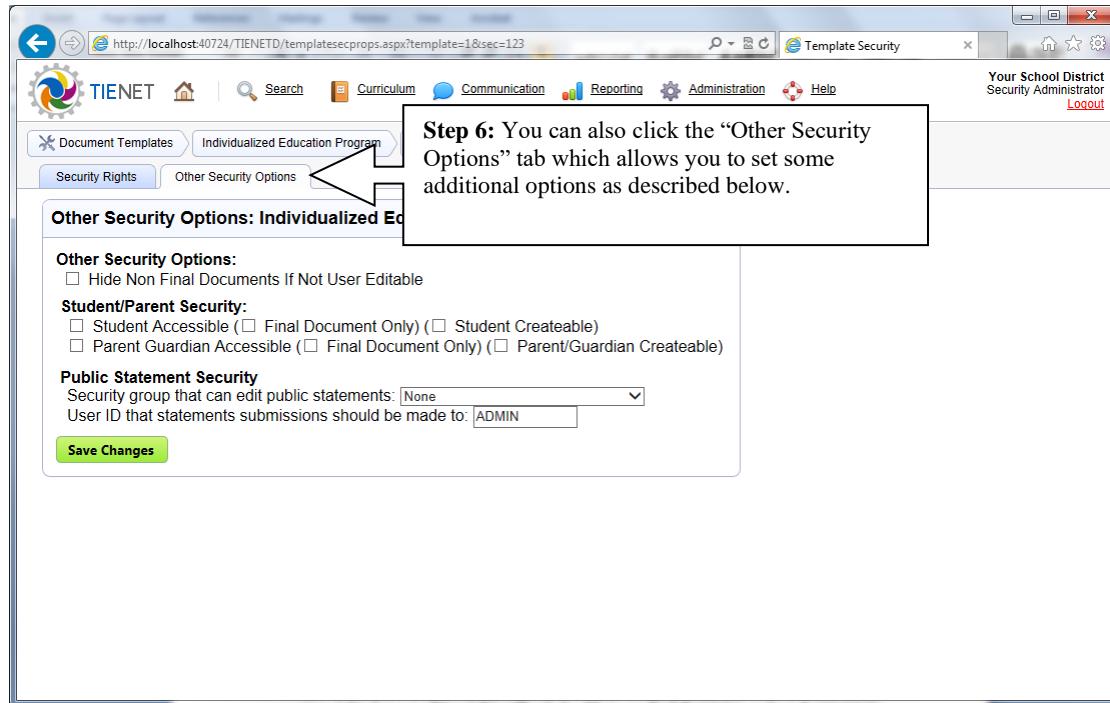
Security Group	Document-Wide View/Edit Rights							Status Change Rights					Print	Sign	Attach Files	Edit Files Attached by Others	
	View	Edit Draft	Edit Review	Edit Final	Translate	Delete	Set Draft	Set Review	Set Final	Set Active							
Administrative Assistants	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Case Managers	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Counselors	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Elementary School Teachers	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FIE members	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
High School Teachers	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Lab Evaluators	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Middle School Teachers	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Step 5: Edit the document-wide rights for the security groups listed in the left-hand column and click “Save Changes”. These rights are explained in detail below.

FYI – About Template Security Rights: These rights, which are applied document-wide, are described below:

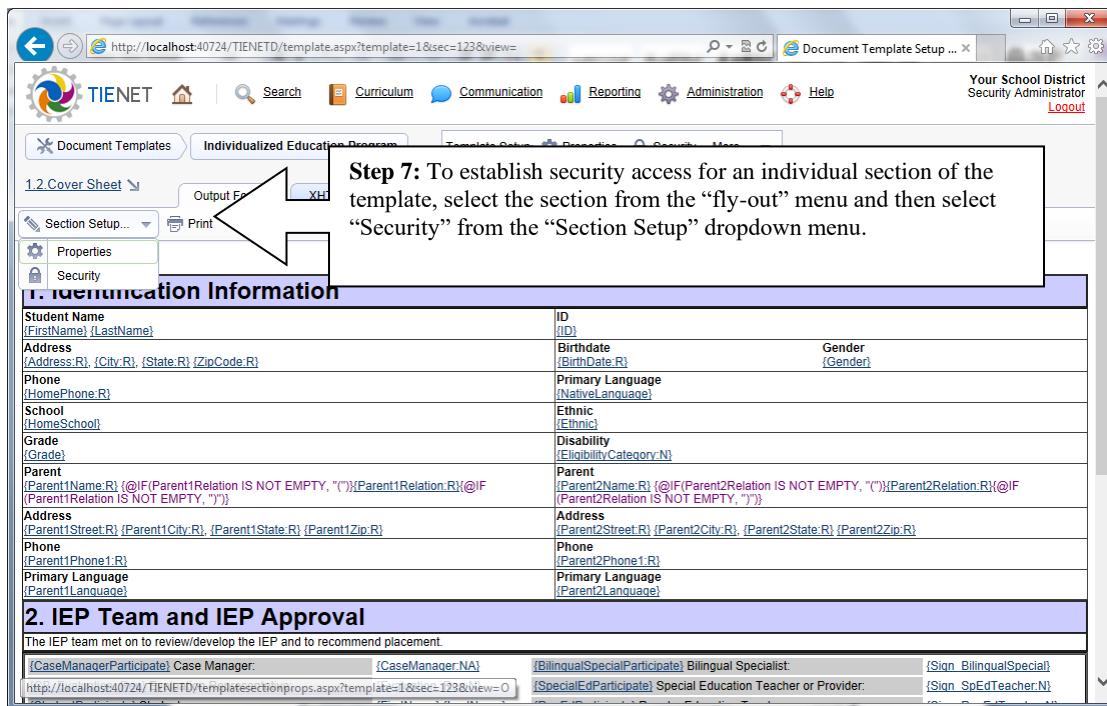
- **View:** Users/Groups with this right can view the entire document (all sections).
- **View If Owner:** This right will only appear for certain document templates that are configured to use the “document owner” security model. If this privilege is available and granted to users, then those users will only be able to access documents that they are the designated owners of. For more information, see the “Document Owner Security” section on page 243.
- **Edit Draft:** User/Groups with this right can edit any section of the document when it is in draft mode. Do not assign this right if edit access should only be granted for individual sections.
- **Edit Review:** User/Groups with this right can edit any section of the document when it is in review mode. Do not assign this right if edit access should only be granted for individual sections.
- **Edit Final:** It should be noted that this right does not actually allow the user to edit finalized documents, which by nature are not editable. In conjunction with other rights described later, it does allow a document to be taken out of final. Only users who should be able to take a document out of final status (normally a highly restricted activity) should be given this right.
- **Translate:** If your school district is utilizing the feature to translate documents into other languages, this assigns the right to translate the document (i.e. edit the translation) but not to change the original (which one would need the above edit rights to do).

- **Set Draft:** Users/Groups with this right can set the status of a document back to Draft as long as they also have the edit right for the status that the document is currently in (i.e. Edit Review or Edit Final).
- **Set Review:** Users/Groups with this right can set the status of a document to Review as long as they also have the edit right for the status that the document is currently in (i.e. Edit Draft or Edit Final).
- **Set Final:** Users/Groups with this right can set the status of a document to Final as long as they also have the edit right for the status that the document is currently in (i.e. Edit Draft or Edit Review).
- **Print:** Users/Groups with this right can edit print the document (the sections they can view).
- **Sign:** Users/Groups with this right can electronically “sign” documents from this template.
- **Attach Files:** Users/Groups with this right can upload and attach files to non-final documents from this template.
- **Edit Files Attached by Others:** Users/Groups with this right can revise or delete files attached by other users.
- **Edit Files Attached to Final Documents:** Users/Groups with this right can attach files to final documents from this template.



FYI – About Other Template Security Options:

- Hide Non Final Documents If Not User Editable:** If this option is enabled, then users who have view rights but not editing rights will only be able to see final documents and not documents in draft or review mode.
- Student/Parent Security:** You can allow students to see their own document from this template, and/or you can allow parents and guardians to see documents from this template for their child. If the “Final Document Only” sub-option is enabled, then students and/or parents/guardians will only be able to view the document if it is final. Note that there is an option to allow the student/parent to even create a document if that makes sense given the type of document, but you also need to mark the relevant individual sections as editable by the student/parent/guardian as explained later in this section.
- Public Statement Security:** By default, only the system administrator can edit the public statements for narrative textboxes in this document template. However, you can designate another security group to allow its members to do this. If you want users to be able to submit their private statements for inclusion in the public statements collection, specify the User ID of the user who will evaluate submitted statements for potential inclusion in the public collection. Designating a user in this field enables the submission feature.

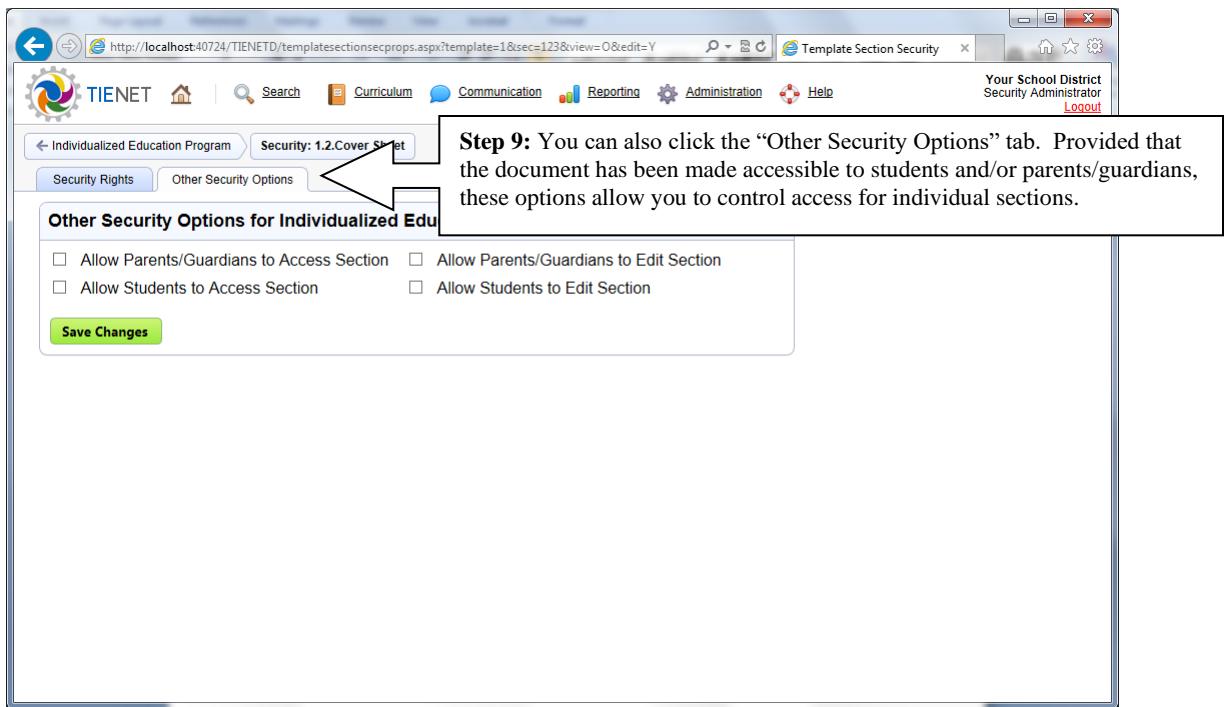


The screenshot shows a web-based application titled "Template Section Security". At the top, there's a navigation bar with links for "Individualized Education Program" and "Security Rights". Below this is a toolbar with letters A through J and a search bar. The main content area is a table titled "Security Rights for Section: 1.2.Cover Sheet". The table has columns for "Security Group", "View", "Edit Draft", "Edit Review", and "Translate". The "Edit Review" column contains several "X" marks, indicating specific permissions. A callout box points to one of these "X" marks with the text: "Step 8: Mark the checkboxes next to the security groups that should have section-wide editing access for Draft and/or Review. Note that any security groups that already have document-wide editing access automatically show a corresponding X." Another callout box at the bottom right of the table area says: "Be sure to click "Save Changes" to save any modifications." A green "Save Changes" button is located in the top right corner of the table area. The table also displays a message: "21 items in 2 pages".

Security Group	Section-Wide Security Rights			
	View	Edit Draft	Edit Review	Translate
Administrative Assistants	X	X	X	<input type="checkbox"/>
Case Managers	X	X	X	X
Counselors	X	X	X	<input type="checkbox"/>
Elementary School Teachers	X	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FIE members	X	X	X	<input type="checkbox"/>
High School Teachers	X	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Lab Evaluators	X	X	X	<input type="checkbox"/>
Middle School Teachers	X	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

FYI – About Section Security Rights:

- **View:** Users/Groups with this right can view this section.
- **Edit Draft:** User/Groups with this right can edit this section when the document is in draft.
- **Edit Review:** User/Groups with this right can edit this section when the document is in review.



Alternate approach to managing template security rights: You can also navigate to a security group and manage the rights for that security group across all document templates. For an illustration of this alternate approach, follow the steps below:

The screenshot shows a web browser window for the TIENET application at the URL <http://localhost:40724/TIENETD/staffsecuritygroups.aspx>. The page title is "Staff Security Groups". The top navigation bar includes links for Assessment, Communication, Reporting, Administration, Help, and a "Logout" button for "Your School District Maximus - Consultant". Below the navigation, there are several tabs: "Security", "Staff Security Groups", "Student/Parent Security", "Sessions", "Audit Log", and "Exception Log". The "Staff Security Groups" tab is active. A callout box labeled "Step 1: Log in as the system administrator and select ‘Security’ from the ‘Administration’ menu." highlights the "Security" tab. Another callout box labeled "Step 2: Click the name of the security group you are interested in" highlights the list of security groups on the right side of the page. The list includes: Administrative Assistants, Case Managers, City School Teachers, Middle School Teachers, Nurses, Occupational Therapists, Physical Therapists, Principals, Provider Supervisors, and Security Administrator (Local). Each group entry includes a small user icon and a descriptive note about who can manage its membership.

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

Step 3: Click the “Document Templates” tab.

Document Templates	View/Edit Rights					Status Change Rights			Set Active	Print	Attach Files	Edit Files Attached by Others	Attach Files to Final Documents	Sign
	View	Edit Draft	Edit Review	Edit Final	Translate	Delete	Set Draft	Set Review						
Individualized Education Program	<input checked="" type="checkbox"/>													
Speech IEP	<input checked="" type="checkbox"/>													
Quality Assurance	<input checked="" type="checkbox"/>													
Eligibility Determination	<input checked="" type="checkbox"/>													
Full Individual Evaluation	<input checked="" type="checkbox"/>													
Referral	<input checked="" type="checkbox"/>													
Pre-referral	<input checked="" type="checkbox"/>													
504 Plan	<input checked="" type="checkbox"/>													
Repeating Demo	<input checked="" type="checkbox"/>													
Repeating Row Demo	<input checked="" type="checkbox"/>													

Legend: ● = Document-wide, ○ = Section-Wide

Step 4: Review the overview grid here. This allows you to see which rights the security group has to which documents. The solid black circles indicate document-wide rights whereas the hollow circles indicate that the security group only has the corresponding right to view certain sections of the document.

Step 5: Use this dropdown menu to select any document template and review/edit the template rights the security group has for that document template.

Step 6: Click Edit here to edit the document-wide rights.

Step 7: Click Edit here to edit the section-wide rights.

Section Name	Document-Wide View/Edit Rights					Status Change Rights			Set Active	Print	Attach Files	Edit Files Attached by Others	Sign
	View	Edit Draft	Edit Review	Edit Final	Translate	Set Draft	Set Review	Set Final					
1.2.Cover Sheet	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3.PLEP	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4.5.Language/FABIP	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
FABIP	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
6.Goals/Objectives	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
7.Modifications and Accommodations	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
8.Assessment Accommodations	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
9.10.Supplementary Aids/Services	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
11.Justification of Placement in LRE	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
12.Services	X	X	X		X				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

These rights can alternatively be assigned for individual sections

User must also have editing right for current document status

Edit Other User's Progress Statements Enter Progress Indicators

Configuring Template and Section Properties

This section covers miscellaneous template and section properties that were seen in previous sections but not fully explained:

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

The screenshot shows the TIENET Document Template Setup interface. At the top, there are navigation links for Search, Curriculum, Assessment, Communication, Reporting, Administration, and Help. A user session is shown as 'Your School District Maximus - Consultant' with a 'Logout' link.

In the center, a large callout box contains the text: "Step 1: Go to the configuration for the document template and then click 'Properties' here".

The main area displays two sections of the IEP template configuration:

1. Identification Information

Student Name {FirstName} {LastName}	ID {ID}
Address {Address.R}, {City.R}, {State.R}, {ZipCode.R}	Birthdate {BirthDate.R}
Phone {HomePhone.R}	Gender {Gender}
School {HomeSchool}	Primary Language {NativeLanguage}
Grade {Grade}	Ethnic {Ethnic}
Parent {Parent1Name.R} {@IF(Parent1Relation IS NOT EMPTY, "({Parent1Relation.R}){@IF(Parent1Relation IS NOT EMPTY, "}))")}	Disability {EligibilityCategory.N}
Address {Parent1Street.R}, {Parent1City.R}, {Parent1State.R}, {Parent1Zip.R}	Parent {Parent2Name.R} {@IF(Parent2Relation IS NOT EMPTY, "({Parent2Relation.R}){@IF(Parent2Relation IS NOT EMPTY, ")))")}
Phone {Parent1Phone1.R}	Address {Parent2Street.R}, {Parent2City.R}, {Parent2State.R}, {Parent2Zip.R}
Primary Language {Parent1Language}	Phone {Parent2Phone1.R}

2. IEP Team and IEP Approval

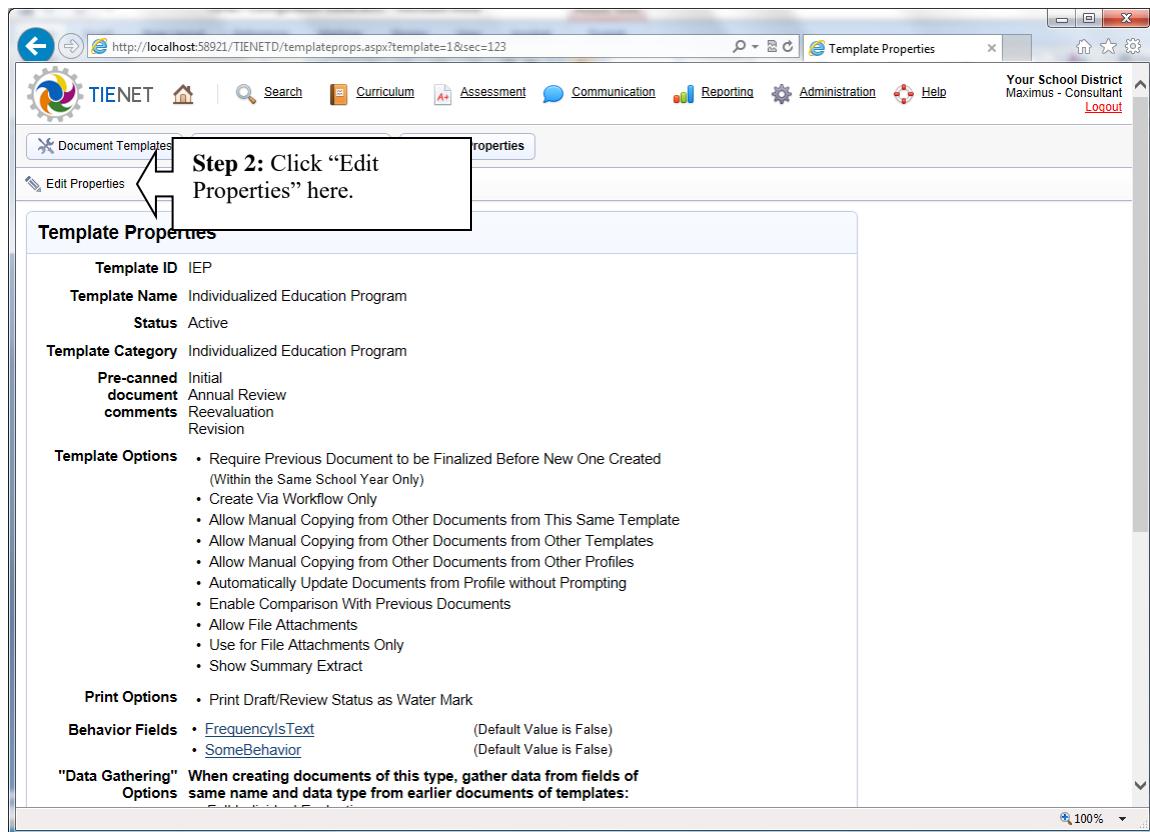
The IEP team met on to review/develop the IEP and to recommend placement.

{CaseManagerParticipate} Case Manager: {CaseManager.NA}	{BilingualSpecialParticipate} Bilingual Specialist: {Sign_BilingualSpecial}
{CB_Evaluation_Rep} Evaluation Representative: {Evaluation_Rep.N}	{SpecialEdParticipate} Special Education Teacher or Provider: {Sign_SpEdTeacher.N}
{StudentParticipate} Student: {FirstName} {LastName}	{RegEdParticipate} Regular Education Teacher: {Sign_RegEdTeacher.N}
{Parent1Participate} Parent/Guardian: {Parent1Name.R}	{Parent2Participate} Parent/Guardian: {Parent2Name.R}
{SurrogateParticipate} Foster Parent/Surrogate Parent: {FosterParent}	{PTParticipate} Physical Therapist: {PhysTherapist.N}
{DistrictRepParticipate} District Representative: {Sign_CST2}	{InterpreterParticipate} Interpreter/Translator: {Sign_SchoolRep}

201

© PowerSchool Group LLC. All Rights Reserved

DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)



The screenshot shows a web browser window for 'Template Properties' at the URL <http://localhost:58921/TIENET/templateprops.aspx?template=1&sec=123>. The page displays various template settings. A callout box with the text 'Step 2: Click "Edit Properties" here.' points to the 'Edit Properties' link located under the 'Document Templates' section.

Template Properties

Template ID: IEP
Template Name: Individualized Education Program
Status: Active
Template Category: Individualized Education Program

Pre-canned document comments: Initial Annual Review, Reevaluation, Revision

Template Options:

- Require Previous Document to be Finalized Before New One Created (Within the Same School Year Only)
- Create Via Workflow Only
- Allow Manual Copying from Other Documents from This Same Template
- Allow Manual Copying from Other Documents from Other Templates
- Allow Manual Copying from Other Documents from Other Profiles
- Automatically Update Documents from Profile without Prompting
- Enable Comparison With Previous Documents
- Allow File Attachments
- Use for File Attachments Only
- Show Summary Extract

Print Options:

- Print Draft/Review Status as Water Mark

Behavior Fields:

- FrequencyIsText (Default Value is False)
- SomeBehavior (Default Value is False)

"Data Gathering" Options: When creating documents of this type, gather data from fields of same name and data type from earlier documents of templates:

The screenshot shows the 'Template Properties' page for a template named 'IEP'. The page includes fields for Template ID (IEP), Template Name (Individualized Education Program), Status (Active), and Template Category (Individualized Education Program). A note about pre-canned document comments is present, listing options like Initial, Annual Review, Reevaluation, and Revision. The 'Template Options' section contains numerous checkboxes for various document management features. A callout box on the right provides instructions for saving changes.

Step 2: There are a variety of options that you can set which are explained in the notes below. If you make any changes that you want to save, click “Accept” at the bottom.

FYI – About Template Properties:

Template ID and Template Name: The template ID must be unique for all the templates and is limited to 10 characters with no punctuation or spaces. In the above example, “IEP” is used for the template ID and “Individualized Education Program” is used for the template name.

Status: Normally the status is “active”. However, the status is “In Development” if the template is still being developed. The status can also be “Retired” if the template is no longer in use, but must be kept around to maintain access to historical documents.

Preset Document Labels/Comments: When users create documents from templates, they can enter a label/comment that describes the specific document they are creating or its purpose. In template properties, you can enter a selection of recommended labels that end users may find useful. Enter the “preset” labels, one per line, separated by carriage returns. These labels will appear to the end user in a dropdown when the end user is creating a new document using this template. However, the end user is not confined to these labels, and can type their own (unless the “Require Document Labels to Include Preset Label” template option is enabled as described below).

Template Options: These yes/no options are described below:

- **Require Previous Document to be Finalized Before New One Created:** If this option is selected, users will be prevented from creating a new document if an existing document that is not finalized already exists for that template. This option has a sub-option labeled “Within the Same School Year Only”, which if enabled, will only prevent a new document from being created if there exists a non-final document from the same template in the same school year.
- **Create Via Workflow Only:** If this option is selected, the document template will not be listed in the dropdown menu for creating a new document. You would enable this option if you want documents from this template to only be created with workflow actions.
- **Allow Manual Copying from Other Documents from This Same Template:** If this option is enabled, when users are working on a document using this template, they can manually copy information from previous documents (for the same student) that were made using this same template.
- **Allow Manual Copying from Other Documents from Other Templates:** If this option is enabled, when users are working on a document using this template, they can manually copy information from previous documents (for the same student) that were made from other templates. The limitation is that only fields with the same name and data type in both templates will be copied, and any fields for which there is data flow from the profile will not be copied.
- **Allow Manual Copying from Other Documents from Other Profiles:** If this option is enabled, then users will be able to copy document content from other documents from other profiles (e.g. student profiles), as long as those documents are from the same template. Currently, this feature is only available to users who have classes or a caseload, in which case the user can copy document content from another student in the class or caseload.
- **Automatically Update Documents from Profile without Prompting:** If this option is set, non-final documents will be updated from the corresponding profile (i.e. data flow) when the document is opened without any prompting from the end user.
- **Do Not Prompt User to Update Documents from Profiles:** This option has no impact if the previous option is set. Otherwise, enabling this option stops PowerSchool Special Programs from automatically checking for data flow updates from the profile and prompting the user. Users can still manually initiate updating however.
- **Allow Statement Banks By Organizational Location:** If this option is enabled, then statement banks can be created and maintained for specific organizational locations. At the current time, only the outer organizational level is supported. For smaller school districts, the outer organizational level is typically “locations”, therefore statement banks associated with specific locations would be enabled by the option. For counties and integrated school districts, the outer organizational level is typically “districts”, therefore statement banks associated with specific districts would be enabled by the option. To authorize a security group to edit statement banks associated with their organizational location and not the system-wide statements bank, assign the “Edit Public Statement Banks” system

administration privilege at the appropriate organizational level (as shown in the first screen snippet below). Then assign the corresponding document template right labeled “Edit Public Statement Banks” for the specific document templates you wish to authorize for that security group (as shown in the second screen snippet below). Note that this template option is overrideable by the system administrator, so it may be preferable to leave the option off in model databases and let the system administrator enable it when needed.

The top screenshot shows the "System Administration Privileges" section of a configuration interface. It lists several options with dropdown menus showing "n/a". The "Edit Public Statement Banks" option is highlighted with a red border and a blue "Grant (+) Location-wide" button below it.

The bottom screenshot shows a grid of document template rights. The columns are labeled "Edit Public Statement Banks", "Force Finalize", and three other columns whose labels are partially visible. The rows represent different document types. The "Edit Public Statement Banks" column for the first row is highlighted with a red border. The "Force Finalize" column for the first row contains three checkboxes: the top one is checked, the middle one is unchecked, and the bottom one is checked. The other two columns have similar patterns of checked and unchecked boxes across the rows.

- Disable Private Statements in This Template:** If this option is enabled, end users will not be able to create private statements for long text fields using this template.
- Enable Comparison with Previous Documents:** If enabled, end users will have an option to compare a document of this template with a previous document of this template, and see specifically what has changed or is different between the two documents.
- Allow File Attachments:** If enabled, end users will be able to upload and attach files to documents from this template.
- Use Document setup for New Documents:** If enabled, when users create a new document of this template, they will be presented with a screen that enables them to choose which sections they want to include in the document. If this is disabled, the default sections are automatically included, and end users go directly into the document.
- Allow Bulk Document Creation:** If enabled, then users authorized to create such documents can go to “Student Search > Utilities > Create Documents in Bulk” to access a utility that allows them to select the document template and a set of student profiles for which to bulk-create documents. Note that if you want to link this to a bulk printing operation, add a date or date/time field named “DocumentPrintedOn” to the document template. PowerSchool Special Programs will automatically populate this field when the document is rendered for printing. Then you can create a

list report of documents that need printing using the formula “DocumentPrintedOn Is Empty”. That report can then be used for bulk-printing.

- **Initiate from Location-Wide Screening Groups:** Enable this option for documents that users need to be able to create from location-wide screening groups. This option is typically used for RtI intervention plan documents.
- **Initiate from Class-Wide Screening Groups:** Enable this option for documents that users need to be able to create from class-wide screening groups. This option is typically used for RtI intervention plan documents.
- **Equip for Delta Export:** If enabled, a SQL index is created that supports good performance when performing “delta” exports of document data.
- **Show Summary Extract:** If enabled, PowerSchool Special Programs will look within each document for a “summary extract” when synthesizing a custom student documents list view. More on this “To Do”.
- **Return to Home Page When Finalized:** If enabled, the user is returned to the user home page when the document is finalized. This is useful if finalizing the document template triggers a guided action that removes the student from active caseloads.
- **Allow Generate Draft Documents as PDF:** If enabled, then users can click “Generate PDF” from the document printing screen of a draft document to download a PDF version of it. See the note below on configuring a PDF page header/footer.
- **Allow Generate Review Documents as PDF:** If enabled, then users can click “Generate PDF” from the document printing screen of a review document to download a PDF version of it. See the note below on configuring a PDF page header/footer.
- **Allow Generate Final Documents as PDF:** If enabled, then users can click “Generate PDF” from the document printing screen of a final document to download a PDF version of it. See the note below on configuring a PDF page header/footer.

PDF Page Header/Footer Format: A PDF page header and/or footer format can now be configured for a document template and is used when documents from the document template are generated in PDF format. To configure this, go to the relevant document template and select “Edit PDF Header/Footer” from the topmost More dropdown. If the menu option is not there, a PDF header/footer may already be defined, in which case you can view it by selecting PDF Header/Footer from the section flyout menu. The PDF header and/or footer format can include directives and supports a ### placeholder for the page number.

- **Do Not Prompt for Document Label:** If enabled, prevents the end-user from being prompted for a document label/comment, which is useful when the document label/comment is being set automatically by a document action.

- **Require Document Labels to Include Preset Label:** If enabled, requires that the end user minimally use one of the labels in the dropdown but with the option of embellishing it with additional text. Note that this requirement is not enforced for revision/amendment documents.
- **Exclude From Transfer Envelopes:** For student document templates only. Excludes the document template from outbound student transfer envelopes. The “Include Live Documents in Transfer Envelopes” includes data to support live documents in outbound student transfer envelopes
- **Include Live Documents In Transfer Envelopes:** For student document templates only. Specifies that data should be included in outbound student transfer envelopes to allow the receiver to create live documents.
- **Compress Long Text Fields: [New in 23.11.0.0]** This option can be a helpful solution when the maximum number of top-level fields in a document template (about 1000) has been reached or is at risk of being reached soon. Note that a warning appears when adding a field and less than 100 additional fields (estimated) can be added until the maximum is reached. When the new option is enabled, any top-level long text fields are compressed into a single database column by the database engine in a way that is largely invisible to the application and to users, although there may be a slight decrease in performance when accessing such fields. With this option enabled, any new or existing top-level long text fields collectively count as 1 towards the limit.

Print Options: These yes/no options are described below:

- **Print Draft Status in Page Header:** If this option is enabled, then draft documents will print with “draft” in the page header.
- **Print Review Status in Page Header:** If this option is enabled, then review documents will print with “review” in the page header.
- **Print Draft Status as Water Mark:** If this option is enabled, then draft documents will print with a “draft” water mark.
- **Print Review Status as Water Mark:** If this option is enabled, then review documents will print with a “review” water mark.
- **Disable Page Breaks Between Sections:** Normally, PowerSchool Special Programs inserts a page break between sections when printing. If you enable this option, page breaks will not be inserted for documents using this template.
- **Prompt for Portrait/Landscape Printing:** Choose these options if you wish PowerSchool Special Programs to prompt the user to set either portrait or landscape printing for documents from this template. If all templates are designed for portrait, it is better to leave this option unchecked.

“Data Gathering” Options: This group of options support a process called “data gathering” which allows field values from one or more previous documents to be incorporated into a new document created from the

template you are working on. Configuring “Data Gathering” is covered in the section “Data Gathering” on page 247.

Event Settings: When a document is set to review or final status, an “event” is generated in the student’s chronology of events. PowerSchool Special Programs will offer a default event description for the event and allow the user to edit that description. These event settings allow you to improve upon PowerSchool Special Programs default event description by specifying default text of your own, both for review status and for final status. In the previous illustration, you can see that text has been set up that uses a macro to personalize the text with the student’s first name, as follows: “IEP Document for {FirstName} has been set to review status. Please proof and set to final.”

Active Documents Enabled: If enabled, this allows PowerSchool Special Programs to track which document of this type is “active” for a profile, while ensuring that only one document from the template can ever be active for a given profile. Active documents are differentiated in the user interface to call attention to them. More details can be found in the “Active Documents” section on page 237.

Progress Monitoring Enabled? These options are only available if the database is licensed for use with the RtI module and are used to equip the document template for use as an RtI intervention plan. See the “RtI Intervention Plan Document Template” section on page 290 for more details.

Revision Documents Enabled: If enabled, the document template is equipped such that an authorized user can make a copy of an original finalized document with just a few clicks, specifically by selecting a “Create Revision of this Document” option. This is useful for implementing a revision or amendment process. The document template is also equipped with a “IsRevision” logical field that is automatically set to true for a document that is a revision document. This field is useful in criteria for guided action and for reporting. . There are several more detailed properties that are available when revision documents are enabled:

- “If Yes, Create Revision Documents In Current School Year” that if enabled, causes revision documents to always be created in the current school year versus the school year of the original document.
- By default, revision documents can only be created from the latest finalized document. To disable this default behavior, enable the “Disable Latest Finalized Document Validation” property.
- In some scenarios, it may be desirable to have a label other than “Revision” appear to end users for this feature. A label textbox is available here to allow a different label, such as “Amendment”, to be configured.

Integrated Meetings Enabled? If enabled, the document template is equipped such that each document can be linked to a group calendar event (meeting) that is created during the process of a user completing the document. When enabling this property, one must specify the name used to refer to this type of meeting (e.g. referral meeting, eligibility meeting, etc.). When this property is enabled, document fields named MeetingDateTime and MeetingLocation are automatically created in the document template as “read only” fields since their values are automatically kept in sync with the group calendar event that gets created for a

document. If it is necessary to constrain when the group calendar item can be scheduled (e.g. within 60 days of a particular date in the document), then an additional date or date/time field named “MeetingDateTimeMax” should be created to identify the maximum date/time. Typically, MeetingDateTimeMax is a calculated date/time field based on another date/time field in the document. A document/section action with a follow up action type of “Schedule Meeting” should be configured to prompt the creation of the group calendar item (meeting) at the appropriate point in the process.

Language Options: The “Allow writing documents directly in language(s) into which template has been translated?” option is relevant if there is an intention to allow certain users to complete documents directly in other languages. PowerSchool Special Programs supports translations of the forms, but this option specifies that the documents themselves will be entered directly in one or more other languages. If this option is enabled, a “LanguageID” character field is added that will automatically contain the two-character ISO code (e.g. es=Spanish, fr=French) for the language that the document was written in.

Data Entry Options > Date Fields -Maximum Days in Future: Specify the maximum days in the future that the user can enter in any date field. This is intended as a document-wide constraint. Additional constraints can be imposed on individual date fields.

Synchronization: If you rename the template code of a document template in a “model” configuration database, enter the old template code in this field. This instructs the Configuration Management Tool to rename the template code (instead of doing a deletion and addition) when synchronizing.

Step 3: To access the properties for a particular section, first select the section from the flyout menu and then click the “Section Properties” tab. Then click “Edit Section Properties”

Section Name	1.2.Cover Sheet
Behavior	<ul style="list-style-type: none"> Include by Default in New Documents Require This Section in Final Document Require Completion Before Review <small>(Configured Value: On, Overridden by ADMIN, Overridden Value: On)</small> <small>(Configured Value: Off, Overridden by ADMIN, Overridden Value: On)</small>
Section Status	Active
Goals & Objectives?	Linked to Curriculum: none
Synchronization	none

Step 4: There are a variety of options that you can set which are explained in the notes below. If you make any changes that you want to save, click “Accept” to save them.

FYI – About the Section Properties:

- **Section Name and Text/Background Color:** The section name is always used to identify the section to the end-user. You can also assign a text and background color to the section. These colors only affect the title/link of the section the end-user clicks to access the section. The colors do not affect the actual content of the section.
- **Section Options:** These are checkbox options as described below:
 - **Letter Section:** If you enable this option, the section will be marked as a “letter” which changes its behavior. Letter sections will not print with the entire document, but are printed individually generally as part of a linear workflow process. Printing a “letter” section will add an event to the student’s event chronology.
 - **‘Other’ Section:** This option is generally used to position ancillary or addendum sections in a separate category labeled ‘Other’ at the end of the document. When the user prints the entire document, ‘other’ sections are excluded by default although it is possible for the user to explicitly select them for printing.
 - **Editable in Final Document:** If this option is enabled, the section will remain editable in final documents. This option should be used with care, because it means that the section

will not be locked down and have its contents historically preserved if the format of this section changes in the future. Note that if you change this option in a section that is already in use, the change will only be applied to non-final documents. The change will not be retroactively applied to final documents. Also, note that sections with this option are not required to be completed before the document is finalized, whereas sections without this option are required to be completed.

- **Hidden Section:** This is an advanced option and means that the section will always be hidden in the document. There is a feature that allows the content of a section of one document to be included in another document, and this option can be useful when content from the current document should only appear in the other document.
- **'Summary Extract' Section:** This is an advanced option described in the "Customizing the Documents List View with Summary Extracts" section on page 287.
- **Behavior:**
 - **Include by Default in New Documents:** With this option enabled, the section is included by default in new documents.
 - **Add/Remove via Workflow Only:** This prevents the end user from choosing whether the section is included or not (with the assumption that it will be automatically determined).
 - **Prevent Removal from Document:** Similar to the previous option except that the user can still control when the section is added, but not when the section is removed. This option does not prevent ADMIN/CONSULTANT users from removing the section since there may be extenuating circumstances where this will need to be done.
 - **Require This Section in Final Document:** With this option enabled, a document cannot be finalized unless it includes this section.
 - **Prevent Copying from This Document:** Useful if you want to make sure that a user cannot copy information from this document into a new document. This only affects manual copying by users, and does not affect a "revision" document that utilizes the document revision feature.
 - **Preset as Completed in New Documents:** This option can be enabled for a section that contains little or no editable information. With this option, there will be no requirement for any users to complete the section before the document is finalized.
 - **Require Completion Before Review:** When this option is enabled, the document cannot be changed from draft to review until this section is completed.
 - **SUPPRESS PAGE BREAK BEFORE THIS SECTION:** Suppresses the page break that normally appears before this section.

- **Suppress Water mark for This Section:** This option is available if water marks are enabled (for printing) in the document template. This option allows the water mark to be suppressed for this particular section
 - **Prevent Printing This Section:** The section will not print through normal printing mechanisms. However, keep in mind that a user can always take a screen shot and print that, or might take other extraordinary measures to print the information.
 - **Always Regenerate When Not Final:** Normally, when a user saves a section, the document content is saved in the database and simply redisplayed in the future for system performance reasons. There may be circumstances where the document section includes calculations referencing data outside of the current document that may change over time, and setting this option can prevent “stale” information from appearing to end users. But this option should be set only when needed since it reduces system performance.
 - **Set Incomplete In New Revision Document:** This option is available if revision documents are enabled in the document template. By default, sections in a new revision document are marked as complete since they would be complete in the original final document. Setting this option in a section causes it to be initially marked as incomplete in a new revision document.
 - **Always Omit When Generating PDF:** Causes the section to be omitted whenever the document is generated as a PDF.
 - **Append Signature Images When Printed:** When the document is printed, the signatures of any users who have electronically signed the document will be appended at the bottom of the section when this option is enabled. Note that multiple sections can be configured with this option, but when a document is printed, the signatures will be printed only once and at the end of the last printed section that has this behavior option. For example, if every section in the document template has this option, then the signature images will always be printed at the end of the document. Each signature will appear with three elements, the actual image of the user’s signature if available, the printed name, and the printed signature date. Text styling can be applied to the printed name and date by defining a CSS class named DOCUMENTSIGNATURES in the CSS style sheet. The image of the user’s signature can come from two sources: 1) image captured via Topaz signature pad, or 2) a staff image field named either ‘SignatureImage’ or ‘Signature’. If no image is available from any of these sources, the printed name and date will still appear.
- **Section Status:** The template section “status” can be set to one of the following:
 - **In Development:** This means that the section is being developed and should remain hidden from end users other than the ADMIN or CONSULTANT.
 - **Active:** This is the normal default status and means that the section should be available to end users as allowed by their access privileges.

- **Retired:** This means that the section is no longer being used and cannot be selected into documents that do not already include it. Note that a section can be deleted entirely without affecting finalized documents. However, retiring the section instead of deleting it can have several advantages. One is that a retired section can continue to exist even in non-final documents. It just cannot be selected into documents that do not already include it. It is especially useful for a goals/objectives section, because if a goals/objectives section is deleted (versus retired), progress can no longer be accessed for it even in final documents.
- **Mapped to Curriculum?** This option is used to enable mapping of data fields in the section to one or more curricula. This is described in detail in the “**Error! Reference source not found.**” section on page **Error! Bookmark not defined..**
- **Insert Document Into All Non-Final Documents:** If you completed work on a new section, you can enable this checkbox to insert the section retroactively into all non-final documents.

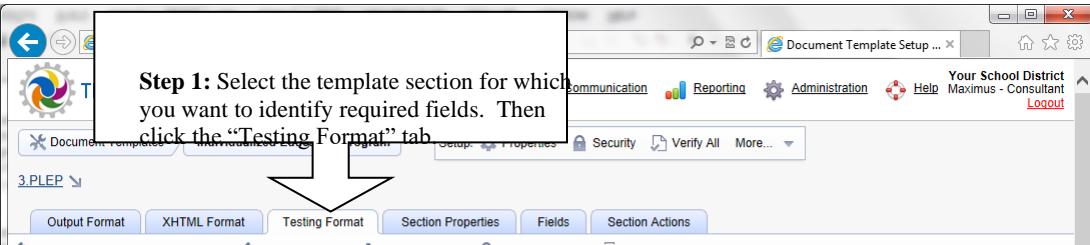
Setting Up Required Document Fields

The section “Adding Fields to a Template Section” on page 180 explained how to individually mark document fields as required. However, the school district may want to mark on a print out of the forms which fields should be required, and so a more efficient way to mark required fields is very helpful.

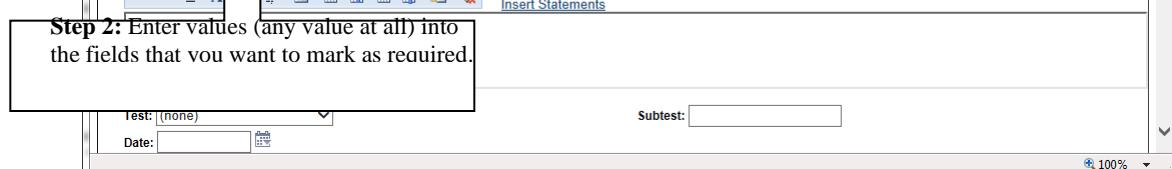
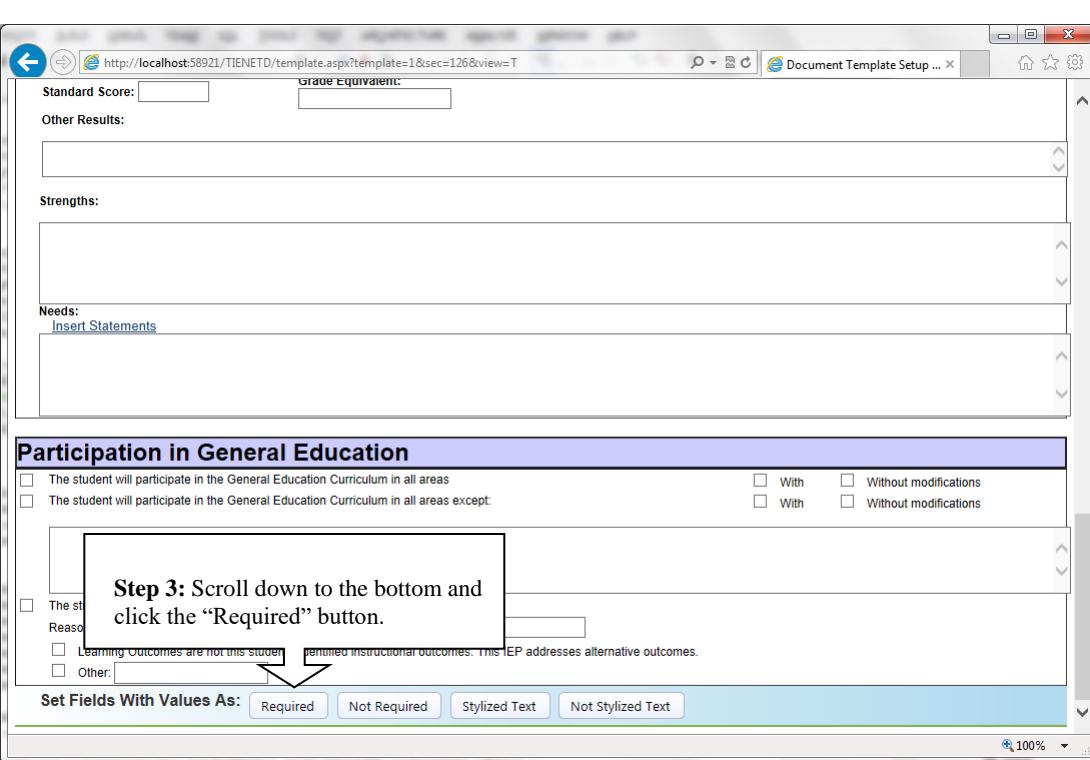
The screenshot shows the 'IEP Document Field Properties' screen in a web browser. The URL is <http://localhost:58921/TIENET/templatefieldprops.aspx?template=1&sec=126&view=F&df=4407&edit=Y>. The page title is 'Document Field Properties'. The 'Document Field Name' is set to 'Reading_Grade_1'. In the 'Other Properties' section, the 'Required' checkbox is checked. A callout box points to this checkbox with the text: 'The document field properties screen allows you to set the "Required" option for document fields on an individual basis.' Other options in the 'Other Properties' section include 'Read Only' (unchecked) and 'Internal Value Only' (unchecked). The 'Synchronization' section contains a dropdown menu with 'No' selected and a text input field for 'Prior fields name(s) to be synchronized to this one:'. The status bar at the bottom right shows '100%'.

To identify all the required fields for a document template section at once, follow the steps below:

Step 1: Select the template section for which you want to identify required fields. Then click the “Testing Format” tab.



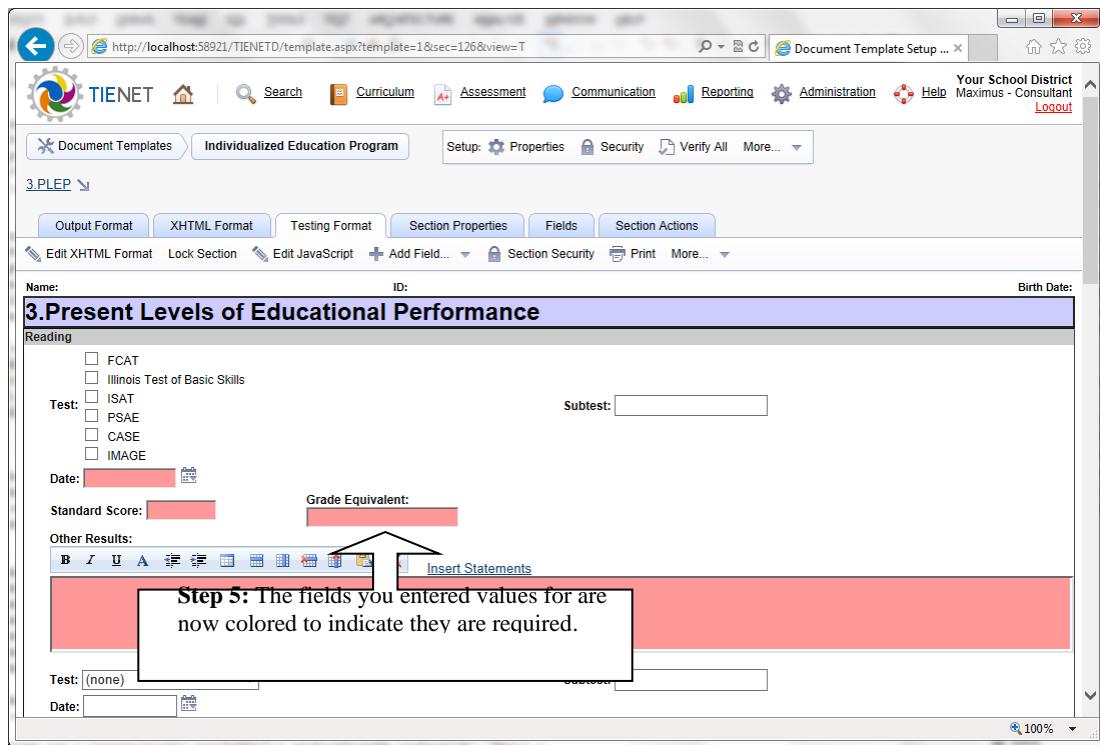
Step 2: Enter values (any value at all) into the fields that you want to mark as required.

Step 3: Scroll down to the bottom and click the “Required” button.

Set Fields With Values As:

- Required
- Not Required
- Stylized Text
- Not Stylized Text



FYI – About Required Logical Fields: A logical field can only be marked as required if it allows an EMPTY value. If a logical field does not allow EMPTY value, it is represented as a simple checkbox and it is meaningless to say a field like this is completed by the user or not completed by the user.

Configuring Section and Document Actions

Section and document actions are added to a template to implement work flow rules and to guide the user. These actions are configured to be invoked conditionally depending on the specific data being entered into a document, and/or upon the security profile or identity of the user entering the information. Actions can be programmed to perform a number of different actions. These actions include preventing a user from entering invalid data, posting an event to the event chronology, including or excluding a form section from the document, providing the user with explanations and/or warnings and much more.

There are two major kinds of actions. Section Actions are linked to a specific template section and are invoked when the user works on that section. Document Actions are linked to the entire template and are invoked during activities that potentially affect the entire document. It is helpful to know that PowerSchool Special Programs processes actions using three steps:

Triggering Event: First, each action responds to a specific type of triggering event (e.g. saving a section, switching a section into edit mode, etc.). When the triggering event for an action occurs, the action is then evaluated.

Evaluation: After the triggering event occurs, the action is evaluated to decide if it should be executed. If the action has a conditional formula, the action will only be executed if the formula evaluates to true. You can also configure a document or section action to trigger only for staff in a particular security group. There is also an option to not trigger the document or section action if a staff member is in a particular security group.

Execution: If the evaluation is positive, the action is executed. In most cases, this results in a message being displayed to the user, but execution can also cause impending activity to be stopped or rolled back, data to be modified, the user to be prompted to take some action, etc.

Section Actions: The triggering events for section actions (actions linked to a specific section) are as follows:

- **Prevent Editing Section:** Section actions of this type are evaluated before the section goes into editing mode. If any actions of this type are executed, the user is prevented from switching into edit mode and an explanatory message is displayed.
- **While Editing Section:** Section actions of this type are evaluated after the section goes into editing mode. If executed, an explanatory message is presented to the user above the editable form.
- **Modify Data Upon Save:** Section actions of this type are evaluated as the section is being saved. The action can be configured to modify data based on the latest field values that the user has just entered. The action can be configured to modify various data using one of the data modification methods described later in this section. This trigger uniquely provides an option to be linked to a button directive in which case the section action is only evaluated if the user explicitly clicks a button configured on the form that references the section action. More information on section action buttons is provided later in this section.
- **Rollback Saving Section:** Section actions of this type are evaluated as the section is being saved and immediately after evaluation of actions of the previous type (Modify Data Upon Save). If any action of this type is executed, the save operation is canceled and no changes are made to the database.
- **After Section Saved:** Section actions of this type are evaluated after the section is successfully saved and can be used to display a message or take some other action (to be covered later).
- **Prevent Section Completion:** Section actions of this type are evaluated after the section is saved and immediate after evaluation of actions of the previous type (After Section Saved). The action can be configured to be evaluated when the user is editing the document, signing the document, or some combination of the two actions. Actions of this type that are configured to run for editing or signing will be ignored when the user is not editing or signing the document. If any action of this type is executed, the section is left as incomplete.
- **Modify Data When Section Completed:** Section actions of this type perform some kind of data modification when a section has been completed. Once a section has been completed, this type of section action will not continue to trigger upon subsequent saves. There may be circumstances where a

section reverts to incomplete (when rules for completion are no longer met due to subsequent editing), and so one should be aware that such section actions may execute more than once over time. This type of section action will not execute when sections are force completed (or documents are force-finalized).

- **After Section Completed:** Section actions of this type are evaluated after the section is successfully marked as complete.
- **Prevent Deleting Repeating Section Copy:** Section actions of this type are used to prevent a user from deleting a repeating section copy based on fields in the repeating section and/or based on the user profile.
- **Modify Data When Prompted:** The action will appear to the end user as a prompt and when the user clicks the prompt, the data modification is performed. The prompt will only appear when the section is complete and not in edit mode, and when the document is not final. The trigger criteria generally should check to make sure the modification has not already occurred.
- **Modify Data Upon Print Section:** This type of guided action is evaluated when there is an attempt to print a section regardless of whether the section is printed individually, or as part of the overall document. The action can be configured to modify various data using one of the data modification methods described later in this section.

Document Actions: The triggering events for document actions (actions linked to the entire template) are as follows:

- **Prevent Creating Document:** Document actions of this type are evaluated before the document even gets created (using a formula based on the profile) and are used to prevent creation of a document based on characteristics of the student profile. If the document template is configured to support revision documents, you can specify if the “Prevent Creating Document” document applies to original documents, revision documents, or both. Note that there is also a “Prevent Creating Revision Document” trigger that only applies to revision documents and uses a formula based on the original document. This type of document actions can also reference DocHistoryYear which will be set to the target history year of the new document (in order to check whether that history year is appropriate).
- **Prevent Creating Revision Document:** Document actions of this type are evaluated before a revision document gets created (using a formula based on the original document) and are used to prevent creation of a revision document based on characteristics of the original document.
- **Modify Data for New Document:** Document actions of this type are evaluated immediately after a new document has been created. The action can be configured to modify various data using one of the data modification methods described later in this section. It is always executed after the new document is populated using data flow or data gathering.

- **Modify Data After Copy from Document:** Document actions of this type are evaluated immediately after data has been copied into the document from another document. The action can be configured to modify various data using one of the data modification methods described later in this section.
- **Modify Data After Updating from Profile:** Document actions of this type are evaluated immediately after data has been copied into the document while updating the document from the profile. The action can be configured to modify various data using one of the data modification methods described later in this section.
- **Modify Data When Document Opened:** Document actions of this type are evaluated whenever a draft or review document is opened and will make a specified modification when trigger criteria are met. Note that when a user creates a document, it is automatically opened and therefore these document actions will also trigger after creation. This trigger type offers a supplementary check box option labeled “Execute for Bulk Printing”. If the option is checked, the document action will trigger prior to rendering the document during bulk printing.
- **Prevent Editing Any Section:** Document actions of this type are evaluated just before any section is edited and can be used to place a custom temporary lock on a document to prevent it from being edited.
- **Any Section Completed:** Document actions of this type are evaluated after any section in the document is successfully completed.
- **All Sections Completed:** Document actions of this type are evaluated after all sections in the document are successfully completed.
- **Prevent Change Status** (Draft, Review, Final): Document actions of this type are evaluated when an authorized user elects to change the status of a document. If any document action with this trigger is executed, the status change is prevented. The document action can be configured to apply only when the new status is final, review, draft, or any combination of these.
- **While Change Status** (Draft, Review, Final): Document actions of this type are evaluated when an authorized user elects to change the status of a document, and the change is not prevented (with a Prevent Change Status). Any messages are displayed while the user is completing the form that finalizes the document. The document action can be configured to apply only when the new status is final, review, draft, or any combination of these.
- **After Change Status** (Draft, Review, Final): Document actions of this type are evaluated after an authorized user successfully changes the status of a document. The document action can be configured to apply only when the new status is final, review, draft or any combination of these.
- **Prevent Deleting Document:** Triggered when the user attempts to delete a document, and if executes, the user is prevented from deleting the document.

- **Bulk Operation:** Use this trigger to allow authorized users to execute this pre-defined data modification action on multiple documents at once through a list report. Once the document action is configured, you can set up a list report that presents the documents for which users should be able to apply this document action. Then go to the report properties of the list report and enable the “Report if under ‘Configuration Management’ option.” Then look for the “Enable Users to Apply Data Modification Constraints via This Report” option (under ‘Output Options’) and enable it too. This second option will only appear if an eligible document action has been created.
- **Modify Data on Repository Attachment Received:** This is a special type of trigger only used when PowerSchool Special Programs is integrated with a document management repository such as Laserfiche®. The document action is triggered when an attachment to the document (such as a wet signature) is received in the repository. The action can be configured to modify various data using one of the data modification methods described later in this section.
- **Modify Data When Scheduled A, B, C, D, E, F:** This type of document action potentially modifies data on the date scheduled by another “scheduling” document action or section action. The modifications occur very early on the scheduled date, well before business hours. The “scheduling” action allows the scheduled date to be specified via a date formula that typically calculates a future date based on a date field in the document. When the scheduled actions are due to be evaluated and potentially triggered, the triggering criteria are evaluated as of the scheduled date, and not the date the scheduling originally occurred. Note that one of the scheduled actions can itself schedule the actions to be evaluated again further in the future. For more information about the possibilities for scheduling, read below about the “Schedule Data Modifications” method. A document can have six independent schedules labeled A, B, C, D, E or F. Data modifications with label A, B, C, D, E or F will be triggered by scheduling actions with the same label (A, B, C, D, E or F).
- **Modify Data when Electronic Signature Status Changes:** This is a special type of trigger only used with DocuSign integration or with our own internal Digital Signature system. The action can be further configured to apply when the signature status is changed to specific values. The document action is triggered when DocuSign or Digital Signature updates a signature request’s status, provided the new status matches one of the action’s configured statuses. The action can be configured to modify various data using one of the data modification methods described later in this section.
- **Modify Data for New Transfer Envelope Live Document:** Document actions of this type are evaluated at the end of the process of initiating a live document from a transfer envelope.

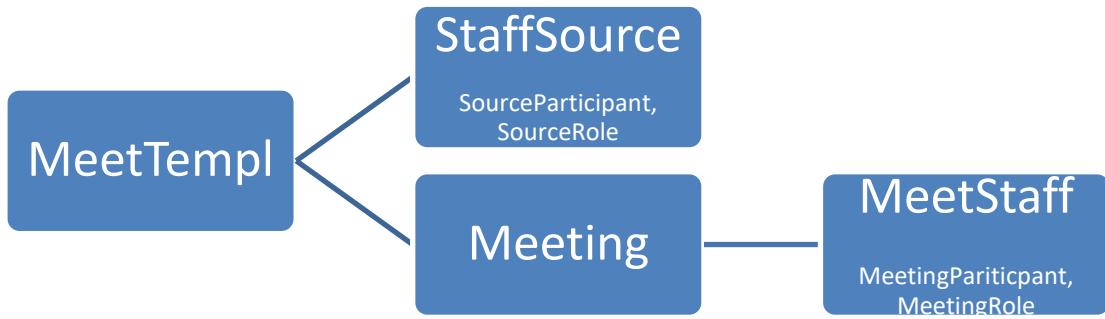
About Data Modification Methods: The trigger types that modify data support the data modification methods listed below. It is important to note that the trigger criteria for all actions with the same trigger type are all evaluated before any modifications are made. Therefore, if there is more than one modification action with the same trigger type, the trigger criteria will only be evaluated based on the state of the document before any of the actions are executed. If the modifications are from a section action that modifies data upon the user saving a section, the trigger criteria will be evaluated after any edits the user made are saved.

- **Modify Document Fields:** Set one or more document field values using the formula language. Note that there are several subtle circumstances in which setting a field using this method is not entirely equivalent to a user manually entering the same value. One is that if the field is set to data flow immediately back to the profile, the data flow does not occur with the action (although it will occur when the document is finalized). The second circumstance, which is esoteric, is that if the field is in a child template and supplies values for a repeating section dropdown menu, the dropdown menu will not be updated. However, in the latter case, a calculated field can supply the dropdown menu with values giving the same flexibility.
- **Set Document Label:** Sets the label/comment of the document. This is especially useful for documents created through workflow where the user is not prompted for a document label/comment. If the document label/comment of a document will always be set automatically, using this type of document action, then it is recommended that the “Do Not Prompt for Document Label” option be enabled in the document template properties.
- **Modify Document Data Via Script:** Supports a data modification script with one or more statements with the same syntax as “CMT Data Transformations” scripts, but the modifications are automatically restricted to the current document. This is intended to support advanced data transformation scenarios within a document that are not covered by other types of document/section actions. See the “CMT Data Transformations” section on page 381 for details on the supported syntax. At this time, this script is limited to modifying data in the current document and cannot be used to modify profile data.

Example: Implementing a document for recording details of ongoing meetings regarding a student.

Background: A student document has a cover page section with editable repeating rows identifying staff participants expected to participate in current and future meetings for a student, along with an additional editable role field for each participant. Each time there is a meeting, an instance of a repeating section is added which is expected to take a snapshot of participants from the cover page and allow additional information to be entered regarding each participant. Earlier instances of the repeating section are required to historically preserve the meeting participants and roles at that time.

As depicted in the diagram below, the document template has ID=MeetTempl. The participants enterable on the cover page section are maintained in a child template with ID=StaffSource that has a staff field named SourceParticipant and a SourceRole field. The repeating section has a top-level child template with ID=Meeting and a grand child template with ID=MeetStaff with a staff field named MeetingParticipant. The structure looks like this:



A section action is needed to synchronize the participants and their roles from the coversheet to the last instance/copy of the Meeting repeating section. After the section action is executed, the participants and roles on the last copy should be the same as those on the cover sheet but without loosing additional field data entered onto the last copy per participant.

Solution: First, you need to prevent duplicate staff from being entered into StaffSource. Next you will need to make it convenient to identify the last copy of the Meeting child template by adding a logical calculated field named LastCopy to Meeting with a formula of “CopyNumber=MAXOF(CHILDA,CopyNumber)”. Then the following three-line script will then sucessful synchronize MeetStaff from StaffSource, but only in the last copy of Meeting.

```

INSERT #MeetTempl#Meeting#MeetStaff (MeetingParticipant, Role) FROM
#MeetTempl#StaffSource (SourceParticipant, Role) WHERE LastCopy=True AND
IFELSE(EXISTS(MeetStaff, SourceParticipant=MeetingParticipant),1,0)=0

UPDATE #MeetTempl#Meeting#MeetStaff FROM #MeetTempl#StaffSource SET
MeetingRole=SourceRole WHERE LastCopy=True AND MeetingParticipant=SourceParticipant

DELETEFROM #MeetTempl#Meeting#MeetStaff WHERE LastCopy=True AND NOT
EXISTS(StaffSource, SourceParticipant=MeetingParticipant)

```

[New in 20.11.1.0] “Modify Document Data Via Script” document/section actions can be used to insert top level profiles from child documents, and if the child template has a profile reference field to the target top-level profile type, you can capture each top level profile inserted back in the child document using the CAPTURE syntax (see capture syntax introduced for profile update scripts). The captured profile reference field can then be used to potentially update the top-level profiles later from the child documents.

```

INSERT top_level_profile_type (target_columns) FROM #doctemplateid#childtemplateid
CAPTURE profile_reference_field (source_columns) WHERE profile_reference_field IS
EMPTY

```

```
UPDATE top_level_profile_type FROM #doctemplateid#childtemplateid SET
targetfield1=sourcefield1, targetfield2=sourcefield2, ... WHERE
THIS=profile_reference_field
```

- **Insert Section:** Adds a particular section to the document. If the section already exists in the document, the action is not executed (and the user message, if specified, is not displayed).
- **Set Section Incomplete:** Ensures that a particular section in the document is marked incomplete. If the section does not exist in the document or is already incomplete, the action is not executed (and the user message, if specified, is not displayed).
- **Delete Section:** Remove a particular section from the document. If the section does not exist in the document, the action is not executed.
- **Modify Profile Fields:** Set one or more field values in the profile using the formula language. Can be used to implement more specialized complex or conditional situations than normal profile to document “data flow” can handle. However, normal data flow should be used in the large majority of cases because it is more efficient and provides field-level auditing. If the document action is configured such that it only triggers when a document is finalized, then a special “pseudo-field” named #DeactivateFlag can be set to true to deactivate the student profile that the document is associated with.
- **Modify Globals Fields:** Set one or more field values in the globals profile using the formula language.
- **Set Fields from Other Document:** This modification copies fields from a source document of a specified template to the current document. The criteria for specifying the other document is a join expression with one or more comparisons, using AND/OR logic, where the left side of each comparison is an expression in the context of the current (target) document and the right side is an expression in the context of the source document. If more than one other document meets the join criteria, the fields are only set from the most recently created document that fits the criteria. The field set expression is also one or more comma-separated equalities, where the left side is a writeable field in the current/document, and the right side is a value expression in the context of the source document.
- **Set Fields In Other Document:** This modification copies fields from the current document to another document of a specified document template. The criteria for specifying the other document is a join expression with one or more comparisons, using AND/OR logic, where the left side of each comparison is an expression in the context of the other (target) document and the right side is an expression in the context of the current document. If more than one other document meets the join criteria, the fields are only set in the most recently created document that fits the criteria. The field set expression is also one or more comma-separated equalities, where the left side is a writeable field in the other document, and the right side is a value expression in the context of the current document.
- **Set Fields in Predecessor Document:** This will only work correctly in situations where one document (Document A) creates another document (Document B) using an “Insert Workflow Document” action. A “Set Fields in Predecessor Document” action can be configured for Document B that sets fields

values in Document A (the predecessor document). In the field assignments, the left side of the equality will be fields in Document A and the right side of the equality will be expressions in the context of Document B.

- **Insert Child Document:** Insert a new child document (i.e. repeating row/section) and set its fields to specified values. This is configured by supplying a “field set” expression (with child document fields being set using document-based expressions); and an optional condition that enables the new child document to only be inserted if the intended child profile does not already exist as detected by the supplied join condition. This join condition compares child document fields with main document expressions, therefore it has a special format consisting of one or more comparisons, using AND/OR logic, with child template fields on the left side of the equality and main document expressions on the right side. If a user message is supplied, it will only be shown to the end user if a child document is actually inserted.

[New in 20.11.1.0] To facilitate using a section action to insert a repeating row in the current instance of a repeating section, an “Insert Child Document” section action for a repeating section supports inserting a grand-child document associated with the repeating section.

- **Update Child Documents:** Update specified field values for the child document(s) that meet the specified criteria.
- **Delete Child Documents:** Remove child document(s) (repeating rows/sections) that meet specified criteria.
- **Create Detail Documents:** Creates detail documents for the master child documents that meet the specified criteria (and that do not already have a detail document). See the section on master/detail documents for more information.
- **Queue Split Child Documents:** This is an advanced modification typically used with staff documents that have a repeating section where each section corresponds to a student and the “Child Document Title” field of the child template is a student profile reference field. When this action fires, the document is queued for a splitting operation that is actually performed by the background service overnight. The child documents specified in the “Queue Split Child Documents Criteria” are converted into “archival” student documents of the specified archival student document template. In this context, “archival” means a document template that is intended to be entirely read only, automatically finalized, and has the “Create via Workflow Only” document template option set so that it can only be created via this type of document action. Any fields in the child document are copied into fields in the archival student document that match by name and date type. Fields in the top level staff document will also be copied to the student archival document where they match by name and data type. The archival document is automatically finalized. Child documents that have been split are removed from the source document.
- **Insert Child Profile:** Insert a child profile for the same profile that the document is associated with. This is configured by supplying a “field set” expression (with child profile fields being set using

document-based expressions); and an optional condition that enables the new child profile to only be inserted if the intended child profile does not already exist as detected by the supplied join condition. This join condition compares child profile fields with document-based expressions, therefore it has a special format consisting of one or more comparisons, using AND/OR logic, with child profile type fields on the left side of the equality and document-based expressions on the right side. If a user message is supplied, it will only be shown to the end user if a child profile is actually inserted. Additionally, if the inserted child profile is in a placement generator queue, and if the user has placement privileges, a link will be added to the message that brings the user to the placement generator.

- **Update Child Profiles:** Update child profile(s) for the same profile that the document is associated with. This is configured by supplying a “field set” expression (with child profile fields being set using document-based expressions); and a “join condition” that determines which child profiles(s) are updated. This join condition compares child profile fields with document-based expressions, therefore it has a special format consisting of one or more comparisons, using AND/OR logic, with child profile type fields on the left side of the equality and document-based expressions on the right side. If a user message is supplied, it will only be shown to the end user if child profiles are actually inserted.
- **Delete Child Profiles:** Deletes child profile(s) for the same profile that the document is associated with. This is configured by supplying a “join condition” that determines which child profiles(s) are deleted. This join condition compares child profile fields with document-based expressions, therefore it has a special format consisting of one or more comparisons, using AND/OR logic, with child profile type fields on the left side of the equality and document-based expressions on the right side. If a user message is supplied, it will only be shown to the end user if child profiles are actually deleted.
- **Set Document Fields from Child Profile:** This modification copies one or more field values from a particular child profile to the current document, where the child profile is the first one found that meets the specified criteria (automatically confined to the parent profile that owns the document). This modification will typically be used when a document is created to populate initial values.
- **Populate Child Documents From Child Profiles:** This modification populates child documents in the current document with child profiles from either a specified parent profile or the default profile that owns the documents. When creating this type of modification, you select the source child profile type (any child profile type of any top level profile type) and the target child template (in the current document template). You specify an expression in the document context that identifies the parent profile of the child profiles to be copied, or you can leave this expression blank to default to the profile that owns the document. You also specify a field set expression where the left side of each equals operator identifies a child template field to be written, and the right side of each equals operator is an expression specifying a source value to be copied from the child profiles. Child documents will only be populated if child documents do not already exist with data in them. This modification can also be used to populate grandchild documents, and in this case, the expression that identifies the parent profile is in the context of the parent child document. In this case, the modification will populate grandchild documents for all parent child documents except when the modification is a section action of a

repeating section (in which case only grandchild documents for the current instance of the repeating section will be populated).

- **Populate Child Documents from Top Level Profiles:** This enables child documents in the current document to be populated by copying data from top level profiles of a specified profile type. A join expression with the format “DocumentExpression1=ProfileExpression1 AND DocumentExpression2=ProfileExpression2” is used to identify the source profiles. You also configure a field set expression in the format “ChildDocField1=ProfileExpression1, ChildDocField2=ProfileExpression2,...” to specify the data that is written.
- **Populate Child Documents from Other Document:** This enables child documents in the current document to be populated in by copying child documents from a source document from another template. From a configuration standpoint, you choose the target child template and the source document template. The action requires that the source document template has a child template with the same ID. Only fields with the same name and data type will be copied. To select the specific source document, you specify join criteria in the format
ThisDocumentExpression1=SourceDocumentExpression1 AND
ThisDocumentExpression2=SourceDocumentExpression2...

Modification actions of this type support a “Include Grand Child Documents” checkbox option. If enabled, when the top level child documents are populated, the action will carry along with them any grand children documents constrained to when the grand child templates match by template identifier and there is at least one field that matches by name and data type

- **Populate Child Documents from Keyword Table:** This data modification action populates child documents from a keyword table. It requires that the child template have a keyword data field with the “child document title” property. Ideally this is used upon document creation and results in one child document for each keyword in the same order as the keyword table. If this modification is used in other circumstances, it will only add missing rows at the end (based on the keyword field).
- **Insert Child Documents from Other Child Documents:** This data modification can be used to insert child documents in a target child template using data selected from existing child documents from a source child template. This is configured by supplying a “field set” expression specifying the target fields that should be set using source expressions, and a “join condition” which supports only copying child documents from the source that do not already exist in the target. This join condition compares target child document fields with source child document expressions; therefore it has a special format consisting of one or more comparisons, using AND/OR logic, with target child document fields on the left side of the equality and source child document expressions on the right side.
- **Update Child Documents from Other Child Documents:** This data modification can be used to update child documents in a target child template using data selected from existing child documents from a source child template. This is configured by supplying a “field set” expression specifying the target fields that should be set using source expressions, and a “join condition” which supports matching child documents in the target with those in the source for the purposes of updating/copying

data. This join condition compares target child document fields with source child document expressions; therefore it has a special format consisting of one or more comparisons, using AND/OR logic, with target child document fields on the left side of the equality and source child document expressions on the right side.

- **Insert Workflow Document Via Child Document Title Field:** This data modification method can be used to implement a cross-profile type workflow. For example, if you have a repeating section in a staff document where the child document title of the associated child template is a profile reference field to a student profile, then this modification can be used to insert a specified student workflow document. This is only available in section actions for a repeating section (with a child document title field that is a profile reference). Such a section action would typically be associated with a button directive where the button creates the new workflow document. The modification will automatically recognize a situation where a workflow document was already created in a previous execution and not create additional documents. If the action is associated with a button, the button will automatically not appear if a workflow document was already created.

As of version 16.0, this modification is now also supported in sections that are not repeating sections. In this newly supported scenario, you select the child template and the criteria for which child documents will launch workflow documents. For example, you could use logical field checkboxes in repeating rows to launch workflow documents (be sure to force them as read only once workflow documents are launched).

- **Update Profiles via Child Document Title Field:** This data modification option that is designed for use in “group plan” documents, typically configured as staff documents with repeating sections/rows focused on students in the group. The data modification relies on the “students” child template within the staff document having a “student profile reference field” marked with the “Child Document Title” property. Based on what student profiles are referenced from the “Child Document Title” field, the modification can be configured to write field data to one or more student profiles from the staff document.
 - If the data modification action is configured as a section action for a “students” repeating section within the staff document, the data modification will write to the individual student profile referenced from one instance of the repeating section.
 - If the data modification is configured as a document action or section action linked to another type of section, it will write to all student profiles referenced by the repeating data, but optionally constrained by the configured join condition. For example, such a document action can be configured to update one or more fields in all referenced student profiles when a staff document is finalized.
 - It should be noted that the join expression formula for this type of modification is typically left blank, but if additional constraints are required for when a student profile is updated, the join expression can be useful.

- **Post Event:** Posts an event that is linked to the document. You specify the subject and body for the event in the section action, and you can use HTML directives to include variable elements that personalize the event to the student and situation. For section actions, note that only fields linked to the section can be referenced from a field directive, e.g. {fieldname}, although formula-based directives are a way around this limitation. For document actions, only the ID and name fields can be referenced from a field directive, e.g. {LastName}, although again, formula-based directives are a way around this limitation.
- **Send Message:** Sends a message to one or more staff users that are referenced via staff fields in the document. You specify the subject and body for the message in the section action, and you can use HTML directives to include variable elements that personalize the event to the student and situation. For section actions, note that only fields linked to the section can be referenced from a field directive, e.g. {fieldname}, although formula-based directives are a way around this limitation. For document actions, only the ID and name fields can be referenced from a field directive, e.g. {LastName}, although again, formula-based directives are a way around this limitation. There is also an option to send a “high importance” message.

Note that if you want to automatically send a message when the status of a document has changed, you do not need to use a guided action. Instead you can enable the auto-notification properties of specific staff reference fields in the document. And if the intended recipient is not directly referenced in the document, you can add a “calculated” staff reference field to the document for auto-notification purposes.

Prior to version 19.4.2.0, document/Section actions with the “Send Message” data modification method previously required a comma separated list of staff profile reference fields or expressions identifying staff to receive the message. Now there is an alternate approach of simply specifying an asterisk (*) wildcard to include all staff referenced in any staff profile reference field with exceptions defined by those staff profile reference fields that have the “No Staff Notification” property. If no child template is selected in the document/section action, the message will be sent to all eligible staff fields even if embedded in nested child templates. However, one can select a specific child template to limit the scope to staff fields in a particular child template. Note that if a specific staff user is referenced in more than one such field, that staff user will only receive one message per document/section action. In some cases, it may be desirable to only send a message via a particular staff field if other conditions are true. If one is configuring staff expressions, one can do this by including IF(<condition>, stafffield) as an expression. But if using the * wildcard, one can accomplish the same thing by marking the staff field with the “No Staff Notification” property, but then creating a corresponding calculated staff field that includes the condition and which does not have “No Staff Notification”.

- **Create Calendar Events:** Creates a personal calendar item on the calendars of one or more staff with optional reminders prior to the date. When setting up this type of document or section action, you specify one or more staff references separated by commas and the expression that gives the scheduled date or date time. You can optionally indicate that calendar item should include a reminder either one week or two weeks prior to the scheduled date. This type of action is designed so that if it is executed more than once, it will replace any previous calendar items it creates.

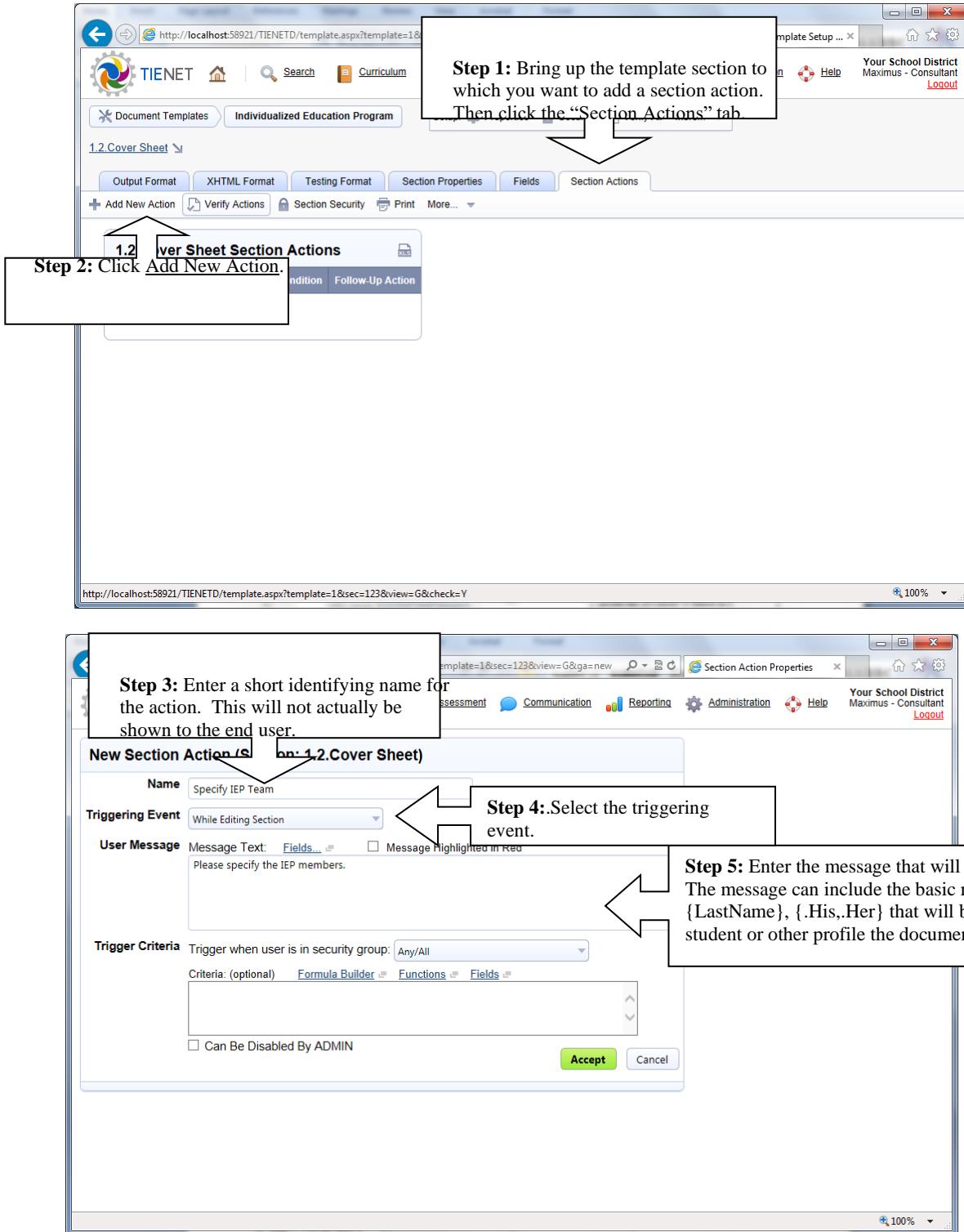
- **Queue Send Email with PDF:** This type of data modification is only allowed for a document action with a “Modify Data While Changing Status” trigger. Only the “To Final” status option is allowed. When triggered (by a document being set to final), the document is placed in a queue. When the queue entry is processed, the document is converted to PDF and sent directly to the specified Internet email addresses. You specific the email addresses in the document action as one or more comma-delimited character expressions that typically reference fields containing email addresses. You also specify the subject and body in the document action, and the text for these can contain macros that reference various fields. The document action also has a checkbox option that will save the generated PDF as a file attachment for the document.
- **Insert Revision Document:** Automates the process of creating a revision document and is only available for a document action that sets the document status to final, or for scheduled document actions in which case the document action is ignored unless the document is final. Note that an original document can only have one revision document, but the revision document of the original document may itself have a revision document and so forth. This document action will execute in the context of both the original document and subsequent revision documents unless the IsRevision value is used in the trigger formula to control what context it executes in.
- **Insert Workflow Document:** Creates a new document of the specified target template such that the new document has a successor workflow relationship to the current document. If a document from the same target document template already exists with a workflow relationship to the user’s current document, the action does not execute. This type of data modification is available for document actions, but not section actions. For more comprehensive information on document workflow, see “Document Workflow” on page 244.
- **Set Fields In Predecessor Document:** This type of modification is used to write field values into the preceding workflow document. A specific preceding type of document template is explicitly identified in the configuration of this action. If a document can potentially have more than one type of preceding document template, one can configure a modification action for each potential predecessor document template.
- **Enable Succeeding Workflow Document:** Enables the creation of new document of the specified target template such that the new document has a successor workflow relationship to the current document. While this is similar to ‘Insert Workflow Document’, it does not actually create the succeeding workflow document. Rather once the creation of a succeeding workflow document is “enabled”, a special link to create the new document is then displayed on the main documents list screen. Note that if a document from the same target document template already exists with a workflow relationship to the user’s current document, the action does not execute. For more comprehensive information on document workflow, see the “Document Workflow” section on page 244.
- **Disable Succeeding Workflow Document:** When triggered, this action reverses the effect of enable succeeding workflow document. This has no effect if the succeeding workflow document is not currently enabled, or if the succeeding workflow document has already been created.

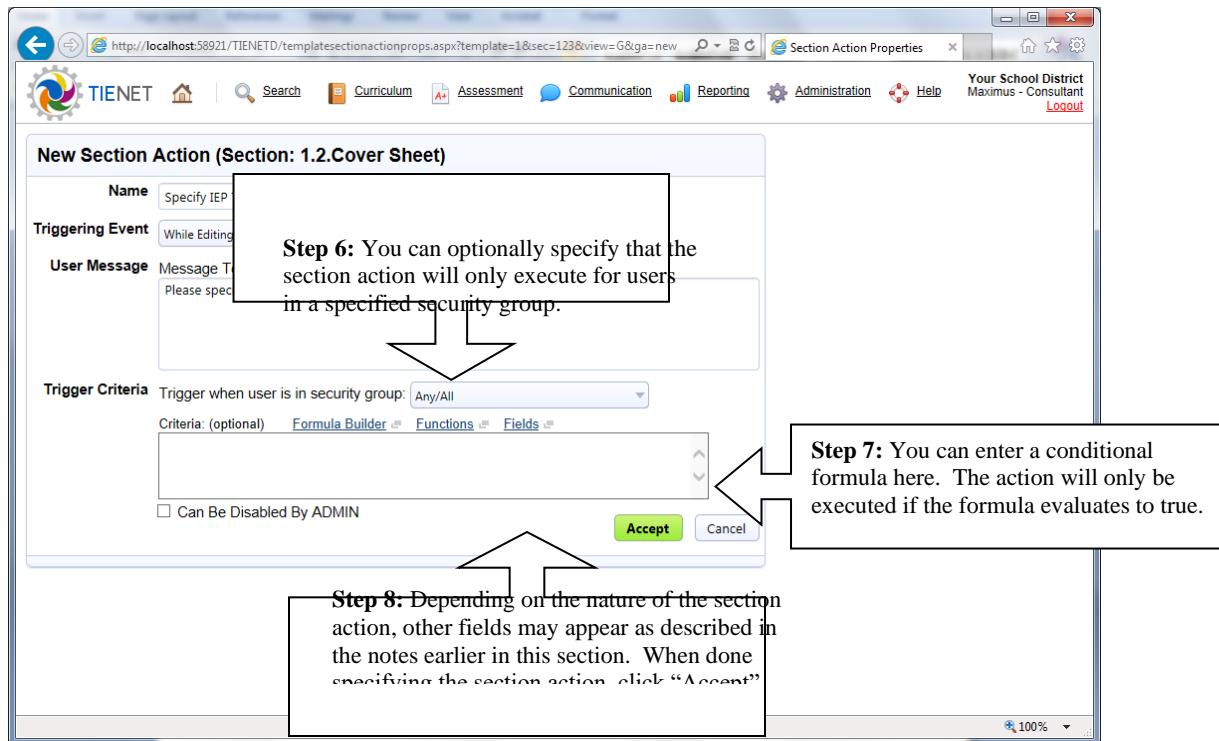
- **Set Document Active Status:** This action is only available in document templates for which “Active Documents” are configured. It automates setting the active status of document to either “active” or “not active”.
- **Clear Active Status in Other Document:** Clears the active status from the active document in the specified document template.
- **Set Document Owner Restricted Status:** When document owner security model is enabled, there is a related suboption “Initially Restrict Document to Owners” that restrict access to the document to the owners only, and excludes users that have general view rights to the document template. This action can be used to remove or set the restricted status of the document. When the restricted status is removed, users with general view access can then access the document.
- **Queue Set Document To Review:** Queues the document to be set to review status. The actual change of status does not occur immediately. Once queued, the background processing service will attempt to change the status one time later in the day. If there are any unmet constraints, the status change may not succeed. Such constraints would be the same ones that would prevent an authorized user from changing the status to review.
- **Queue Set Document To Final:** Queues the document to be set to final status. The actual change of status does not occur immediately. Once queued, the background processing service will attempt to change the status one time later in the day. If there are any unmet constraints, the status change may not succeed. Such constraints would be the same ones that would prevent an authorized user from changing the status to final.
- **Schedule Data Modifications:** Schedules the document such that the document actions that have a trigger type of “Modify Data When Scheduled” will be evaluated and potentially triggered on a specified date (occurs very early on the schedule date, before business hours). A document can actually have three independent schedules labeled A, B, C, D, E or F. Data modifications with label A, B, C, D, E or F will be triggered by scheduling actions with the same label (A, B, C, D, E or F). The date is specified via a date formula that typically calculates a future date based on a date field in the document. Note that the document actions that are scheduled to be evaluated in the future will have their trigger criteria evaluated on the scheduled date, rather than the date that the scheduling occurred. It is also possible for a scheduling action to “de-schedule” the document. If the date expression returns an EMPTY/NULL value, then any scheduling for the document will be removed. A scheduling section action may have a user message configured for it that is displayed at the time that scheduling occurs. It is important to note that this message will only appear to the end user if the “scheduling status” of the document changed as a result of the action. It may be helpful to note that a document can have a single scheduled date associated with each label (A, B, C, D, E or F); therefore the “scheduling status” of a document is considered to change if the scheduled date actually changes to a different value or is removed for that label.

Follow-up Actions: Actions typically display a message for the user when executed. However, depending on the triggering event, actions may allow for the types of “follow-up actions” listed below:

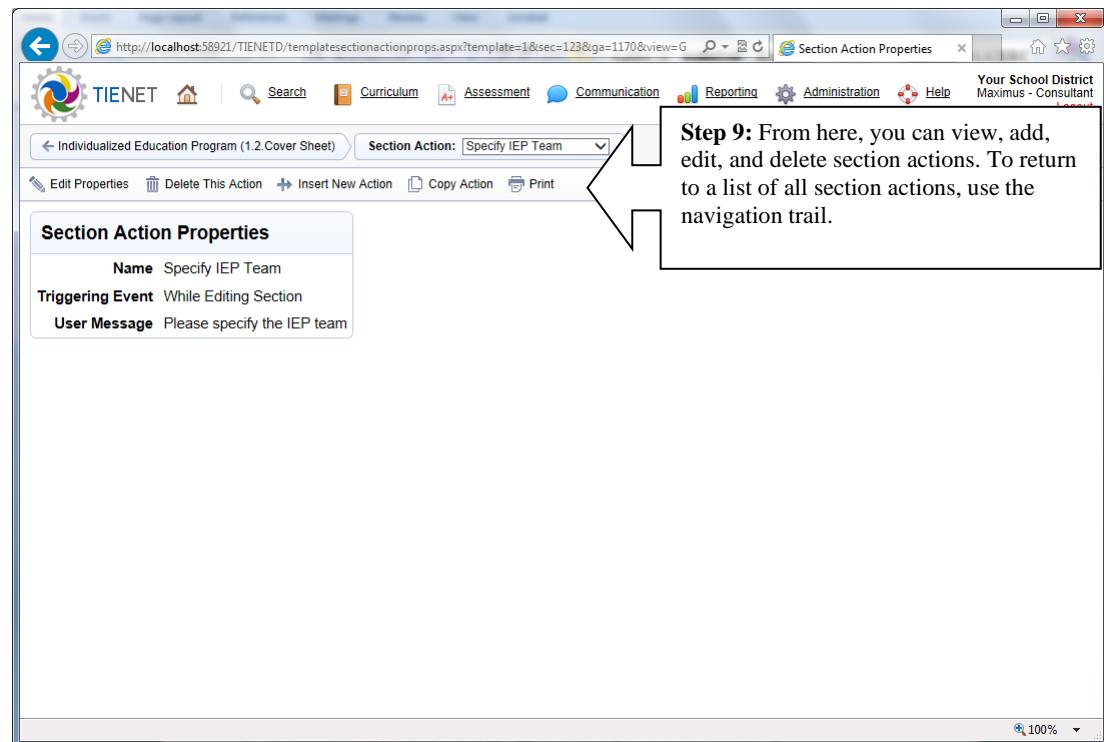
- **Go To Section:** Switches to a particular section without user prompting.
- **Prompt Go to Section:** Prompts the user to switch to a particular section. Action is not executed if the section does not exist in the document.
- **Prompt Add and Go to Section:** Prompts the user to switch to a particular section, adding the section if necessary.
- **Prompt Remove Section:** Prompts the user to delete a particular section from the document.
- **Schedule Meeting:** Prompts the user to create a group calendar item linked to the document (i.e. schedule a meeting), or opens the existing one if it already exists. If the document template has the integrated meetings option set in its properties, certain fields in the document like “MeetingDateTime” and “MeetingLocation” are automatically set in the document as well as in the group calendar item.
- **Post New Event:** Prompts the user to create a new event linked to the document. When you select this option, you will find additional fields in the action that allow you to specify a default subject and description for the new event.
- **Select Review Status:** Prompts the user to set the document status to review.
- **Select Final Status:** Prompts the user to set the document status to final.
- **Go To Other Document:** Prompts the user to go to the latest existing document from the specified target document template as long as the existing document is from the same school year or later as the document the user is working with. If said existing document is not found, the document action prompts the user to create a new document. Note that this action only executes for users who have any view rights to the target document template. This action also only executes if the user’s current document is from the current school year or later.
- **Prompt Create Document:** Prompts user to create a new document from the specified document template (regardless of whether one already exists). Note that this document action only executes for users who have any draft edit rights to the target document template.
- **Prompt Create Workflow Document:** Prompts user to create a new document from the specified document template. In this case, the new document will have a workflow relationship to the user’s current document (more on this). However, if a document from the same target document template already exists with a workflow relationship to the user’s current document, then the document action navigates to the existing document rather than creating a new one. Note that this document action only executes for users who have any draft edit rights to the target document template. For more comprehensive information on document workflow, see “Document Workflow” on page 244.

The procedure below demonstrates how to add a section action:





FYI – About “Can Be Disabled By ADMIN”: This checkbox option allows the configurator to indicate that certain section actions can be optionally be disabled by the system administrator.



DOCUMENT TEMPLATE CUSTOMIZATION (BASICS)

FYI – Verifying Section Actions: To verify all the section actions linked to the selected section, click Verify Actions here.

Action Name	User Message	Condition	Follow-Up Action
While Editing Section	Please document all the modifications and accomodations this student needs.	N/A	
After Section Completed	Check braille is valid You have select braille but the student is not marked as having a visual impairment. Complete Worksheet Please complete Worksheet A because you have indicated the child requires individualized support in the classroom.	Mods_BlindBraille AND NOT Lang_StudentDisabilitiesVisionImpairment Mods_ClsRmAide	Prompt Add And Go To Section > Worksheet A: Individualized Assistance

FYI-Changing Processing Order: If there are more than one action in a triggering event category, you can use the arrow icons to change their processing order.

FYI-Working with Document Actions: To work with document actions, use the flyout menu to select “Document Actions” here. Working with document actions is very similar to working with section actions.

FYI – About Button Directives: A section action with a trigger type of “Modify Data Upon Save” can be configured to trigger only when the user clicks a button that is placed somewhere on the form with a button

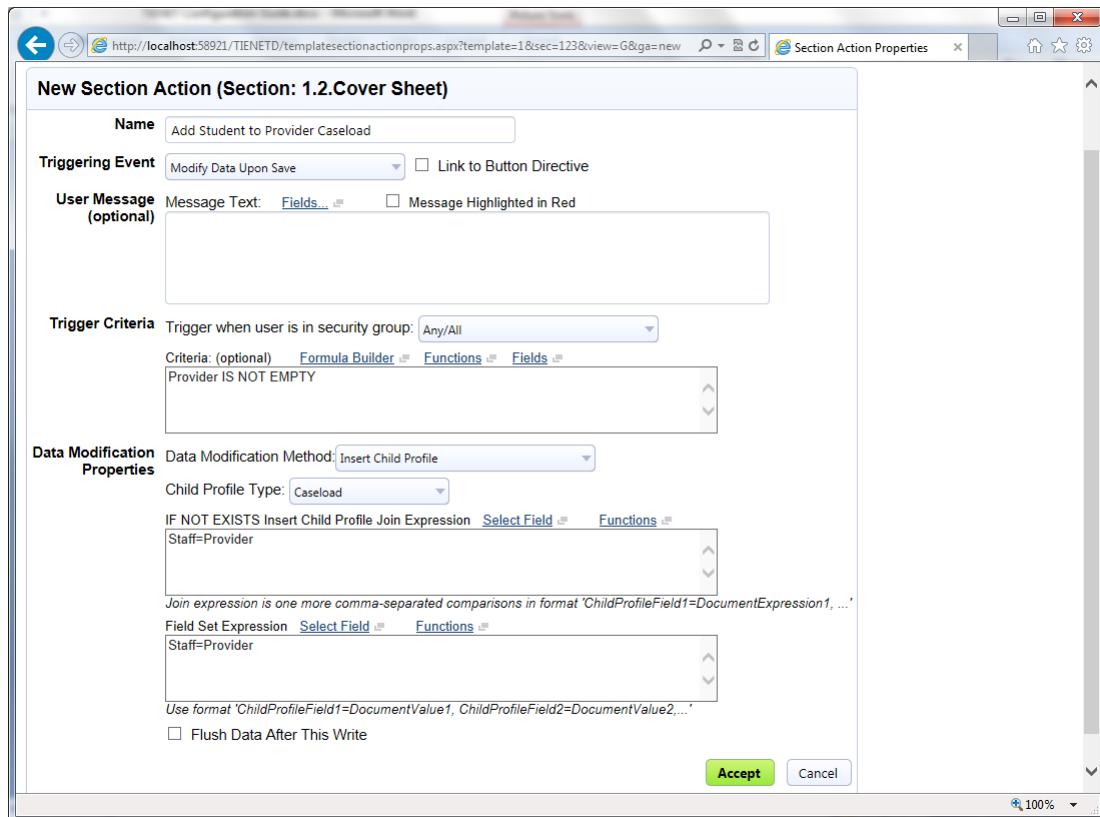
directive. The section action must have the “Link to Button Directive” option checked, and the button directive placed in the HTML format has the form `{*sectionactionname}`. The button will only appear in edit mode. Clicking the button will behave similar to clicking “Save and Continue” with the exception that the section action referenced by the directive will be evaluated in addition to the section actions that evaluate normally.

By default, the button will use the name of the section action as the button label. However, you can use the format `{*sectionactionname:L"alternate label"}` to specify a specific label for the button. Note that if you are translating the form into alternative language, you should provide an explicit label for all buttons so that the label can be translated. The button will use a default CSS class named “SECTIONACTIONBUTTON” which can be configured in the style sheet. However, you can use the format `{*sectionactionname:C"classname"}` to specify the CSS class for a specific button.

It is also possible to insert a button directive not linked to a section action that behaves like the “Save and Continue” button. Omit the section action name and provide a label using the format `{*:L"label"}` (note the colon after the asterisk).

Advanced Data Modification Actions: As an example of the power of data modification actions, it is possible to automatically add a student to a particular staff caseload where the staff member is identified in a staff document field (in this example, a staff field named “provider”). When insert into a many-to-many relationship like caseload, care must be taken to ensure that only valid caseload entries are inserted with no possibility of a duplicate entry; otherwise, the execution of the action will cause application exceptions.

In the example below, the trigger formula of “Provider IS NOT EMPTY” ensures that the action does not attempt to insert an empty provider. The data modification method is “Insert Child Profile” with “Caseload” selected as the child profile type. The “IF NOT EXISTS” join expression should be Staff=Provider, which ensures that the action does not attempt to insert the student more than once into the provider’s caseload. Finally, the “field set expression” actually inserts the current student into the provider’s caseload.



Generally you determine the sequence in which data modification actions execute by configuring their relative order in the list. There are scenarios where it is useful to have one data modification action write a value to the document, and then have a subsequent data modification action read and include that same value in a calculation. It should be understood that the trigger criteria of all actions are evaluated before any data modifications are considered. Still data modifications expressions themselves may reference values modified by previous actions, or include IFELSE logic in them. It should be understood that a series of data modification actions may execute in parallel, not consecutively, although an action lower in the list will never execute before an action higher in the list. Data modification actions have a “Flush Data after This Write” to ensure that the current modification is fully written before any subsequent actions are executed (versus executing in parallel to them). This is useful in certain situations to ensure that the current modification is available to be read by subsequent actions.

Best Practices for Validating User-Entered Document Data

The following features and techniques should be considered for validating user-entered document data while keeping in mind that the simplest technique or feature that can do the job is the best option.

- Field-Level Properties:** In addition to general properties like “Required” or “Prevent Empty Values”, some of the specific field data types have simple data validation checkbox options in the field properties, as follows:

- Date: No Future Dates
 - Date/Time: No Future Dates, Time Required
 - Character: Require Numeric (e.g. zip code), Require Alphanumeric, Email Address.
2. **Section Action with “Rollback Saving Section” trigger:** You can configure a section action triggered by specific formula criteria that identifies invalid data. If the condition is met, the data is not saved and the user message is displayed to the end user. By default, the user message is displayed above the form, but there are two options for displaying the message “in context” inside the form. In the section action, if you enable the “Anchor user message to first referenced field” checkbox, PowerSchool Special Programs will display the user message next to the first field referenced in the formula. This will only work if that field is actually an editable field on the form. Alternatively, you can specify exactly where the user message will appear on the form using a `{*sectionactionname}` directive in the HTML format.
3. **Section Action with “Prevent Section Completion” trigger:** This section action allows the data to be saved, but prevents the section from being marked as complete (and therefore also prevents the document from being finalized). One might wonder when it is better to use this versus the rollback action. Use the rollback action to prevent data from being saved that does not make sense at all, and use this when it makes sense to allow the user to save incomplete work temporarily. It is also worth noting that even if a user completes all sections, all “Prevent Section Completion” section actions are rechecked when the user goes to finalize the document. Without this final check, in some circumstances, some invalid data might get into the final document because of cross-section dependencies and the like.
4. **Value Filtering:** This technique applies to keyword fields and profile selection fields.
- Keyword Fields: You can use a directive in the format `{keywordfieldname:F"keywordtablefieldname"}` to show a subset of keyword choices to the user. Use the “`keywordtablefieldname`” modifier to identify a field in the keyword table to use for filtering the keywords. The field will only show keywords for which the specified keyword table field/column is neither EMPTY nor FALSE. If the keyword table field is itself a keyword selection field referencing another keyword table, the PowerSchool Special Programs will actually produce a “hierarchical” keyword dropdown menu that categorizes the keywords based on the values in that field/column. An example would be “diagnostic code” keywords that are categorized by the service type to which they apply. In this case, ‘service type’ would be a field/column in the diagnostic codes keyword table that references a services types keyword table.
 - Profile Reference Fields: The F modifier can also be used in conjunction with a profile reference, to allow for specification of a filter formula to filter the results displayed by the lookup popup window. The formula should be in the context of the profile type being looked up. When a filter is used within a document, in some circumstances, it is possible to

embed information from the document in the formula using a placeholder enclosed with \$ characters. This is described in the example below. Note that this filter does not prevent the user from manually typing a profile ID that is not in the filter. Therefore you may want to consider also implementing a section action to validate all data.

Example filters for a profile field look-up:

A simple directive to filter a staff “look-up” might look like the following:

```
{Psychologist:F"PersonnelPosition=Psychologist"}
```

The above directive filters the look-up for the Psychologist field to those staff for which the PersonnelPosition is "Psychologist". But let's say, you wanted to filter a Provider field to those staff for which the required personnel position is given in a profile/document field within the same profile/document section. This can be done with a placeholder as shown below.

```
{Provider:F"PersonnelPosition=$RequiredPosition$"}
```

In the above case, the \$RequiredPosition\$ placeholder will get replaced by the value of the RequiredPosition profile/document field. Note that this directive requires that the RequiredPosition field is linked to the section the directive appears in. A final example would be a filter to only show staff that are teachers of the student in the document. This example uses two nested EXISTS statements as follows:

```
{Teacher:LF"EXISTS(ClassStaffRoster, EXISTS(ClassStudentRoster, Student=$Profile$))"}
```

The \$Profile\$ syntax in this example, only works in the context of documents (not profiles) and is set equal to the profile for the current document.

As stated previously, it is important to note that these filters do not prevent the user from manually entering a profile ID that is not in the filter. Therefore, you should consider also configuring a section action to implement a comprehensive check.

Active Documents

In the properties for a document template, there is an “Active Documents Enabled?” option that can be enabled to allow PowerSchool Special Programs to track which document of this type is “active” for a profile, while ensuring that only one document from the template can ever be active for a given profile. Active documents are differentiated in the user interface to call attention to them.

The “active documents” feature is enabled for a document template through the template properties shown below:

Active Documents Enabled? Yes No

- Auto-set documents as active when finalized
- When document set active, queue previous active document for finalization
- When document set active, queue previous documents for status upgrade (draft > review > final)
- Allow view of active documents in user class view

However, some planning is needed to make sure the feature is properly integrated into the business workflow. For planning purposes, it is helpful to understand that a document can be set as the active document in three ways:

- 1) A document can be set as the active document automatically when the document is finalized. In template properties, the “Auto-set documents as active when finalized” option (see above) enables this behavior. An exception to this behavior is made when the document being finalized is not from the current school year and there is already an active document that was created later than the one that is being finalized. In this case, the later document will remain the active document.

IMPORTANT: When you first enable the “active documents” feature for a document template, if you also enable the “Auto-set documents as active when finalized” checkbox, PowerSchool Special Programs will immediately set as “active” all the existing latest finalized documents based on the “Document Finalized On” date. If you turn on the sub-option at a later time, this behavior will not occur. However, there is a feature to set documents as active in bulk that will be described later.

- 2) A document can be set as the active document automatically via a workflow document or section action when the configured trigger conditions are met. Note that if the document the workflow is operating on is not from the current school year and there is already another active document that was created later; the later document will remain the active document.
- 3) A document can be set as the active document manually by an authorized user as long as the document is in review or final mode. There is a new template security right that authorizes this. This security right appears only if the feature is enabled. Note that draft documents cannot be manually set to active since comprehensive quality assurance is not feasible in a draft document. In a future version, this restriction may be relaxed if there is demand for it.

Delete	Status Change Rights				
	Set Draft	Set Review	Set Final	Set Active	
✓	✓	✓	✓	□	✓
✓	✓	✓	✓	✓	✓
✓	□	□	□	✓	✓
□	□	□	□	□	□
□	□	□	□	✓	□

There is a new “Set Active” template security right for document templates that have this feature enabled.

If a document template has “Active Documents” feature enabled and also has progress reporting enabled, then the user can only enter progress for an active document. Progress reports for documents that were once active can be viewed on a read-only basis. A logical field named “WasActive” is maintained in the data dictionary to track which documents were once active. When this feature is first turned on, the WasActive flag is nominally set in all documents from previous years, and this helps ensure that people users can still access progress reports for previous year’s documents.

To set documents to active in bulk, create a list report that lists the documents to be set active. This report must have the option set to include only the latest document as that option assures that the report has only one document per profile in it. An authorized user can then use this report to set documents as active en masse by selecting the “Set Documents as Active” from the “More Actions” dropdown menu. Users with template rights to set documents as active (and view access to the documents) are considered authorized (including ADMIN or CONSULTANT). The bulk operation will skip any draft documents found in the list report.

When creating reports on document that have the “Active Documents” feature enabled, you will find an “Active Documents Only” checkbox that conveniently only includes active documents on the report as shown below.

Selection Criteria	Document Selection Formula	Formula Builder	Functions	Fields
Click 'Formula Builder' for assistance preparing the formula.				
	<input checked="" type="checkbox"/> Include Only 'Not Exited' Students <input type="checkbox"/> Final Documents Only <input type="checkbox"/> Active Documents Only <input type="checkbox"/> For each student, include only th	When create a report on documents that support “active documents”, this checkbox conveniently appears when setting report criteria.		

Explanation of Active Documents Sub-Options:

An explanation follows for each of the sub-options shown below:

Active Documents Enabled?	<input checked="" type="radio"/> Yes <input type="radio"/> No
	<input type="checkbox"/> Auto-set documents as active when finalized <input type="checkbox"/> When document set active, queue previous active document for finalization <input type="checkbox"/> When document set active, queue previous documents for status upgrade (draft > review > final) <input type="checkbox"/> Allow view of active documents in user class view

- Auto-set documents as active when finalized:** A document automatically becomes the current active document when the document is finalized. An exception to this behavior is made when the document being finalized is not from the current school year and there is already an active document that was created later than the one that is being finalized. In this case, the later document will remain the active document.
- When document set active, queue previous active document for finalization:** When this option is enabled, when a document is made the active document, the document that was the previously the active document (if any) will be put in a queue to be set as final (if it is not already final).
- When document set active, queue previous documents for status upgrade:** When this option is enabled, when a document is made the active document, any previous documents (active or not) of the same type that are not already final are queues for a status change. Draft documents are set to review and review documents are set to final.
- Allow view of active documents in user class view:** This enables teachers to access a view of the active documents for all students in the class without leaving the class roster screen. Additionally, this document list view can be configured to display specific document field values by setting the “Include in ClassView” option in the properties of the desired document fields

Enforcing one active document across two or more document templates: It is a common scenario that there should be only one active document per profile across two or more document templates. In this case, each of the document templates should have a document action with a trigger event of “Modify Data While Setting Document as Active” and a modification method of “Data Modification Method: Clear Active Status in Other Document > (*Other document template to clear*)”.

Special Document Fields

If you add fields to a document template that have certain special names and data types, PowerSchool Special Programs will automatically recognize them, populate them in the appropriate circumstances, including in past documents:

DocumentCreatedBy: If a staff reference field is added with this name, PowerSchool Special Programs will automatically set it to the staff user who created the document (or EMPTY for ADMIN/CONSULTANT). Alternatively, the system will also recognize a character field (length=100) of

the same field name and set it to the name and ID of the user who created the document (including for the ADMIN/CONSULTANT).

DocumentPrintedOn: If a date or datetime field is added with this name, PowerSchool Special Programs will automatically set it to the current date or date/time when the document, or any part of it, is printed.

DocumentPrintedBy: If a staff reference field is added with this name, PowerSchool Special Programs will automatically set it to the latest staff user who printed any part of the document. PowerSchool Special Programs will also recognize a character field and set it to the name and ID of the current user.

DocumentFinalizedBy: If a staff reference field is added with this name, PowerSchool Special Programs will automatically set it to the current user when the document is finalized (or EMPTY for the ADMIN/CONSULTANT). Alternatively, the system will also recognize a character field (length=100) of the same field name and set it to the name and ID of the user who finalized the document (including for the ADMIN/CONSULTANT).

ForceFinalized: If a logical field with this name is created for a document template, it will be automatically populated with true for force finalized documents.

Snippets

Snippets are a feature that is primarily targeted towards the system administrator (ADMIN) or CONSULTANT users to enable database-specific customization of document templates without deviating from the corresponding state model database. Snippets are fragments of HTML formatting that can be inserted at specific points in the existing HTML section format. Snippets typically incorporate custom document template fields (fields marked as custom in the data dictionary for the document template).

To add snippet content to a section as a consultant user, just go the configuration for that section and select “Add Snippet” from the lower “More” dropdown menu. You then specify the snippet position and HTML formatting which typically references “custom” document fields. Additionally, there is a “Deployed” checkbox that allows you to save changes to the snippet without necessarily deploying it yet. When ready to deploy, check the “Deployed” checkbox and save. Keep in mind that snippets are not synchronized by the Configuration Management Tool.

[New in 20.6.3.0] To support a new and more robust approach to snippets which will culminate in a major new feature set called “Easy Edit” in the future version 20.11, the system now automatically injects snippet-id attributes into key HTML tags of document section formats, but only if the database is on a specific model version. When you create snippets for document templates with 20.6.3.0, there are now three main choices: Beginning of Section, Tag with Snippet ID, and End of Section. Only “Tag with Snippet ID” is new, and when you select this choice, you can enter the target tag’s snippet ID attribute value and select one of the following actions:

- 1) **Insert Above Outside:** Inserts the new snippet content above the outside of the target tag. For example if the target tag is `<p snippet-id="p-21">contents</p>` and you specify the snippet content as a new p tag, your new p tag will be inserted above the target p tag (not inside of it).

- 2) **Insert Above Inside:** If the target tag is <p snippet-id="p-21">contents</p> the snippet content you specify will be inserted as <p snippet-id="p-21">(your snippet)contents</p>.
- 3) **Replace:** If the target tag is <p snippet-id="p-21">contents</p> the snippet content you specify will be used as a replacement : <p snippet-id="p-21">(your snippet)</p>.
- 4) **Insert Below Inside:** If the target tag is <p snippet-id="p-21">contents</p> the snippet content you specify will be inserted as <p snippet-id="p-21">contents(your snippet)</p>.
- 5) **Insert Below Outside:** Inserts the new snippet content below the outside of the target tag. For example if the target tag is <p snippet-id="p-21">contents</p> and you specify the snippet content as a new p tag, your new p tag will be inserted below the target p tag (not inside of it).

System administrators are expected to take a focused training before utilizing the snippets functionality because of the potential adverse effects of improperly configured snippets. For this reason, the ADMIN user does not have access to the functionality until enabled by the CONSULTANT user, and the snippets functionality is not documented in the “System Administration Guide”. The CONSULTANT can enable access by the ADMIN user by selecting “Security” from the “Administration” menu, and then “Set Admin Options” from the “More” dropdown.

Document Template Customization (Advanced)

Chapter

7

Document Owner Security

Document templates can be configured to allow only specified owners of a document to view that document, although administrative security groups can still be authorized to view documents regardless of owner. To configure this form of security, follow the steps below:

1. Enable the “Allow Document Owner Security” option in the template properties of the document template. Note that once this option is enabled, there will be an addition document template security right labeled “View if Owner”. Users with this right can view documents of this type only if they are the owner. However, users with the standard “View” right will be able to view documents of this type whether or not they are the owner.
2. Configure one or more staff profile reference fields within the document template and assign the “Document Owner” field property. The staff referenced by these fields will be the “owners” of the documents. You must configure how these fields will be populated. An administrator could create the document and fill the field thereby assigning the document to the specified owners. Alternatively, document or sections actions could be used to populate these fields in order to assign ownership. For example, to set ownership of a document to the creator of the document, you can configure an action triggered by document creation and that sets the “document owner” field to the current user.

Note that document owner security is enforced in all areas of PowerSchool Special Programs including standard reports and advanced reports.

Document owner security, when implemented in staff documents, can provide a full-featured alternative to self-service documents. If you enable the “Allow Document Owner Security” in a staff document template, you can also enable a secondary option labeled “Staff Profile Is Document Owner”. This secondary option causes the staff for whom a document is created to automatically become the owner. Follow the steps below to configure a security group that allows it members to create/edit/view staff documents for their own staff profile (while excluding other staff from accessing those same documents):

1. Under staff privileges, make sure the basic view staff privilege is enabled. Also enable “View Documents” and “Create Edit Template Documents”. Note that this will not allow access to any template-based documents unless template rights are assigned. However, if you are using this approach, do not use file-based documents for staff because file-based documents do not support document owner security and therefore other staff will be able to see them. If you need the equivalent of file-based documents for staff, use file-attachment only document templates instead.

2. For each relevant staff document template, grant the “View If Owner” right and any additional editing rights to the document template. The bottom line is that any member of this group will only be able to view/access those staff documents for which the user is the owner. If you grant the standard “View” right, that will enable members to access other staff users’ document.
3. Setting up the security like above will cause a “My Staff Documents” link to appear on the home page of members that give them quick access to their documents. However, make sure the “Access Self Service Documents” privilege under “Special Access Privileges” is turned off because otherwise a link to “My Self-Service Documents” will appear instead of the link to “My Staff Documents”.

There is a related “Initially Restrict Document to Owners” document template property option that can be enabled in conjunction with the document owner security model. The option initially restricts access to new individual documents to the owners only, and prevents even staff users that have general view access from accessing them (restriction does not apply to ADMIN/CONSULTANT users). In the profile document list, documents with this restriction will have a padlock icon next to the document name. A document action modification of “Set Document Owner Restricted Status” is used to set or remove the restricted status at the appropriate time. After removal of the restricted status from a document, staff users with general view access will be able to access it.

Document Workflow

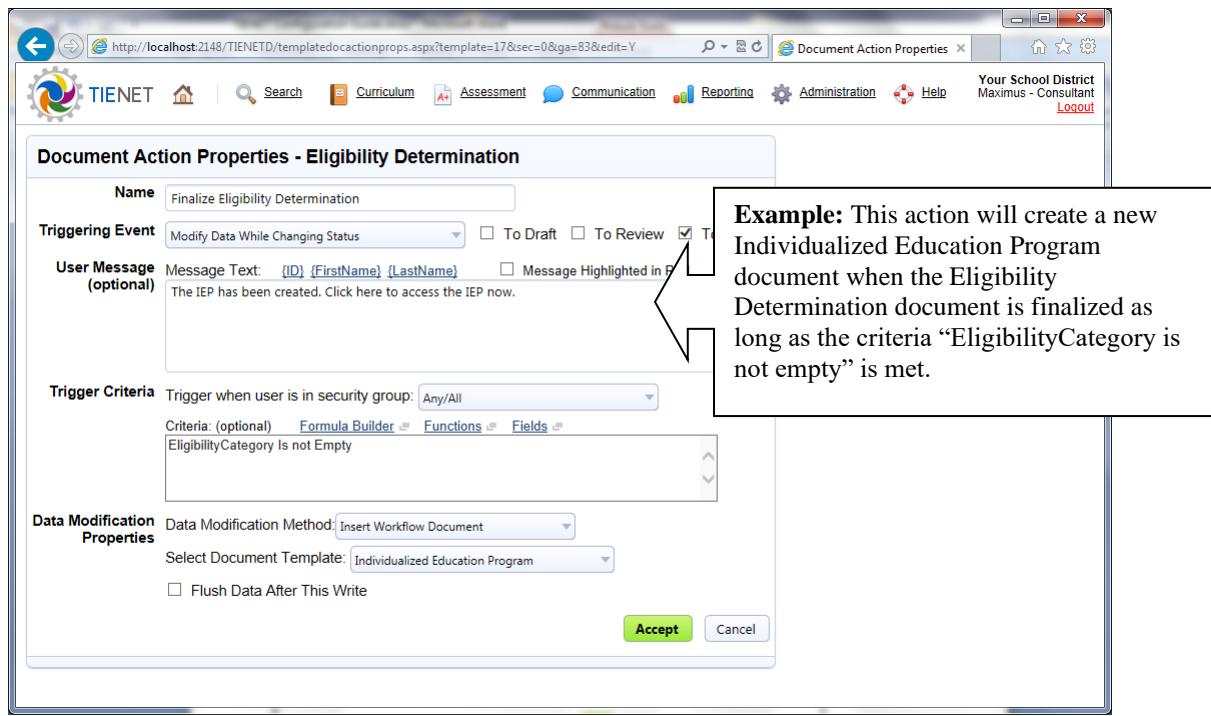
Document workflow refers to the ability of PowerSchool Special Programs to support a process that is larger than one document template through sequencing of documents and potential data flow from one document directly to the next. A simple typical document workflow for Special Education might be:

- Pre-Referral or Interventions
- Referral
- Assessment
- Eligibility Determination
- Individualized Education Program

The document-to-document sequencing is established by creating a document or section action in each relevant document template that triggers creation of a new document in the next document template. The document/section action should be designed to trigger at the right time, and have the right criteria for moving forward in the workflow process. The new document that is created has an explicit workflow link to the document that was used to create it, and this relationship is explicitly shown in the document list.

There are three types of document/section actions that can be used to create new workflow documents (i.e. documents that have a successor relationship to a previous document in a flow). The three approaches differ with respect to the degree of automation with which the succeeding workflow document is created.

- Insert Workflow Document:** This is a data modification option associated with document action triggers that modify data, such as “Modify Data While Changing Status”. Section actions do not support this. Such document actions create a new document of the specified target template such that the new document has a successor workflow relationship to the current document. The new document is created regardless of whether the user actually has access to the new document. If a document from the same target document template already exists with a workflow relationship to the current document, the action does not execute to avoid creating multiple successor documents. This type of action can optionally have a user message, but the user message will only appear to the end user if the end user has view rights to the next document created. Such messages appear also a hyperlink to the new document created. When using this type of document action, consider enabling the “Create Via Workflow Only” option in the template properties of the target document template to prevent documents of that type from being created with no workflow relationship.

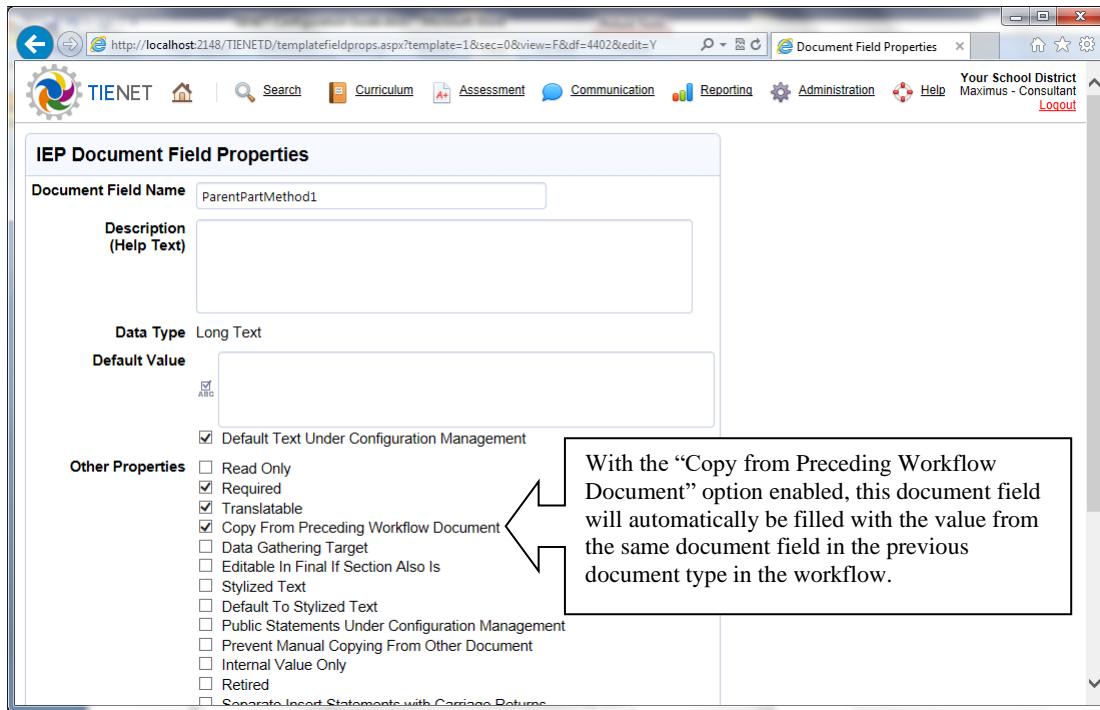


- Enable Succeeding Workflow Document:** This is a data modification option associated with triggers that modify data, such as “Modify Data When Section Completed”. This enables the creation of a new document of the specified target template such that the new document has a successor workflow relationship to the current document. If a document from the same target document template already exists with a workflow relationship to the user’s current document, the action does not execute. While this is similar to ‘Insert Workflow Document’, it does not actually create the succeeding workflow document. Rather once the creation of a succeeding workflow document is “enabled”, a special link to create the new document subsequently appears on the main documents list screen. Additionally, if the user attempts to create the next document directly from the new document menu, PowerSchool Special Programs offers the user the option of attaching the

new document to the pending workflow process. Also, note that the action will enable creation of the succeeding document even if the current user does not have access to the succeeding document.

- Prompt Create Workflow Document:** A number of different document and section actions support a “Prompt Create Workflow Document” follow-up action where one can specify the document template for the document that the action will create. In actual use, the action will prompt the user to create a new document that will have a workflow relationship to the user’s current document. If a document from the same target document template already exists with a workflow relationship to the user’s current document, the action does not execute. Note that this document action only executes for users who have any draft edit rights to the target document template. Note that this type of guided action does not guarantee creation of the new document, but rather offers the user the option of creating it. Since this approach is purely a prompt and does not enforce the workflow, this could be used in conjunction with one of the previous approaches.

FYI – Copying data from preceding workflow document: When configuring document workflow, you can configure document fields to flow from Document A to Document B by marking the fields in Document B with the “Copy from Preceding Workflow Document” option as show below. To successfully flow, fields marked this way must also exist in Document A (with the same name and data type). A similar property can be set in child template properties within Document B that enables child template data to flow from Document A to Document B. To successfully flow, the child template must also exist in Document A with the same child template ID. Then any fields that exist in both child templates (with the same name and data type) will flow.



A possible approach is to have a few documents on the new document menu that are starting points into the Special Ed Process, and then to completely automate creation of the remaining documents via a workflow process. To do this, you would:

- Identify the document templates that are entry points into the Special Ed process like "Referral". Make sure these have workflow actions that create the subsequent documents according to logic of the business requirements.
- For the subsequent documents, enable the 'Create via workflow only' template option. Documents with this option enabled do not appear on the new document menu.

Note that the data gathering mechanism described in the next section is not enabled when the document workflow mechanism described here is being used.

Data Gathering

PowerSchool Special Programs has a previously described feature called “data flow” which allows document fields to be linked to profile fields such that any field values in the profile automatically flow into the corresponding document fields, either when the document is created or when it is later updated from the profile. You may have a situation where it would also be beneficial for values entered into previous documents (from the same template or even from a different template) to flow directly into a new document. The “Data Gathering” feature allows you to do this, but keep in mind that “data gathering” will only work if the document fields involved have the same name and data type in the source and target templates. Additionally, in the “document workflow” scenario described previously, the data gathering mechanism will not be used (i.e. document workflow overrides data gathering).

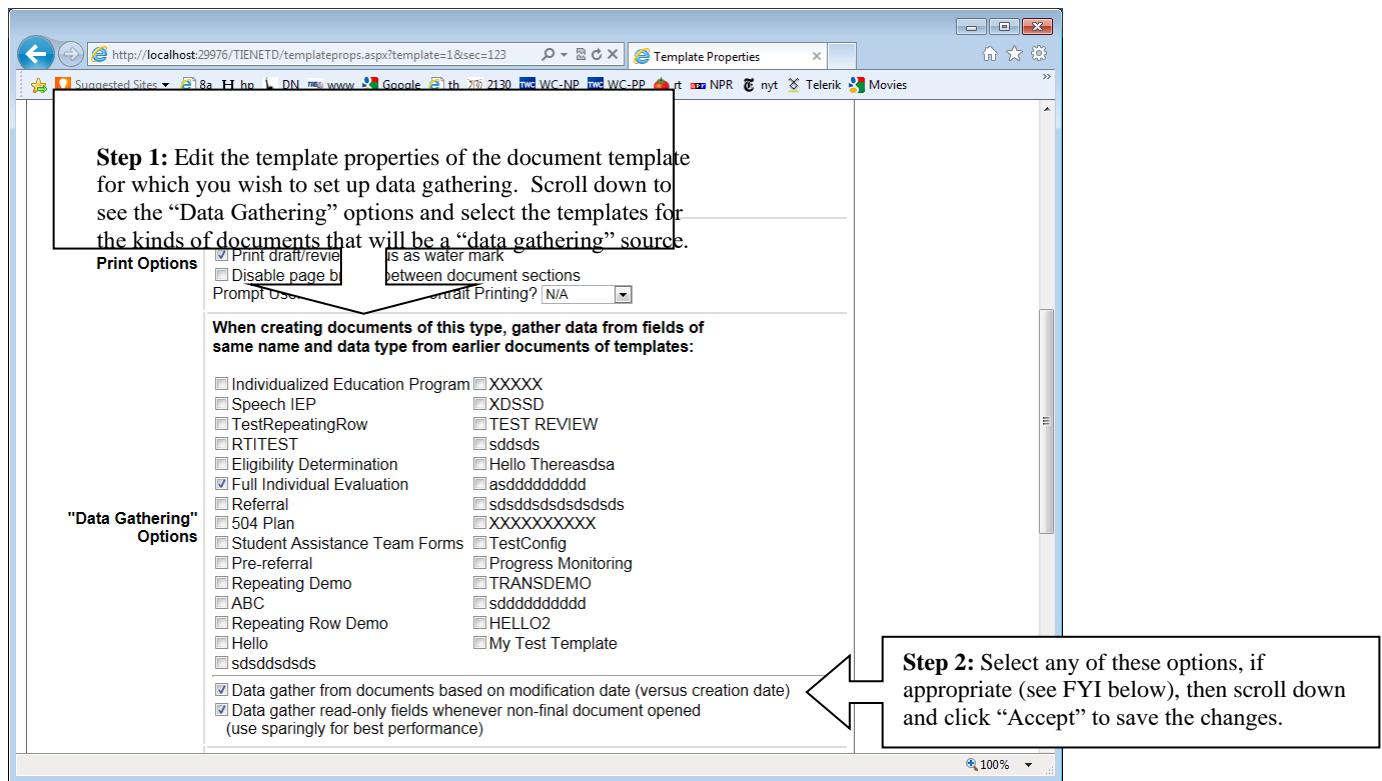
The material in this section will refer to data gathering sources and targets. A source is a template, document or field that data will flow FROM. A target is a template, document or field that data will flow TO. When a new document is a “data gathering” target, data can flow into it from more than one source document, but at most one source document per source template. If there is more than one potential source document for a source template, PowerSchool Special Programs selects the latest document based on the document’s creation date (or alternatively the modification date if so configured).

Configuring data gathering for a target template has two major steps:

In the target template, you identify which templates can act as source templates (via template properties).

In the target template, you mark the document fields that are to be target fields (via field properties). Note that a document field that is linked to a profile field cannot be a target for data gathering.

To set up “data gathering”, follow the steps below:



FYI – About “Data Gathering” options: There are several yes/no options in template properties that modify the behavior of “data gathering”, as follows:

Data gather from documents based on modification date (versus creation date): When PowerSchool Special Programs performs data gathering, it will gather from only one source document per template. Normally, if there are several potential source documents, PowerSchool Special Programs selects the latest based on creation date by default. If this option is enabled, it selects based on modification date instead.

Data gather read-only fields whenever non-final document opened: Normally, data gathering occurs only when a target document is created. The yes/no option described here can be useful in situations where source documents may be modified after the target document is created. If this option is enabled, data gathering occurs whenever a non-final target document is opened, but this is limited to fields that are read-only in the target document.

DOCUMENT TEMPLATE CUSTOMIZATION (ADVANCED)

Step 3: Edit the field properties of the document fields that you want to set as a data gathering target. Enable the “Data Gathering Target” checkbox and click Accept. Note that this option will not appear if the document field is linked to a profile field. Once all the fields have this property enabled, “data gathering” should work. However, since setting this property for each individual field can be tedious, there is a shortcut as described next.

Step 4: There is a shortcut for setting data gathering all for multiple fields at once. With the fields list showing, click Set Data Gathering. You will then see a list of fields that can be potentially gather from other templates, and you will be able to easily set data gathering for multiple fields at one time.

	Properties	Description
Address	Contains the student's address. Hello There	
Birthdate	Contains the student's birthdate.	
CaseManager	Contains the student's case manager. Entry comes from the 'Personnel' list.	
CaseManagerParticipate	Logical Value (prevent empty values)	
CB_Evaluation_Rep	Logical Value (prevent empty values)	

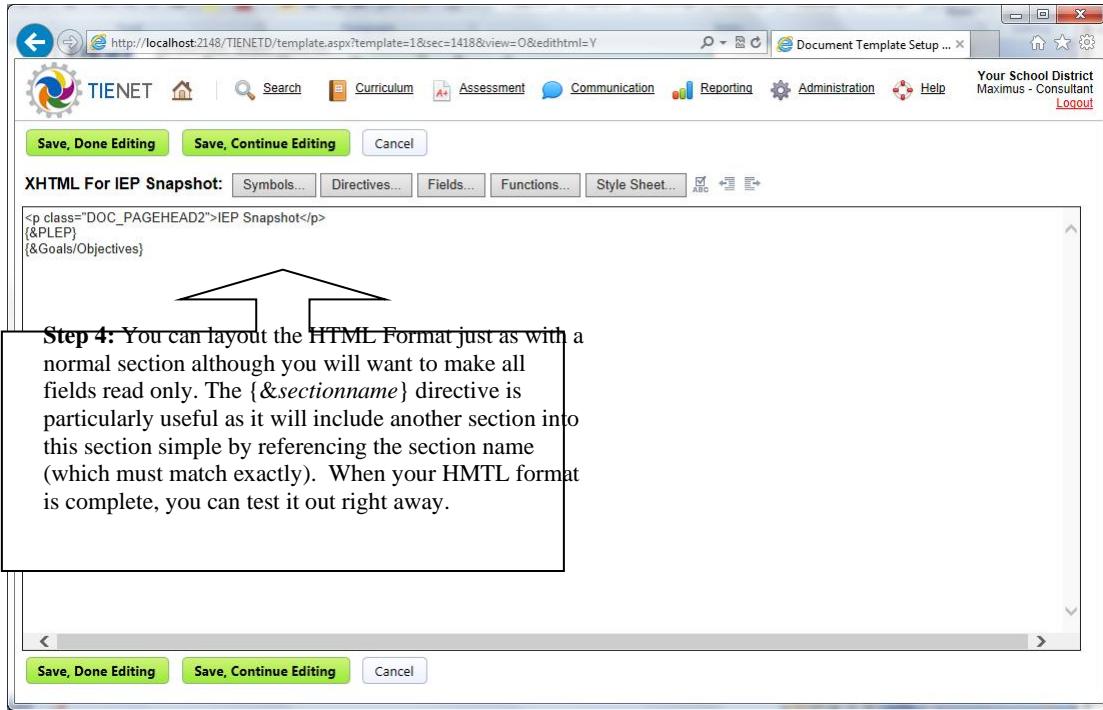
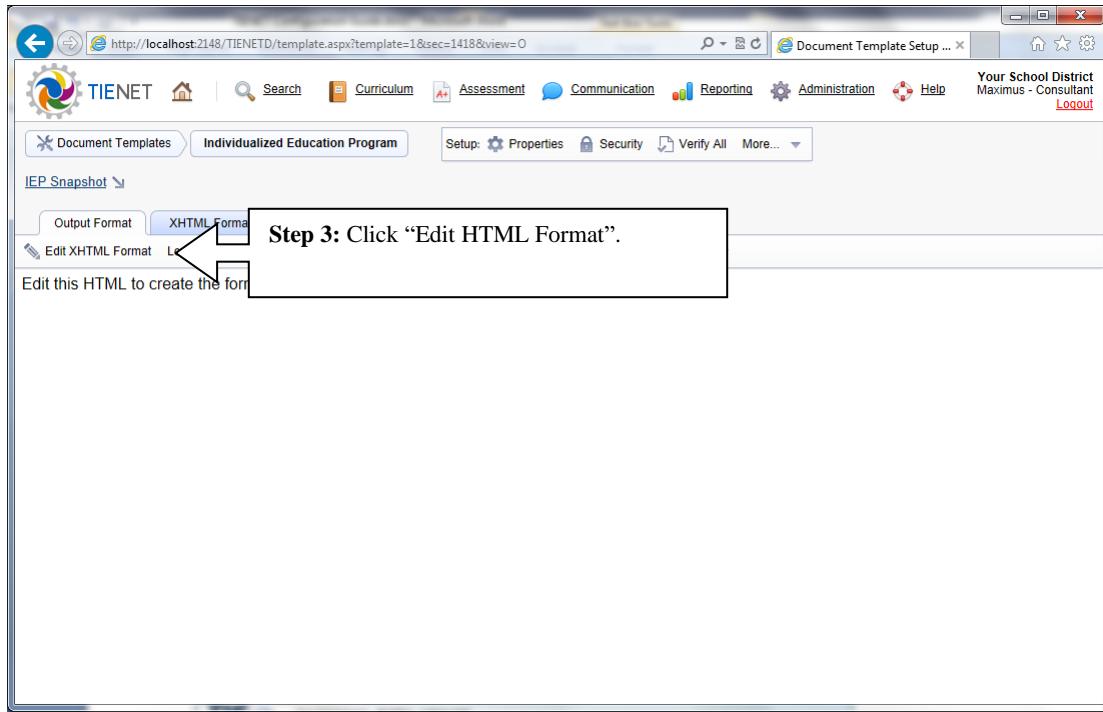
FYI – About Data Gathering Report: If you wish to see possible data gather sources for a particular section, bring up that section in the document template via “Data Schema”. Switch to the “Linked Fields” display and click the Data Gathering Report link.

FYI – Data Gathering versus Data Flow: When a new document is created, first any fields marked for “data flow” from profile to document are copied into the new document. Then the “data gathering” process is applied which could possibly override values copied from profile to document via “data flow”. This functions differently than the process of explicitly copying from a previous document that the user can select via the user interface. When a user explicitly copies from a previous document, fields configured to flow from profile to document are never copied.

IEP “Snapshot” Section

An “IEP Snapshot” section is simply a section at the end of the IEP template that repeats the most critical information from the IEP document to produce an abbreviated version of the IEP document. There are several techniques that facilitate configuring such a section. Follow the steps below:

The screenshot shows the 'Template Section Properties' page for TIENET. The URL is <http://localhost:2148/TIENET/templatesectionprops.aspx?template=1&sec=new&secret=123&view=0>. The page title is 'Add New Section - Individualized Education Program'. The 'Section Name' field is set to 'IEP Snapshot'. A callout box labeled 'Step 1:' provides instructions: 'Initiate adding a new section to the document template for the snapshot. Name the section.' The 'Repeating Section?' dropdown is set to 'No'. Under 'Section Options', the 'Include by Default in New Documents' checkbox is checked. A second callout box labeled 'Step 2:' provides instructions: 'If users will not be entering any information directly into the snapshot, it is appropriate to set the "Preset as Completed in New Documents" option.' Below this, another callout box says 'Click Accept to add new section.' Other settings include 'Section Behavior' options like 'Prevent Removal from Document' and 'Section Status' set to 'Active'. The 'Section Mapped to Curriculum?' dropdown is set to 'Curriculum'.



Repeating Sections/Rows and Child Templates

The district may need to be able to have multiple copies or instances of a particular section within the same document. To provide this capability, you need to configure that template section as a “repeating section”.

This feature can be used, for example, to implement a goals/objectives section. Alternatively, the district may require a table of information with rows and columns, including the ability to add additional rows as needed. This can be provided using the “repeating rows” feature. To implement a repeating section or repeating rows within a section, you must first create a child template within the document template. A child template is simply a storage area for a subset of fields/values that can repeat themselves in the same document, either in the form of a repeating section or repeating rows within a section. So the child template has its own data dictionary and the data records it contains are generally referred to as “child documents”.

A child template can be used to store and provide the data simultaneously for a repeating section and a repeating row layout in another section. It is also possible to implement a child template within a child template (i.e. a grand-child template), which can be used to implement repeating rows inside of repeating sections.

Creating a Child Template: To create either a repeating section or repeating rows, you must first determine the exact set of fields with values that will be repeated and create a child template containing those fields. To create a child template, follow these steps:

1. Navigate into the configuration for the document template for which you want to add a child template, make sure a configuration task is selected, and then select ‘Add New Child Template’ from the top-most More dropdown menu. A dialog box appears allowing entry of key properties of the new child template.
2. To support a repeating section or basic repeating rows, leave the “Parent Child Template” property as n/a. If you do not see this option, it is because no child template already exists that could be a parent child template. Note that this property is generally only used to support inner repeating rows nested within outer repeating rows or repeating section.
3. Leave the “Child Template Type” as its default value “Standard (Repeating)”. The other option will not support repeating sections or repeating rows.
4. In the “Child Template ID” field, enter a identifier that must be unique across all child templates in the document template.
5. In the “Child Template Name” field, enter a descriptive name for the child template. The name will appear in the user interface for end users, so the name should work grammatically in phrases like ‘Insert New ____’.
6. All other options will be covered later in this section and can be left as the default settings until you are ready to configure them. At this point, click Accept to add the child template.
7. At this point, the child template will appear in the flyout menu, and is in fact already selected. With the child template selected, can add the fields you need to the child template using the “Add Field” dropdown menu. These are the fields that would repeat in either the repeating sections or repeating rows. Note that calculated fields in a child template can reference both fields in the child template and in the main document template.

Creating a Repeating Section: Once you have created a child template, you can create a repeating section based on that child template. You add a repeating section the same way that you add a non-repeating section, except that in the dialog box for adding the new section, you set the “Repeating Section?” dropdown to the child template you just created. When configuring this new repeating section, keep in mind the following points:

- When defining the HTML format for the repeating section, you can reference repeating fields from the child template as well as non-repeating fields from the main document template. However, only the repeating fields from the child template will be editable on that section.
- When defining “section actions” for the repeating section, formulas can reference both the repeating fields from the child template as well as non-repeating fields from the main document template.
- The end user with editing rights to the repeating section can add new “instances” or “copies” of the repeating section as needed. Since the user can view or edit one instance at a time, there is a dropdown menu that allows the user to select a particular instance to view or edit. By default, the instances are labeled by number (1,2,3,...). However, to make it user-friendly, you can specify that a particular field in the child template supplies user-friendly labels for this dropdown. Edit the field’s properties and check the “Child Document Title” property for that field. This property is available for character fields, short text fields, long text fields, keyword fields, and profile reference fields (e.g. staff fields). Other data types are not supported. The property can also be applied to a calculated field for these same data types as long as the formula for the calculated field only references fields in the child template (does not use the dot operator to reference specific keyword table fields or fields in other profiles). Note that you can work around any of these limitations by setting up a character data field and writing to it from the Data Form API, as follows:
 1. Create the character data field, which in this example is named MenuText.
 2. Place it on the form such that it is invisible but still writeable using JavaScript and the Data Form API. The @ modifier is used to do this, i.e. {MenuText:@}. However during development, you can use {MenuText:@"V"} to keep it visible temporarily.
 3. For each field directive that contributes to the menu text, introduce the J modifier to identify a new javascript function you will create in the JavaScript area to set the value of MenuText. The JavaScript function can utilize methods like DataFormAPI.getFieldKeywordTableValues to pull any values out of the keyword table (e.g. Description) and combine that with other fields. Below is an example of such a JavaScript function.

```
function updateMenuText()
{
  var keywordValue = DataFormAPI.getFieldValue('KeywordField');
  var characterValue = DataFormAPI.getFieldValue('CharacterField');
  if (!keywordValue) {
    //If no keyword chosen, then write character field to Menu Text
    DataFormAPI.setFieldValue(characterValue, 'MenuText');
  }
}
```

```

else {
    //If keyword chosen, then write Description column of keyword to Menu
    Text.
    DataFormAPI.getFieldKeywordTableValues('KeywordField',
        function(strFieldName, keywordRow) {
            DataFormAPI.setFieldValue(keywordRow.Description, 'MenuText');
        }
    )
}
}

```

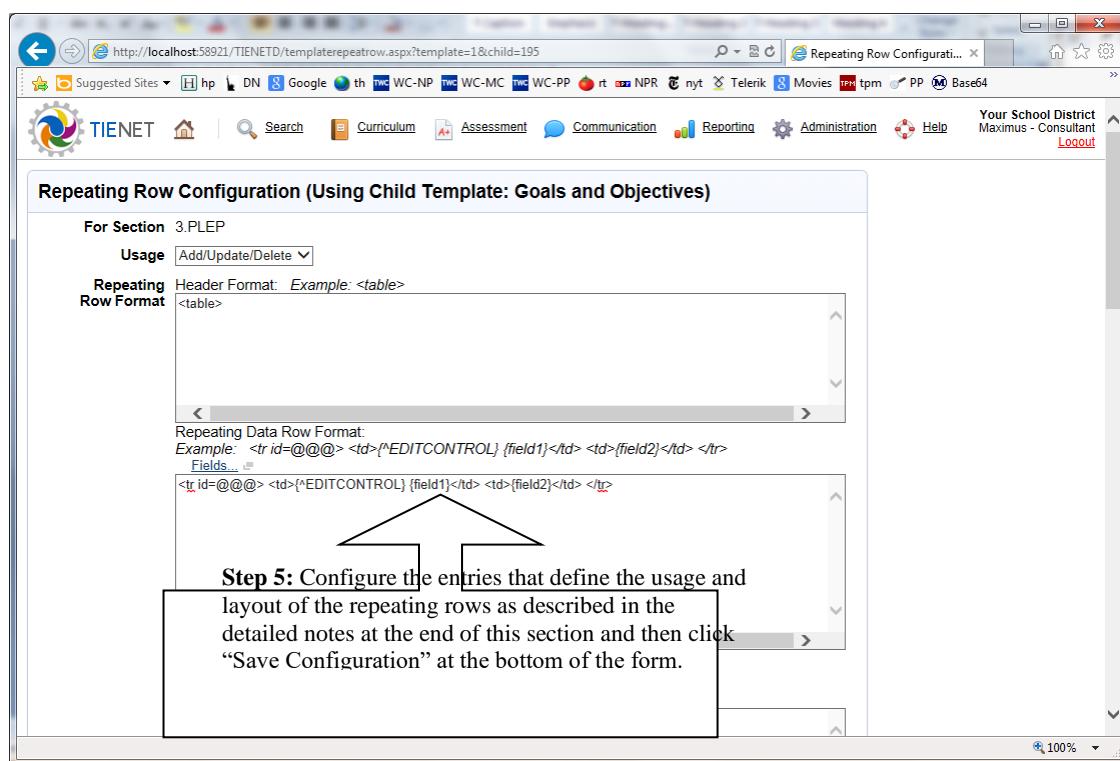
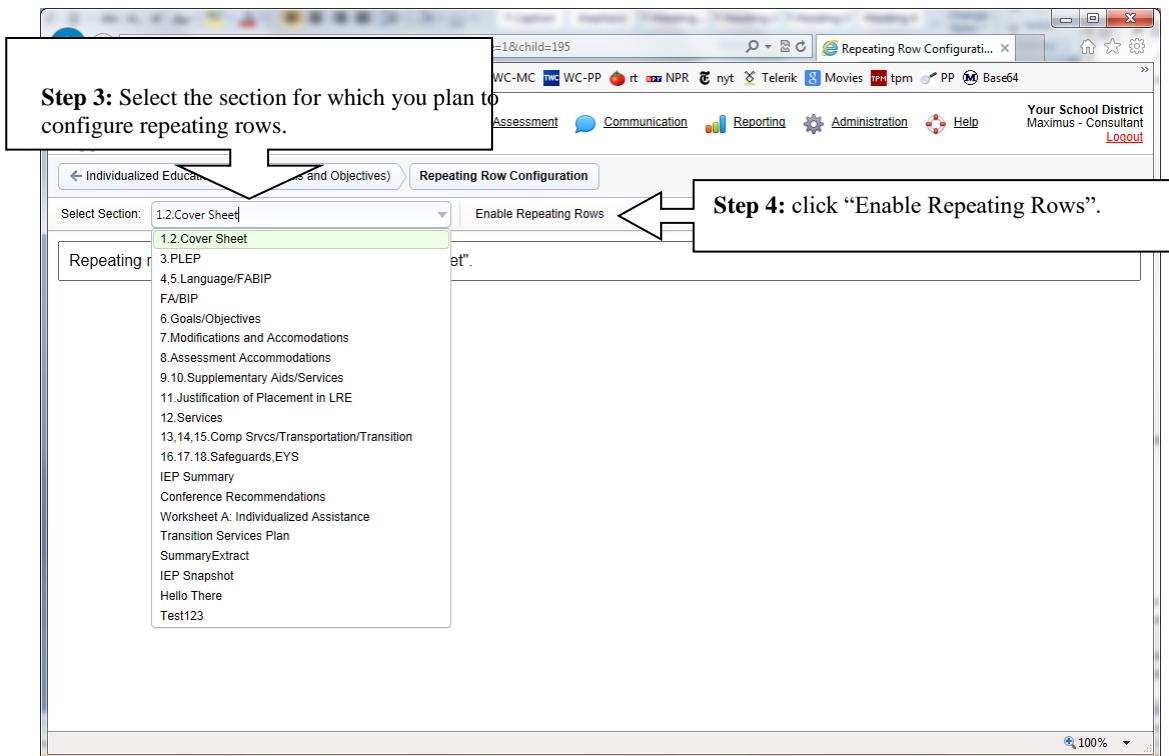
- By default, a user with edit rights to a repeating section will be able to add and delete instances of the repeating section. However, it is possible to “automate” the addition and removal of instances of the repeating section and hide the options that allow the user to do it. First, you would edit the properties of the repeating section and set the “Hide Repeating Section Insert Delete Links” property. You can then develop “Modify Data” section actions that add or remove child documents when executed. When the child documents are added or removed, the corresponding instances of the repeating section are added or removed. Another possibility is to allow the user to add the child documents via repeating rows in a different section, and then the repeating section will automatically be updated to match the repeating rows given that they are both based on the same child template.
- Certain use cases may require you to prevent an otherwise authorized user from deleting a repeating section copy based on fields in the repeating section and/or based on the user profile. You can create a section action with the “Prevent Deleting Repeating Section Copy” trigger to accomplish this.

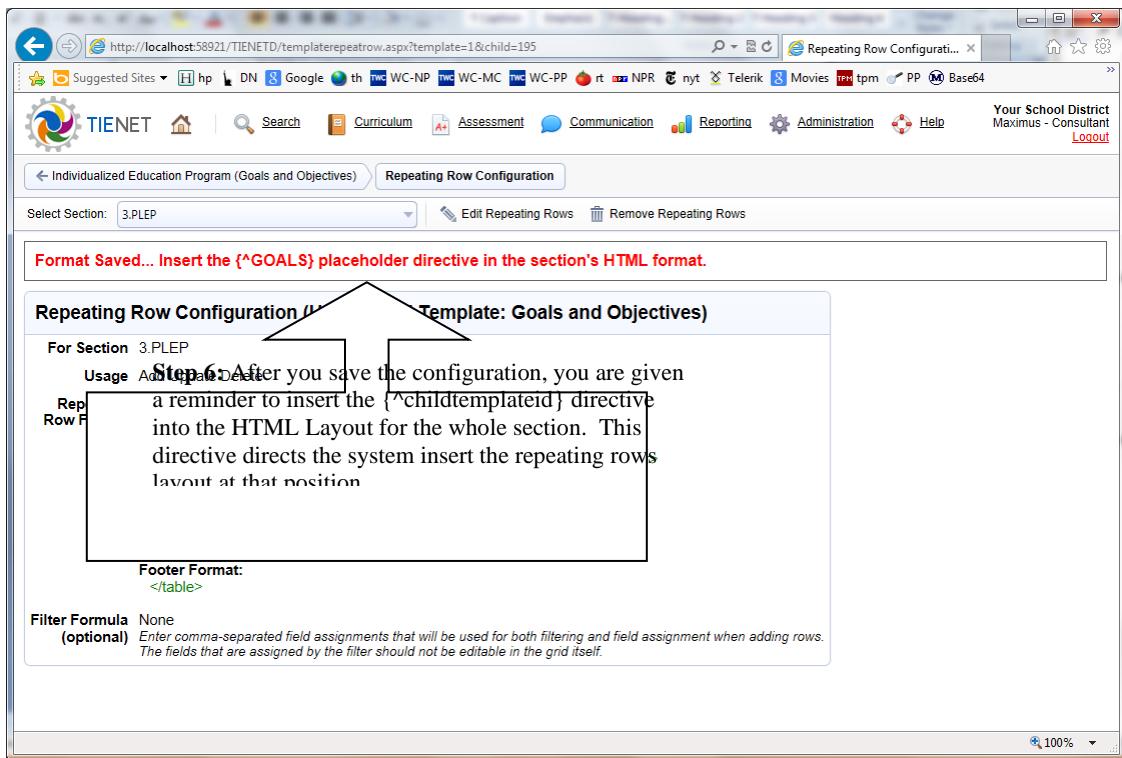
Configuring Repeating Rows: Once you have created a child template, you can configure a repeating rows layout within a particular template section. Follow the steps below to configure repeating rows:

DOCUMENT TEMPLATE CUSTOMIZATION (ADVANCED)

Step 1: Access the configuration screen for the document template for which you wish to define repeating rows. From the flyout menu, select the specific child template for which you want to configure repeating rows.

Step 2: Click Repeating Row Configuration.





The following describes the various configuration fields for configuring repeating rows:

- **Usage:** This option specifies how the repeating row layout will behave when the user is editing the section.
 - **View Only:** With this option selected, the repeating rows will be view only and no fields in the repeating rows will be editable, even in situations when the end-user can edit that section as a whole. This can be useful when the underlying data for the child template is entered somewhere else, perhaps in a repeating section elsewhere in the document.
 - **Update Only:** With this option selected, fields in the various repeating rows can be edited, but rows cannot be added and deleted by the end user who is editing that section. As with “View Only”, this is useful primarily when the repeating data for the child template is initially inserted somewhere else, perhaps in a repeating section elsewhere in the document.
 - **Add/Update/Delete:** With this option selected, the end user can add, update and delete rows while edited that section.
- **Header Format:** This is an HTML format that is rendered at the start of the layout. For example, it might include a `<table>` tag and column headers. Note that this format can reference fields in the main document template, but not the child template.

- **Repeating Row Data Layout:** This HTML format is rendered for each repeating row. For example, it might include table row (enclosed by <tr> and </tr>) and normally references various child template fields. Note that this and other HTML formats in the repeating row layout can reference styles defined in the document template style sheet, and this is in fact advised. If the usage is set to Add/Update/Delete, this HTML format must meet several additional requirements. First, there must be an initial opening HTML tag that has an id of @@@, for example <tr id=@@@>. Without this, the user interface for adding, deleting and moving rows will not work properly. Secondly, the HTML format must include a special {^EDITCONTROL} placeholder directive that specifies where PowerSchool Special Programs should render icons that allow the user to add, delete or move rows.
- **Minimum # Rows:** You can use this to optionally specify a minimum number of rows that should be outputted. If actual number of data rows in a specific document is less than this configured value, then the “Empty Data Row Layout” configuration field described below will be used to render an HTML layout for non-existent rows. For example, if there are five actual rows in a document, and this configured value is 10, then the “Empty Data Row Layout” will be rendered five times. This is primarily used for cosmetic purposes, since a repeating row layout with zero rows might look strange.
- **# Extra New Rows For Adding:** For Add/Update/Delete usage only. Indicates the number of new empty rows to present to the user when in edit mode to allow the user to quickly enter multiple new rows. When the user fills in the empty rows, the user needs to click “Save and Continue” to generate a new set of empty rows unless the “0 (fast mode)” option in this dropdown is selected, which allows the user to simply add new rows as needed with no need to click “Save and Continue”. More information about the “fast add” mode is given later in this section.
- **Empty Data Row Layout:** If there are fewer data rows than the “Minimum # Rows” value specified above, this field specifies the HTML layout to be used to render non-existent rows.
- **Footer Format:** This is an HTML format that is rendered at the end of the layout. For example, it might include a closing</table> tag. Note that this format can reference fields in the main document template, but not the child template. If the “fast mode” described in the description above for “# Extra New Rows For Adding” is configured, the footer format should include an {^EDITCONTROL} directive to indicate where the add row icon will be. You can also supplement the add row (plus) icon with a text label using the L modifier, i.e. {EDITCONTROL:L"Add Item"}.
- **Optional Filter Formula:** This formula, which can reference child template fields and document template fields, can be used to select a subset of data rows to display as repeating rows. However, when the usage is set to Add/Update/Delete, one must instead enter comma-separated field assignments that will be used for both filtering and field assignment when adding rows. The fields that are assigned by the filter, in this case, should not be editable in the grid itself.

- **Sort Formula #1-#4:** These optional sort formula(s), which can reference repeating fields and non-repeating fields, can be used to sort the rows in a particular order, with descending order as an option.

The behavior of the sort formula is very different for Add/Update/Delete repeating row grids than it is for “View Only” or “Update Only” grids.

Sort Formulas for “Add/Update/Delete” grid: In this case, the sort determines the order in which the rows are stored in the database as the user enters them, and if the same data also appear somewhere else in the document, the same ordering would by default be used there as well. Note that if this type of grid has no explicit sort, the end user will be able manually set the order of the rows (using arrow buttons). It is important to note that sorting takes place only when the section is saved, and so specifying a sort formula or changing it will not retroactively impact previous documents.

Sort Formulas for “View Only” or “Update Only” grid: In this case, the sort is applied to the rows while they are being displayed. This means that if the same data appears somewhere else in the document, the sort will have no effect there.

The “Optional Sort Group Header Format” can be used to create sub-headers at each configured level of sorting. The format is inserted into the document whenever the corresponding sort values change. These formats can include data references to the corresponding sort value fields or, more generally, any values that would be the same in all the rows encompassed by the sort group header.

Using the #IF directive in a repeating row grid: The #IF directive will work as expected as long as it does not reference fields in the child template, but if it does reference child template fields, then it will not work as expected in newly added rows that have not been saved yet. The issue is that a PowerSchool Special Programs formula can only be evaluated against real data rows, and the new repeating rows have no underlying data until they are saved. For this reason, it is better to use the #JSIF directive instead when the logic is based on child template fields. But if for some reason you must use the #IF directive, be aware that the #IF directive simply assumes the formula is true in new unsaved repeating rows. This allows for a work-around where you can put the content you do not want in new repeating rows in the #ELSE part of the directive.

Implementing “fast add” mode: This is a special mode of Add/Update/Delete repeating rows only that allows new rows to be added to the grid without postback. This mode is set by selecting “0 (fast mode)” for the “# Extra New Rows For Adding” field in the repeating row layout. To make this mode work correctly, you must configure a {^EDITCONTROL} directive somewhere in the “Footer Format” (described below) to serve as a placeholder for where the “add new row” icon will be. The end user will click this icon to add a new empty row. An example of such a footer format is as follows:

```
{#IFEDIT}<tr><td colspan=3>{^EDITCONTROL}</td></tr>{#ENDIF}
</table>
```

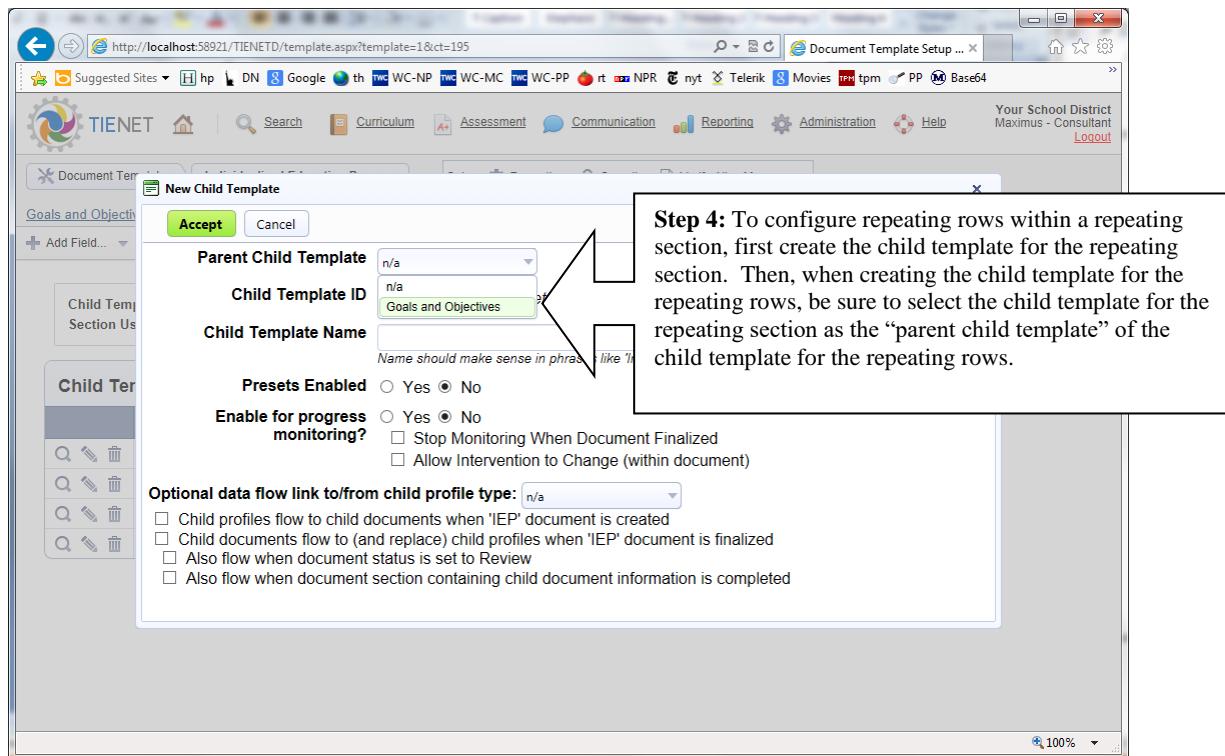
When fast mode is enabled, it is very important that the “Repeating Row Data Layout” be encapsulated within a single HTML tag. If you want the repeating row data layout to have two or more table row tags (<tr>), you can encapsulate the table rows within a single tbody tag. Similarly if you want the repeating row data layout to have two or more paragraph tags (<p>), you can encapsulate the paragraph tags within a single div tag. If this is not done or if there is other malformed HTML in the layout, the add button and/or the shift row buttons will not function correctly.

Implementing a requirement that the user enter at least one repeating row: To require the user to enter at least one repeating row for the section to be considered complete, simply introduce “:R” into the main repeating rows directive, as follows: {^INNERROWS:R}.

Allowing a user to drill down into repeating rows to access repeating sections: If you have a scenario where a document has repeating rows to contain the main details and then later repeating sections for more detailed information, it may be desirable to incorporate “zoom” icons in the repeating rows to allow the user to click into the corresponding repeating section. To implement this, simply introduce “:V” into the main repeating rows directive, as follows: {^DETAILROWS:V}. The zoom icons will always appear except on printed output. If there are certain users who should see the repeating rows but not the repeating sections, it is best not to use this feature since the zoom icons always appear.

Preventing a user from deleting particular repeating rows: Certain use cases may require you to prevent an otherwise authorized user from deleting certain rows based on fields in those rows and/or based on the user profile. You can create an “AllowDelete” logical field to allow or prevent rows from being deleted. Typically this would be a calculated field based on fields in the repeating rows and/or based on the user. If the “AllowDelete” logical field exists, then the delete icon is only shown for repeating rows for which “AllowDelete” is true.

Configuring Repeating Rows within a Repeating Section: You can implement repeating rows within a repeating section. Start by setting up the repeating section with its corresponding child template. Then, when creating the child template for the repeating rows, be sure to select the child template for the repeating section as the “parent child template” of the child template for the repeating rows. Other than that, the process is exactly the same as configuring repeating rows for a non-repeating section.

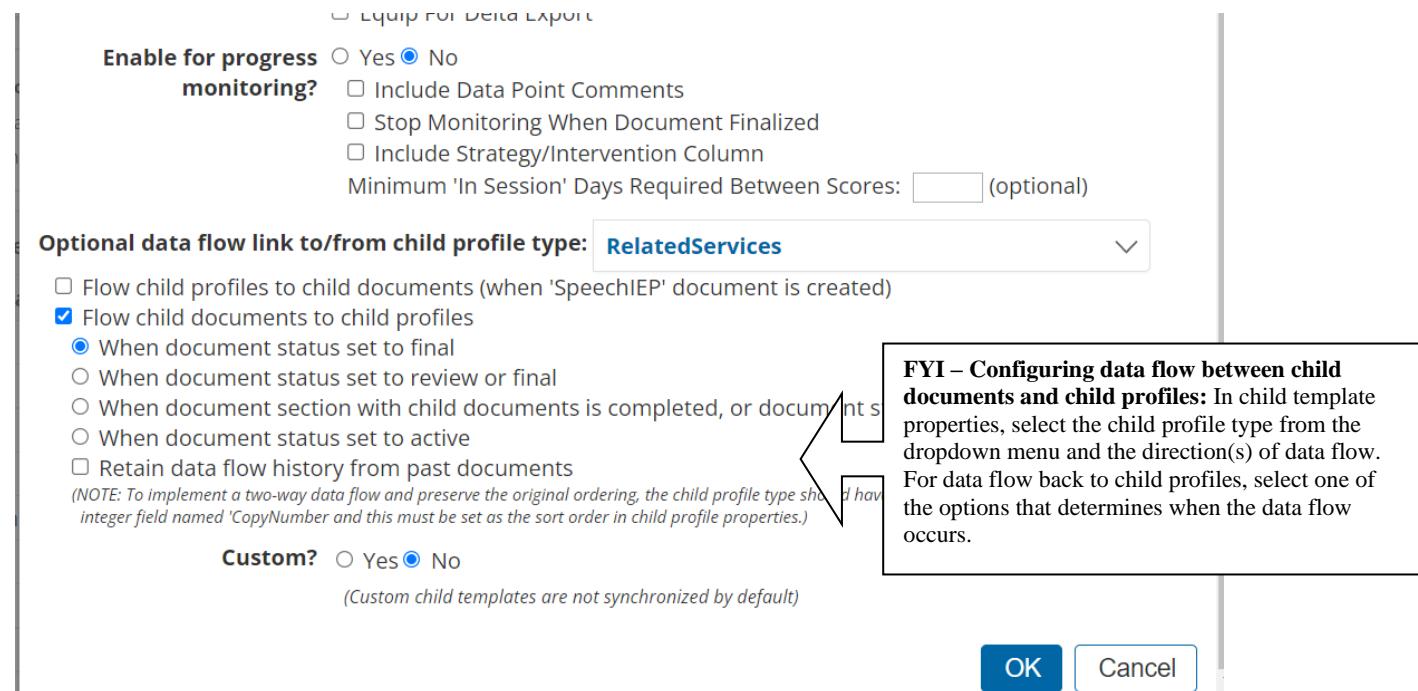


Configuring Repeating Rows that Selectively Introduce Repeating Sections: This describes how to support a scenario where you have 1) repeating rows that support adding and deleting rows, 2) repeating sections based on the same child template that are either view only or update only (no directly adding or deleting repeating sections), and 3) repeating sections should be omitted for some of the repeating rows. This scenario can be supported by introducing a logical field named “IncludeAsRepeatingSection” into the child template which can be either a data field or a calculated field. If a calculated field, you will find it can only reference other fields in the same child template (which helps guarantee that inclusion behaves in a deterministic manner). Also note that if you change the criteria associated with IncludeAsRepeatingSection after documents have already been created, the existing documents will not automatically update to reflect the new criteria. Lastly, you might want to mark the IncludeAsRepeatingSection as an internal value only so that it does not appear on report creation screens.

Nested Repeating Rows: Nested repeating rows (repeating rows inside repeating rows) are configured by defining a repeating row layout for a grand-child template, and then positioning a reference to it (e.g. {^INNERDATA}) in the repeating body format of the outer repeating rows.

Optional data link to/from child profile type: Circumstances may arise where it would be useful to have the data in child documents flow to child profiles of the main profile triggered by specific events such as document finalization, and/or for data in child profiles to automatically populate child documents of a new document that is created. To set this up, simply access the child template properties, select the child profile type and the directions of data flow as shown in the screen below. Note that this kind of data flow relies on fields in the child template and child profile having the same names and data types. If either the name or the data type differs, no data will flow. For fields that have a length (e.g. character field), the

length should also be the same in the child profile and child template. Clicking Verify All will identify mismatches that may exist. Note that it is acceptable for a calculated field to flow to a data field although this clearly will not allow bi-directional flow.



The following data flow options are available in the child template properties screen:

- **Flow child profiles to child documents:** If enabled, any child profiles will flow to child documents when the document is created based on matching field names and data types.
- **Flow child documents to child profiles:** If enabled, any child documents will flow back to child profiles. The timing of when this occurs depends on the sub-option selected, as follows:
 - **When document status is set to final:** Child documents flow back to child profiles only when the document is set to final.
 - **When document status is set to review or final:** Child documents flow back to child profiles only when the document is set to either review or final.
 - **When document section with child documents is completed, or document is set to final:** Child documents flow back to child profiles when the section that contains them (either as repeating section or repeating rows) is completed. The child documents will flow back again when the document is set to final.

(Advanced) There may be certain cases where a child template calculated field flows back to a child profile data field, and the calculated field formula refers to a top-level field

editable in a different section that does not contain the child template. In this edge case, it may be desirable to have the child documents flow back when the section where the top-level field is editable is completed, given that this top-level field will influence the data that flows back. You can accomplish this by enabling the “Trigger Child Profiles Flowback When Referenced Fields Edited” field property of the child template calculated field. Note that this property is only visible when the child template is configured to flow back to child profiles when the section containing the child template is completed.

- **When document status is set to active:** This option will only appear if the document template supports active documents. If this option is set, the child documents will flow back when the document is set as the active document, which occurs only after the document is set to final.
- **Retain data flow history from past documents [New in 21.11.1.0]:** When child documents flow back to child profiles, by default, any existing child profiles are wiped out and replaced by a new set of child profiles copied from the child documents. However, if you enable this property, then only child profiles that may have flowed earlier from the same document are replaced by a new set of child profiles copied from that document. This allows a history to accumulate across multiple documents. Note that prior to version 21.11.1.0, you would need to enable the “Child Document Data Flow History Enabled” property in the child profile type to achieve the same effect, but that child profile property is no longer needed and has been deprecated and removed as of 21.11.1.0.
- **Replace child profiles from original document with those from revision document:** As of 21.11.1.0, this option is only available when the document template supports revision documents and the “Retain data flow history from past documents” property described above is enabled. When this option is enabled (along with “Retain data flow history from past documents”), then when the child template flows back for a revision document, the child profiles for the original document or the previous revision are removed. Without this option enabled, child profiles that flow back from each revision will be retained.

How to limit which rows flow back from child template to child profile type: If a logical data or calculated field named AllowFlowback is added to the child template, only child documents for which AllowFlowback is true will flow back, and so this gives you precise control over which child documents flow back. Note that if child documents with AllowFlowback=true flow back during an initial flowback, but then flowback a second time with AllowFlowback having changed to false, child profiles from these child documents will be removed.

How to enable reflow from child profiles when the document is updated from the profile: Under certain conditions, child templates with “child profile to child document” data flow can be configured to “reflow” when a draft document is updated from the profile at a time later than its creation. At this time, this is not supported for child templates used in any repeating sections. A prerequisite to enabling “reflow” is to configure a profile reference field in the child template that refers to the source child profile type and has the “Child Document Title” and “Read Only” properties enabled. Note that only one field in the child template can have the “Child Document Title” property enabled. After

configuring this field with the required properties, a “Reflow When Draft Document is Opened” checkbox option will appear in child template properties. When this is enabled, the initial data flow upon document creation will automatically populate the profile reference field with the “Read Only” and “Child Document Title” properties so that it refers to the source child profiles. When the draft document is later updated from the profile, child documents that refer to a source child profile will be updated (or deleted if the child profile has been deleted), and any new child profiles will be inserted with the profile reference field populated.

Implementing a Goals Section

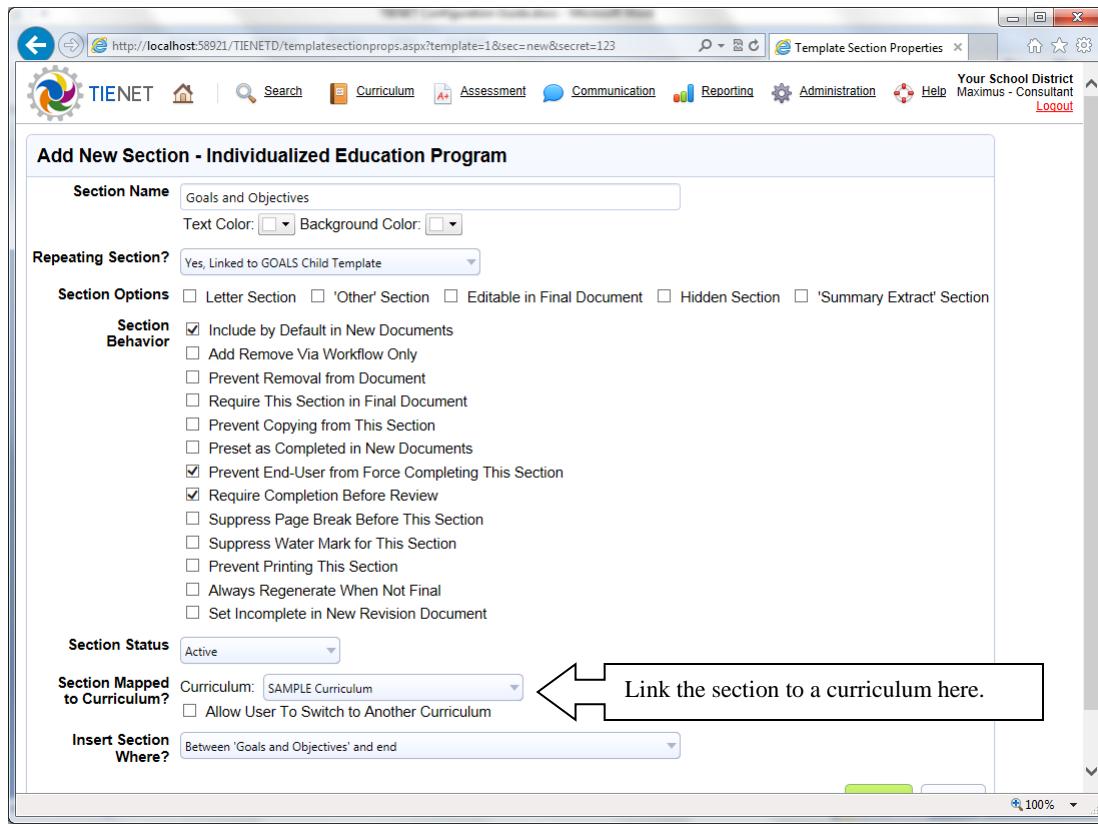
A goals section in an Individualized Education Program or similar plan typically has the following characteristics:

1. Allows entry of multiple goals, and multiple objectives for each goal.
2. Either requires or allows the user to populate certain text fields by selecting or referencing curriculum standards.
3. Supports subsequent reporting or monitoring of student progress towards the goals.

Creating the Goals Section: The first characteristic above is typically accomplished by implementing the section as a “repeating section” with a corresponding “Goals” child template as described in the previous section of this chapter. In most cases, this repeating section will repeat for each goal, although alternative models are supported as well. Multiple objectives for each goal can be implemented as repeating rows within the repeating sections.

It can be beneficial to enable the “Presets Enabled” property of the goals child template. The presets productivity feature allows individual end users to save personal sets of commonly used goals and objectives under a name so that users can insert those presets into documents they are working for. End users can also share their presets with other users.

Mapping the Goals Section to Curricula: To allow end-users to populate fields in the section from the curriculum, the section must be first linked to a curriculum using section properties as shown in the screen shot below. If the section will only be populated from one curriculum, simply select that curriculum in the section’s properties. If the section will be populated from more than one curriculum, then select a default curriculum in the section’s properties and check the “Allow User to Switch to Another Curriculum” checkbox.



The fields in the section must also be mapped to the names of the curriculum outline levels they will be populated from. There are two ways to this mapping.

The “implicit mapping” approach is more useful when the section will only be mapped to one curriculum. In this approach, the mapped fields have the same names as the curriculum outline levels (specifically the singular form). Specifically, you need to add character or long text fields to the child template that have the same name as the corresponding curriculum level (singular form). The deepest level represented in the section can have multiple character or long text fields suffixed with a number. For example, if your curriculum has levels with the singular name of Subject, Goal, and Objective, then you might add long text or character fields to the child template with the names: Subject, Goal, Objective1, Objective2, Objective3, Objective4. Alternatively, you can have repeating rows within the repeating section to implement the objectives, and in this case, you could just have a single field named Objective within these repeating rows. Either way, with implicit mapping, you perform the mapping by naming the fields to match the outline level names.

The “explicit mapping” approach is often required when the section will be mapped to two or more curricula and the curriculum outline levels are named differently in each curriculum. Because the levels are named differently, the implicit mapping approach will not work here. In the explicit mapping approach, a text field can be mapped to any number of curriculum outline level names across multiple curricula. In the directive for each text field to be mapped to the curriculum, you use the F modifier to specify a comma-delimited list of curriculum outline level names (singular form) to which the text field will be mapped. For

example {Standard:F"Anchor Standard,State Standard"} maps the “Standard” field to both “Anchor Standard” and “State Standard”. As a special case, if the level names are prefixed with a tilde character (e.g. F"~Anchor Standard, State Standard "), the curriculum statement labels will be inserted into the long text field instead of the descriptions.

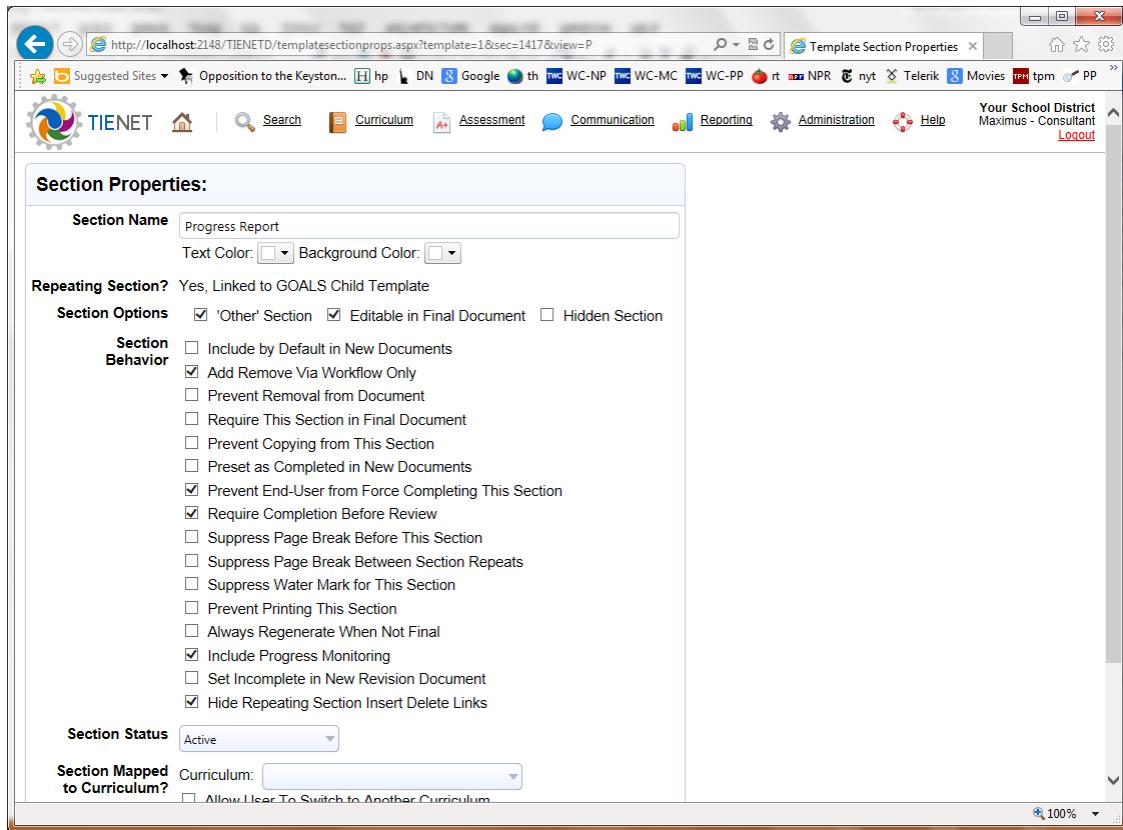
An additional consideration is whether the form needs to dynamically change based on which curriculum is selected. For example, perhaps certain form fields must be presented for one curriculum and different fields must be presented for other curricula. To support this, you will want to have a keyword dropdown on the form that allows the user to pre-select the curriculum on the form itself. The keyword table must provide the curriculum root corresponding to each keyword. This can be done either by making the keywords identical to the curriculum root, or alternatively by adding a character column to the keyword table named “CurriculumRoot” that identifies the curriculum root for each keyword. You will use either #JSIF or auto-postback with #IF statements to dynamically change the form depending on what the user selects from the keyword field. Finally, you will want the user to go directly to the pre-selected curriculum and not be able to switch to another curriculum. To implement this, you introduce the C modifier to the keyword field directive, which instructs the system to bring the user to the pre-selected curriculum when the user clicks the “Select from Curriculum” button. Specifically, you will want to use the C"=" variation, that further instructs the system to not allow the user to change the curriculum in the curriculum popup, and to furthermore prevent the user from bringing up any curriculum until one is selected in the keyword field. An example of the full directive is {ReferenceCurriculum:C"="}.

You can also override the label of the “Select from Curriculum” button. To set the label to “Select from Standards” for example, include the directive below at the top of the section:

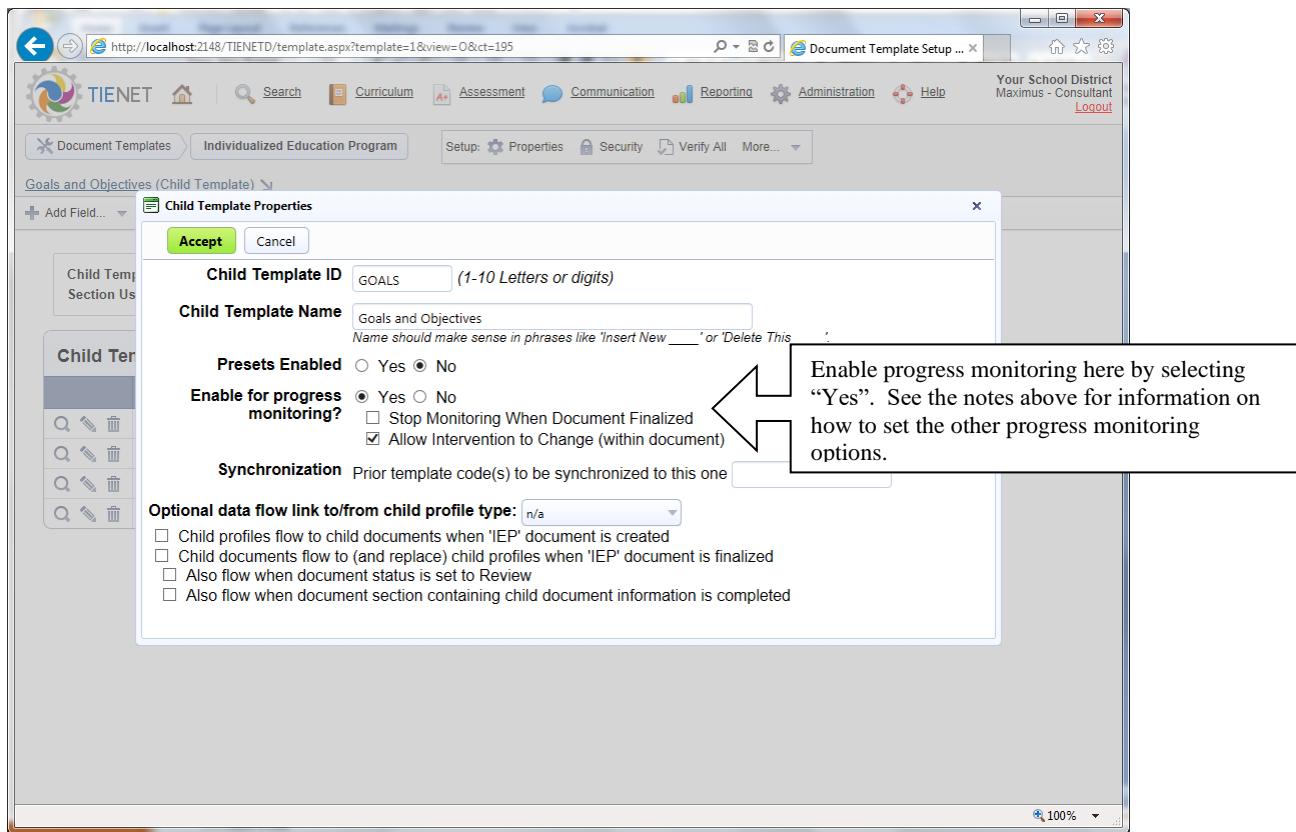
```
{*<SELECTFROMCURRICULUM>:L"Select from Standards"}
```

Configuring Progress Reporting/Monitoring: A progress report is typically implemented as a separate repeating section based on the Goals child template with text boxes to support entry of the narrative progress. There are certain options you will want to set in the section properties for the progress report to make it behave as expected:

- Enable the “Other Section” option in section properties, which essentially establishes the section as an addendum to the main document.
- Enable the “Editable in Final Document” if you wish to allow users to enter progress after the IEP or plan is finalized.
- Make sure the “Hide Repeating Section Insert Delete Links” behavior option is enabled to prevent users from adding or removing goals from the progress report. Goals should only be added or edited from the main goals section.
- Consider turning off the ‘Include by Default in New Documents’ option and turning on ‘Add Remove Via Workflow Only’. Then implement a document action that adds the progress report section when the document is finalized, or some other appropriate time.



To add support for progress monitoring, edit the properties of the goals child template and enable progress monitoring. For IEP goals, the “Stop Monitoring When Document Finalized” check box option should be disabled since progress reporting occurs after the document is finalized. The “Allow Intervention to Change” option should typically also be checked for IEP goals, which allows the end user to modify the intervention over the course of progress monitoring.



Enabling progress monitoring for the goals child template automatically creates the following fields in the child template:

- **UseProgressMonitoring** (logical): Determines whether progress monitoring is enabled for particular goals. If not enabled, then anecdotal/subjective approaches would be used instead. Note that the “UseProgressMonitoring” field can be made editable for every goal, or set to a default value.
- **BaselineDate** (date): When the goal line and progress monitoring in general begins.
- **BaselineScore** (score): The baseline data point that defines where the goal line begins.
- **TargetDate** (date): When the goal line and progress monitoring in general ends.
- **TargetScore** (score): The goal data point that defines where the goal line ends.
- **TrackStudentErrors** (logical): Determines whether the user is allowed to track error rates while progress monitoring. This field defaults to false, and as such, can be ignored or utilized in the configuration as the use case demands. It can be exposed to the user as a document checkbox or its default value can be set to true if student errors should be tracked in all cases.

- UseInitialDataPointAsBaseline (logical): Determines whether the initial data point entered during the progress monitoring process should flow back and populate the BaselineScore field. This field defaults to false, and as such, can be ignored or utilized in the configuration as the use case demands. It can be exposed to the user as a document checkbox or its default value can be set to true if the data point should flow in all cases. If exposed to the user as a checkbox, it is important to use #JSIF or auto-postback to hide the BaselineScore field which would otherwise be a required field.

These fields are generally entered by the user in the main goal and objectives section, but then utilized on the corresponding progress reporting/monitoring section to support progress monitoring. Typical basic HTML formatting for these fields in the goal section is shown below:

```
<tr><td><b>Start Date:</b> {BaselineDate} &nbsp;&nbsp;&nbsp; <b>End Date</b> {TargetDate}</td></tr>

<tr> <td><b>Baseline Data Point:</b> {BaselineScore} &nbsp;&nbsp; <b>Goal:</b> {TargetScore}</td></tr>

{#IFEDIT}<tr><td><b>Initiate Progress Monitoring:</b>
{UseProgressMonitoring:A}<br><br></td></tr>{#ENDIF}
```

To include the progress monitoring chart in the progress report section, you will need to enable the “Include Progress Monitoring” checkbox in the section properties of the progress report. You will also need to insert a {&ProgressMonitoring} directive at the point in the HTML where you want the chart to appear.

In certain circumstances, you may want to stop progress monitoring (stop the collection of data points while continuing to show the already collected data points and the chart). To do this, use a construct as follows:

```
{#IF ProgressMonitoringStopped}
  {&ProgressMonitoring:R}
{#ELSE}
  {&ProgressMonitoring}
{#ENDIF}
```

The {&ProgressMonitoring:R} includes progress monitoring in read only mode only.

Section Extension Child Templates

[New in 23.11.1.0] A document template is limited to about 1024 top-level fields: Normally, this is not a problem, but if a document template has many sections or has been around for a long time, the limit can become a problem. For go-forward model configuration where it is known that this limit will be approached, “section extension child templates” can be used to effectively support an unlimited number of fields that behave like top-level fields. Note that when the platform estimates that there is room for less than 100 additional top-level fields in the document template, it will start showing a warning and a remaining estimate when adding new fields. This warning may be helpful when deciding whether to use section extensions.

At a high level, a section extension child template is a child template that can contain at most one child document/row per top level document, and this is absolutely enforced even by SQL Server itself. Therefore, you can add certain fields to the child template that would otherwise need to be top-level fields with the knowledge that there will be no repetition of the data. Configuring with section extensions will be familiar if you have configured a repeating section as the techniques are very similar, but getting the best results requires a thorough awareness of best practices for using it. At the simplest level, you follow these steps:

1. Add a new child template and set the “Child Template Type” property to “Section Extension (Child Template)”.
2. Add fields to this new child template that might otherwise need to be top-level fields.
3. Create a new section that is linked to this child template in the same way that a repeating section is linked to a normal repeating child template.
4. You can use both top-level fields and section extension fields on the section. **Important:** You will find that both top-level fields and section extension fields will be editable in the section. This is different than a repeating section where only fields in the child template are editable.

Beyond these basic steps, be aware of the following practices:

- Section extension fields do not support data flow to top-level fields or child profile fields. If you need fields on the section that require data flow, simply make those top-level fields and they can be editable on the same section as the section extension fields.
- Section actions and HTML format embedded formulas for sections linked to a section extension can freely mix top-level fields and section extension fields in a manner similar to repeating section. The best practice is that fields that need to be referenced from document actions should be top-level fields. While it is possible to reference any field from any section extension from a document action, this access will be less performant than top-level fields.

- Section extension fields can be more difficult to pull into reports. If one is aware that certain fields will drive document logic or reporting requirements outside of the section extension context, make those top-level fields. Less critical fields can be positioned in the section extension child template.
- Just like a normal section that is not linked to a child template, a section linked to a section extension can be configured to have repeating rows and even nested repeating rows using standard repeating child templates. The section extension child template itself does not repeat and therefore cannot be configured for use in repeating rows, nor can it have grandchild templates of its own.
- Two or more sections can be linked to the same section extension child template, which is particularly useful when those sections have a lot of interrelated logic that reference common fields. On the other hand, one can add multiple section extension child templates and link unrelated sections to them.
- A section linked to section-extension fully supports lookup/non-lookup field combinations; however, both the lookup field and the non-lookup field must be from the same template (either both are from the top-level document template, or both are from the section extension child template).
- A section linked to a section-extension fully supports signing areas configured either DocuSign style or the newer digital signature style. In this type of section, if a DocuSign style directive maps a signer field/role back to a field, the mapped field must be from the same template (either both are from the top-level document template, or both are from the section extension child template).
- If one needs to access field values from a section extension when the formula context is the top-level document, there are ways to do that. Since those ways are less performant than accessing top-level fields, this is to be avoided where reasonable (by using top-level fields for those cases in the first place), but nonetheless situations may surface where such access is required. From the top-level context, one can use the syntax X.Y where X is the section extension child template ID and Y is the name of the section extension field. Note that this same syntax is supported when the formula context is the section extension child template itself, and is no less performant, but is not necessary since the name of the field by itself will do.
- Aggregate formula functions like TopOneValueOf, SumOf, MinOf, and MaxOf are not supported for section extension child templates. The X.Y syntax described in the previous point achieves the same goal for section extension child template as TopOneValueOf does for repeating child templates. However, the Exists function is supported for section extension child templates and in some cases will be more performant than the X.Y syntax. For example, EXISTS(X, Y='Value1' AND Z='Value2') is more performant than X.Y = 'Value1' AND X.Z='Value2'). The reason for this is that the EXISTS function reaches into the section extension to check values only one time.

Additional, more subtle, nuances are as follows:

- Manual copying of document sections from another document by the end-user works the same whether a section is not linked to a child template or the section is linked to a section-extension child template. For example, consider the case where two sections share the same section extension but have different fields from the section extension associated with them. If the end-user copies only one of the two sections, only section extension field values associated with the section copied by the end user will copy.
- If the end-user fills in section extension fields, but then the sections associated with the section extension are removed from the document before the document is finalized, all fields in the section extension will effectively reset their values at the time the document is finalized (by automatically removing the one child document/row for the section extension child template).

One may wonder when to use section extensions versus the ‘Compress Long Text Fields’ document template option which also is useful given the approximate 1000 field limitation. The ‘Compress Long Text Fields’ document template option is more useful for document templates that have been in production for some time and therefore cannot easily be rearchitected to use section extensions. Section extensions are useful for future and upcoming configuration work. Note that there is no mechanism to convert existing fields and child templates into section extension child templates. Such a mechanism was considered, however, the architecture of such functions as digital signature, profile archive and student transfer envelope rely on fields remaining in the same template and not being moved to other templates.

Storing Uploaded Files Associated with Profiles & Documents

There are four approaches for storing uploaded files associated with documents and/or profiles. Each approach has strengths and weaknesses as described below:

1. **File-Based Documents:** Each file-based document is essentially a collection of one or more files. This is the simplest approach to implement since it does not require any configuration other than to assign a few privileges. . However, since it has a number of weakness detailed below, other approaches should be used instead when building state models.
 - Any user who can view a particular profile’s documents via the ‘Access Documents’ privilege can access any file-based documents for the profile. If more fine-grained security is needed, one of the other file storage options should be implemented.
 - A user with access to a file-based document can access the individual files directly from a profile’s document list.
 - Users with the ‘Maintain Own File Based Documents’ privilege will be able to create file-based documents and upload files into them. Users with the ‘Edit File Based Documents From Other Users’ will be able to edit file-based documents created by other users except for those that have been set to final. Users with the ‘Unfinalize File Based Documents’ privilege will be able to set final file-based documents back to draft or review mode, and so this privilege should be restricted to system administrators.

- **Weaknesses:** 1) All such documents are listed in a generic “Other Documents” category rather than a specific category, 2) security is all or nothing in the sense that you cannot allow a security group to access certain file-based documents and not others. Note that the “File Attachments Only Document Template” approach described below was specifically designed to address these weaknesses.
2. **Document File Attachments:** With this approach, files are uploaded and attached to documents from a document template.
- File attachments are enabled in the configuration of a document template by setting the “Allow File Attachments” option in the document template properties.
 - Any user who can access a document can also access its file attachments. However, the ability to upload and attach files is controlled by three document template security rights, namely “Attach Files”, “Edit Files Attached by Others”, and “Attach Files to Final Documents”.
 - Users can see and access the individual files directly from the profile’s document list as well as from within the document.
3. **File Attachments Only Document Template:** This approach combines some of the best features of the two previous approaches and involves configuring a document template that will appear on the dropdown menu for creating a new document. But the subsequent user interface will essentially mimic that for uploading a file-based document.
- To create a document template for file attachments only, set the “Use for File Attachments Only” document template property while creating it.
 - The security rights are identical to that for document file attachments.
 - Document templates of this type can be assigned a document template category. Hence documents of this type can appear in a specific document category other than the ‘Other’ default category.
 - In advanced applications, you may wish to add fields to the document template that data flow from the profile in order to “capture” the values of those fields at the time the document was created. Additionally, you can even create guided actions that get triggered when such documents are finalized. In short, documents of this type can participate in workflow.
 - It is possible to write reports on documents of this type, whereas there is no mechanism for writing reports on simple file-based documents.

4. **File Fields:** You can add one or more file fields to the data dictionary of a document or profile. Each file field stores both the file name and binary image of an uploaded file. This approach has both advantages and disadvantages as shown below.

- You can precisely position each file field on a specific section of a document or profile.
- File fields are very useful when there are multiple independent requirements to upload external documentation for a given PowerSchool Special Programs document. For example, if the user is required to upload one file into one section of a document, and another file into another section of the document, the only reliable way to implement this is to use file fields. With file attachments, you could not be sure what file is associated with what section.
- By default, files stored in file fields do not appear in a document list like file attachments do, in which case the only way to access a file stored in a file field is to navigate to where the file field is on a specific section. However, file fields in document templates have an optional checkbox property labeled “Show File as Read Only File Attachment”. When this property is enabled, the file stored in the field will also show as a file attachment just like any other file attachment with the exception that the file can only be modified or removed via the file field. This property is only available in the context of a top level document template and not in child templates.
- File fields can be configured as required fields.
- When a file field is referenced in a formula, it gives the file name as a character value. A file field be tested within a formula to see whether it has a value yet (i.e. a file) using “FileName IS EMPTY”.
- When using file fields to allow the user to upload more than one file there are limitations to be aware of, and related best patterns and practices as described below.

Use File Fields to Support Uploading Multiple Files:

When there is a need to use file fields to allow uploading multiple files in a document, the standard solution is to add the file field to a child template. The first challenge here is that the file field will only be editable after a child document is saved or otherwise stored in the database. This is not an issue when the child template of the file field is the basis of a repeating section but can be a problem when the file field is in repeating rows, especially in the fast add mode. The issue can also be relevant when the file field is in a child profile. One technique is to simply inform the end user of this using markup like the examples below:

The example below only works in profiles and child profiles.

```
{#IFNEW}
    Files can be attached after saving this item
{#ELSE}
    {FileField}
{#ENDIF}
```

The example below will work in repeating rows under the condition that the repeating rows have editable fields other than the file field.

```
{#IF IDT IS EMPTY}
    Files can be attached after saving this item.
{#ELSE}
    {FileField}
{#ENDIF}
```

To implement repeating rows that only present a file field to the user, follow this approach:

1. First, note that this approach will not work in the file field will be put in a grand-child template to be used in repeating rows nested within repeating rows. An alternative technique to be described later in this section may be applicable to that scenario.
2. Create a child template with the file field plus an additional date/time field with a recommended name of FileDateTimeAdded with a default value of CurrentDateTime() and optionally marked as “Internal Value” in its properties. Set up the repeating rows with the “Input Sort Formula #1” set to the data time field.
3. Set the repeating rows to fast add mode with no minimum number of rows, but do not configure the {^EDITCONTROL} directive in the footer.
4. At this point, the repeating rows will behave like fast add but will not have any way to add rows. You can now add that yourself by creating a section action (perhaps named “Add File”) with a trigger type of “Modify Data Upon Save” and the “Link to Button Directive” option. Place your own button next to the repeating rows to add a file via a {*Add File:L"Add File"} directive. You can optionally specify a CSS class for the button directive with the C modifier.
5. Consider how to handle when the user does not supply a file for a row that has been added. Such empty rows can be removed via a document action when the document is finalized. The file field can optionally be made required, which will force the user to explicitly delete unused rows.

In some scenarios where the requirement is to allow the user to upload up to N files at a particular point on a form, it may make sense to consider the following pattern that leverages JSIF and a finite number of file fields numbered 1 through N in the same record. If you need to attach 1-N

files directly to a child profile, this will be the only option. It may also make sense if you already have repeating rows using fast add but need to allow attaching 1-N files to each repeating row. In this case, the previously described approach will not currently support putting the file field in a grand child template configured for repeating rows. The following configuration in a child profile form will provide the end user with an experience that as soon as they upload a file, the next file field appears up to four files. It also nicely handles the user deleting any one of the four files later.

```
{FileField1:!}
{#JSIF FileField1|FileField2,tag=span}{FileField2}{#ENDIF}
{#JSIF FileField2|FileField3,tag=span}{FileField3}{#ENDIF}
{#JSIF FileField3|FileField4,tag=span}{FileField4}{#ENDIF}
{-FileField2,FileField3,FileField4}
```

If you have repeating rows where it is necessary to allow the user to attach up to N files to each repeating row, the following variation of the above behaves nicely:

```
{#IF IDT IS EMPTY}
    Files can be attached after saving this item
{ELSE}
    {FileField1:!}
    {#JSIF FileField1|FileField2,tag=span}{FileField2}{#ENDIF}
    {#JSIF FileField2|FileField3,tag=span}{FileField3}{#ENDIF}
    {#JSIF FileField3|FileField4,tag=span}{FileField4}{#ENDIF}
    {-FileField2,FileField3,FileField4}
{ENDIF}
```

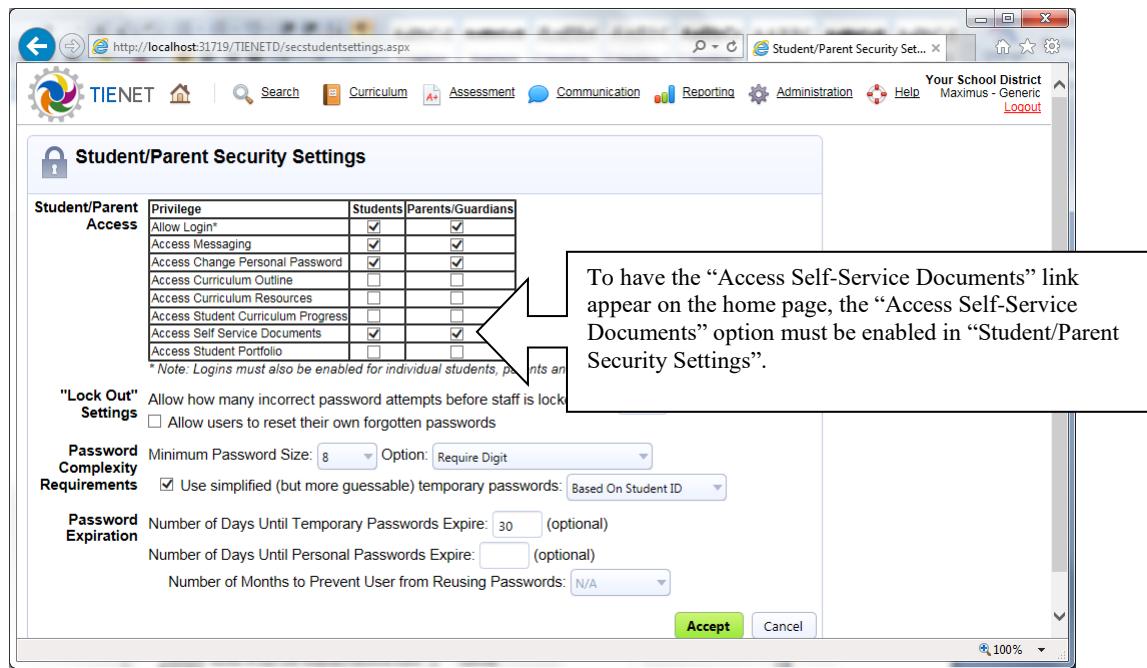
Self-Service Documents

The self-service documents feature can be used in two contexts:

1. It can be used to grant student, parent and guardian users view access, and potentially even edit access, to the corresponding student documents without allowing access to other students' documents.
2. It can also be used to grant staff access to their own documents without allowing access to documents of other staff. However, as of Version 14.1, there is a more robust alternative to staff self-service documents. See the "Document Owner Security" section on page 243 for more information on this recommended alternative.

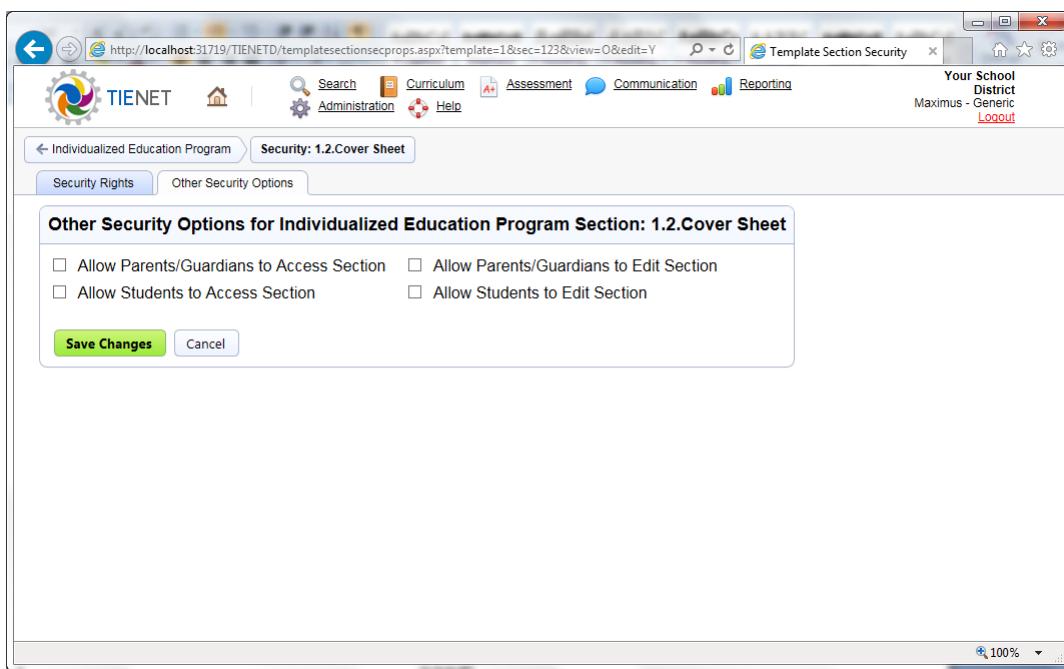
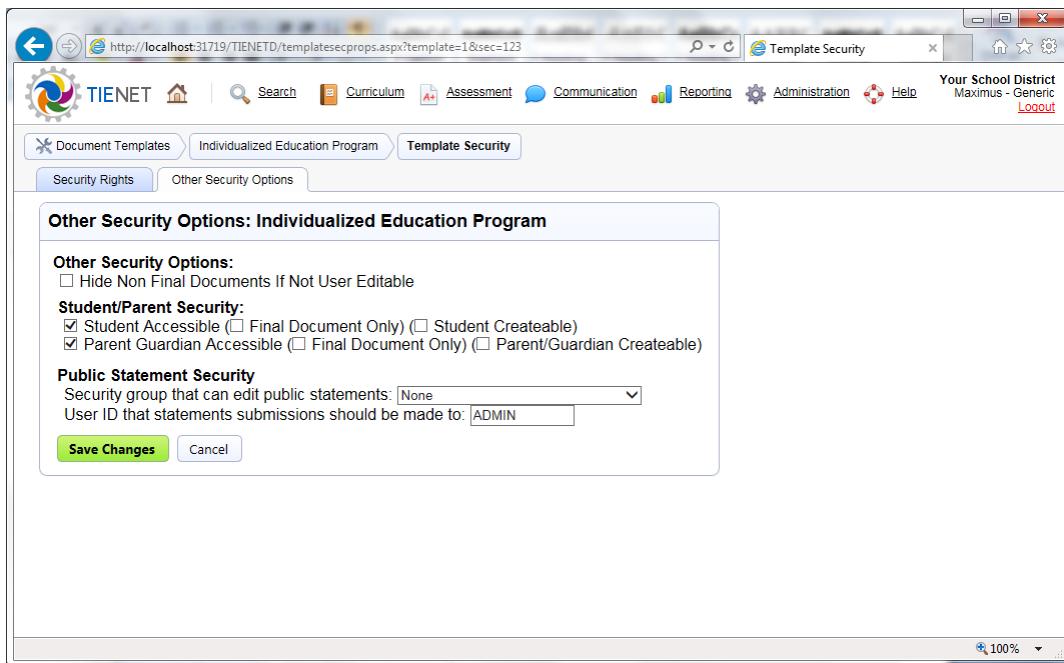
To configure one or more student document templates for self-service access by students, parents and/or guardians, follow the steps below:

- First you will need to make sure that the “Access Self-Service Documents” link is on the home page of students, parents and/or guardians. Select “Security” from the “Administration” menu. Click the “Student/Parent Security” tab and then click “Security Settings”. Make sure the “Access Self-Service Documents” checkbox for students and/or parents/guardians.



- In each document template for which you want to enable student and/or parent/guardian access, go to the document template’s main security screen, and access the student/parent security options on the “Other Security Options” tab as shown below. However, you will also need to enable access to each individual section of the document template as shown in the second screen shot below.

The Student/Parent security options are on the main document template security screen in the “Other Security Options” tab.



3. Of course, you will need to activate the logins of the students and/or parents/guardians for them to be able to access self-service documents. This is described in the system administration guide.

Self-Creatable and Self-Editable Documents: The security options for self-service documents allow for a scenario of students, parent and/or guardians editing documents or even creating documents. Here are some tips for these scenarios:

- Documents, when editing in self-service mode, only support section actions with trigger types of “Modify Data Upon Save” and “Rollback Saving Section”. Document actions are not supported in self-service mode. You can implement a “Modify Data Upon Save” section action to queue the document to be set to final. Documents queued this way do not change status immediately, but rather change status during non-peak hours by the PowerSchool Special Programs background service.
- Note that users of self-service documents can never finalize their own documents. But it is possible to establish a document action to automatically queue a document to be set to final, or to use “mass finalization” approaches at the end of the school year.
- If allowing self-creation of documents, you may wish to establish some controls over the user’s ability to create multiple documents of this type all at once. The “Require Previous Document to be Finalized before New One Created” and the related “Within the Same School Year Only” template options are the easiest ways to establish control.

Staff Self-Service Documents: The procedure to configure staff self-service documents is almost exactly the same as for student self-service documents, but with two key differences. The first is that such document templates would be for the “Staff” profile type and any documents created would be staff documents. The other difference is that to have the “Access Self-Service Documents” link appear on the home page, users must be granted the special access privileges labeled “Access Self-Service Documents”. But as stated earlier in this section, the “document owner security” functionality is a better approach to these scenarios.

Document Template Translations

The main focus of this section is how to translate a document template so that it supports one or both of the following two modes.

Translation: Documents are written in the default language (e.g. English) and then later translated into one or more other languages.

Localization: Documents are written directly in one or more languages other than the default language. Once the document template itself is translated, individual documents are not translated because they are written directly in the intended language.

Before translating document templates, it should be kept in mind that there is a broader checklist of items that must be completed related to this. These are as follows:

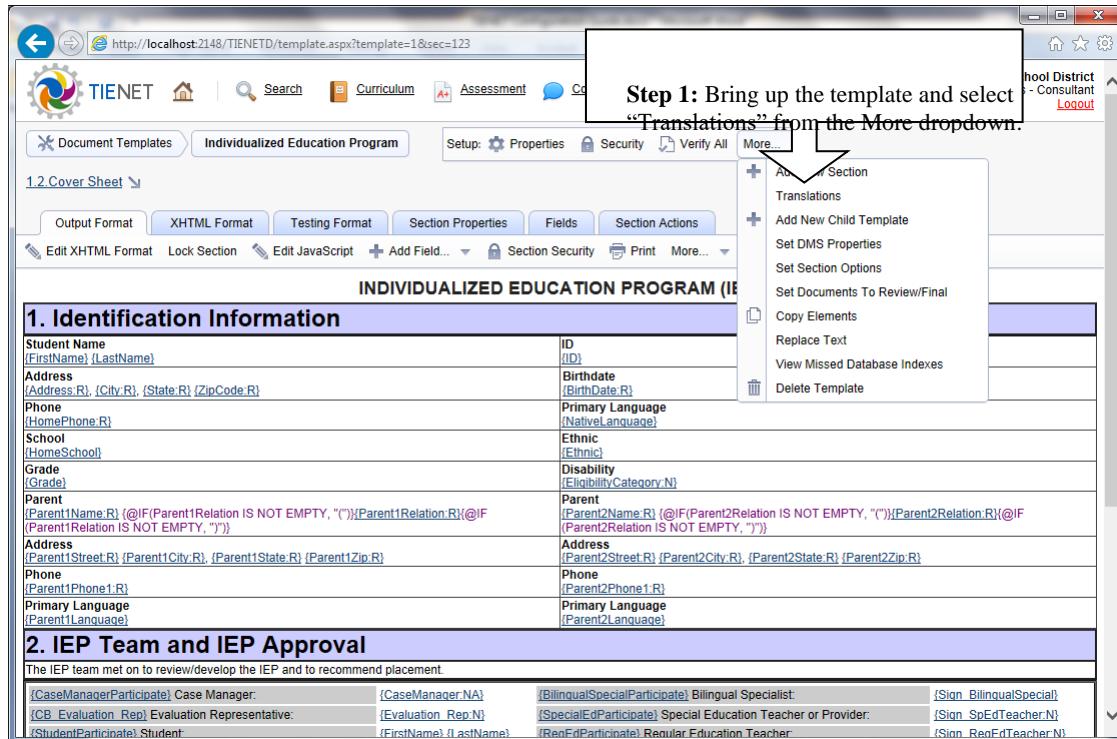
If machine translations are to be supported, the PowerSchool Special Programs servers must be configured with a Google Language Services API account as described in the “Installation and Management Guide”.

If a goals and objectives curriculum will be used, that should be translated as covered in the “Curriculum Development Guide”.

The translated document templates will typically have dropdown menus that utilize text from keyword tables. For example, if the text values are currently being obtained from a keyword table field named “Description”, a new field named “Description_xx” must be added to the keyword table and populated to contain the translated values. In the new field’s name, replace the xx suffix with the ISO language code for the required language (e.g. es for Spanish).

Translation security privileges and document template rights must be assigned to those security groups that will be performing translation on an ongoing basis.

To set up a translation for a document template, follow the steps below:



Step 1: Bring up the template and select “Translations” from the More dropdown.

The screenshot shows the TIENET Document Templates interface. A callout box highlights the "More..." button in the top right corner of the toolbar. A dropdown menu is open, showing various options like "Add New Section", "Translations", "Add New Child Template", etc. The "Translations" option is specifically highlighted with a red box and an arrow pointing to it.

INDIVIDUALIZED EDUCATION PROGRAM (IEP)

1. Identification Information

Student Name (FirstName) (LastName)	ID (ID)
Address (Address.R) {City.R} {State.R} {ZipCode.R}	Birthdate (BirthDate.R)
Phone (HomePhone.R)	Primary Language (NativeLanguage)
School (HomeSchool)	Ethnic (Ethnic)
Grade (Grade)	Disability (EligibilityCategory.N)
Parent (Parent1Name.R) @IF(Parent1Relation IS NOT EMPTY, "()") {Parent1Relation.R} @IF(Parent1Relation IS NOT EMPTY, "()")	Parent (Parent2Name.R) @IF(Parent2Relation IS NOT EMPTY, "()") {Parent2Relation.R} @IF(Parent2Relation IS NOT EMPTY, "()")
Address (Parent1Street.R) {Parent1City.R} {Parent1State.R} {Parent1Zip.R}	Address (Parent2Street.R) {Parent2City.R} {Parent2State.R} {Parent2Zip.R}
Phone (Parent1Phone1.R)	Phone (Parent2Phone1.R)
Primary Language (Parent1Language)	Primary Language (Parent2Language)

2. IEP Team and IEP Approval

The IEP team met on to review/develop the IEP and to recommend placement.

{CaseManagerParticipate}: Case Manager:	{CaseManager.NA}	{BilingualSpecialParticipate}: Bilingual Specialist:	{Sign_BilingualSpecial}
{CB_Evaluation_Rep}: Evaluation Representative:	{Evaluation_Rep.N}	{SpecialEdParticipate}: Special Education Teacher or Provider:	{Sign_SpEdTeacher.N}
{StudentParticipate}: Student:	{FirstName} {LastName}	{RenEdParticipate}: Regular Education Teacher:	{Sign_RenEdTeacher.N}

Step 2: Any existing translations are listed. To add a new one, click **Add New Language**.

Step 3: Choose a language and also provide a translation of the template name. If Google language services has been configured on your server, a machine translation is suggested for you. Click "Accept" to continue and add the language.

Step 4: Click the magnifier icon next to the language to view/edit the template in that language.

Step 5: To edit and/or proof translated phrases that appear in the selected section of the template, click Edit Translated Phrases.

The document template translation contains untranslated phrases.

INDIVIDUALIZED EDUCATION PROGRAM (IEP)

1. Identification Information

Student Name (FirstName) {LastName}	ID (ID)
Address (Address:R) {City:R} {State:R} {ZipCode:R}	Birthdate (BirthDate:R)
Phone (HomePhone:R)	Gender (Gender)
School (HomeSchool)	Primary Language (NativeLanguage)
Grade (Grade)	Ethnic (Ethnic)
Parent (Parent1Name:R) {@IF(Parent1Relation IS NOT EMPTY, "()")}{Parent1Relation:R}{@IF(Parent1Relation IS NOT EMPTY, ")")}	Disability (EligibilityCategory:N)
Address (Parent1Street:R) {Parent1City:R} {Parent1State:R} {Parent1Zip:R}	Parent (Parent2Name:R) {@IF(Parent2Relation IS NOT EMPTY, "()")}{Parent2Relation:R}{@IF(Parent2Relation IS NOT EMPTY, ")")}
Phone (Parent1Phone1:R)	Address (Parent2Street:R) {Parent2City:R} {Parent2State:R} {Parent2Zip:R}
Primary Language (Parent1Language)	Phone (Parent2Phone1:R)
	Primary Language (Parent2Language)

2. IEP Team and IEP Approval

The IEP team met on to review/develop the IEP and to recommend placement.

[CaseManagerParticipate] Case Manager. [CaseManagerNA] [BilingualSpecialParticipate] Bilingual Specialist. [Sign BilingualSpecial]

FYI: Since no translations have been made yet, the form still appears in the default language (i.e. English) but phrases are highlighted in red to indicate they have not been translated yet.

Step 6: You can edit the translated phrases and indicate which ones are proofread using the checkboxes. When editing the phrases, you will need to know some basic HTML tags which are explained in the notes below. Click "Save" to save your work.

Untranslated Phrase	French Translation	Machine Translation
INDIVIDUALIZED EDUCATION PROGRAM (IEP)	Programme d'éducation individualisé (PEI)	Programme d'éducation individualisé (PEI)
Identification Information	Informations d'identification	Informations d'identification
Student Name	Nom de l'élève	Nom de l'élève
{FirstName} {LastName}	{FirstName} {LastName}	{FirstName} {LastName}
ID	ID	ID
Address	Adresse	Adresse

FYI: If Google Language Tools is enabled on your server, machine translations are provided here. You can click the arrow icon below each translation to use it.

Step 7: Click Edit Translation Properties.

FYI: Note that the red highlighting disappears when all the phrases are translated.

Step 8: Provide a translation of the section name and click "Accept". Ignore the other (advanced) options on this screen for now.

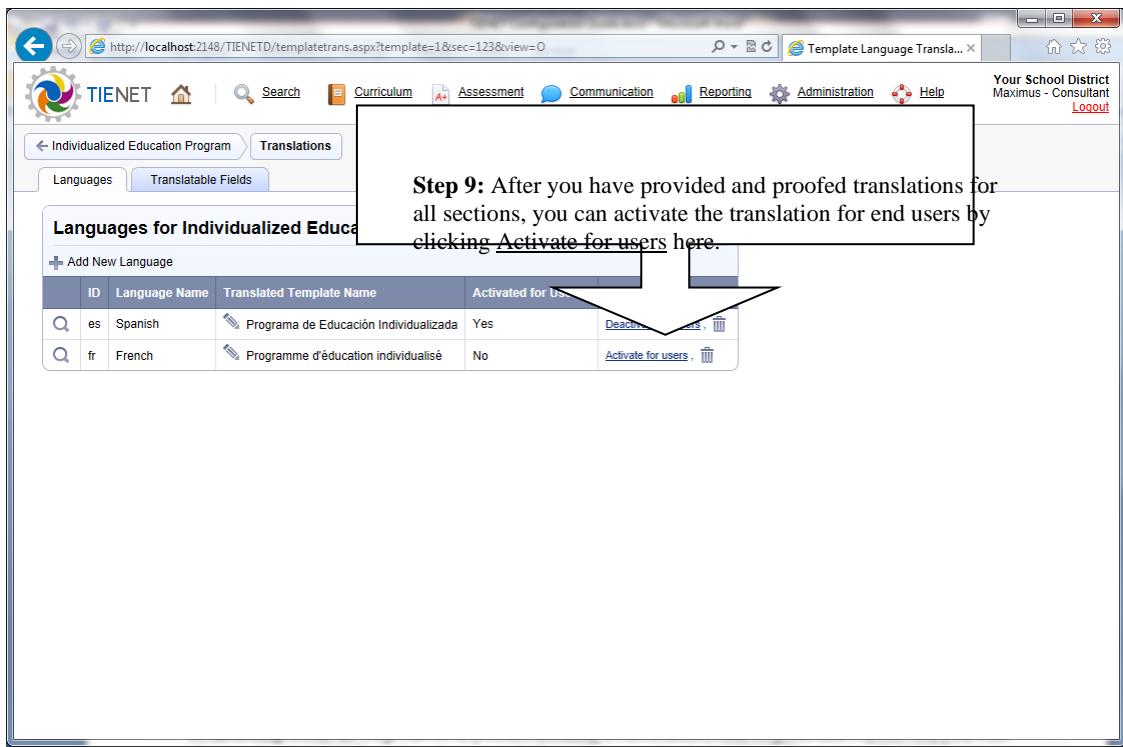
FYI – About HTML Tags: You will notice translated phrases that contain HTML tags. This is necessary to ensure that phrases include enough text to have a sufficient context for translation. Common tags to see

in a translated phrase are given below. You can control exactly which tags are included in translated phrases by clicking the [Edit Translation Properties](#) link available when viewing a template section.

text	Text is in bold face.
 	Line break
text	<i>Text is in bold face.</i>
text	Text span for applying style
<i>italicized</i>	<i>Italicized text</i>
<u>text</u>	Underlined text
^{text}	<i>Superscript text</i>
_{text}	<i>Subscript text</i>

FYI - Designing Templates for Better Translation – It is possible to design the non-translated HTML format of a section in a way that either facilitates or hinders translation. For example, if the intention of the non-translated format is to show the sentence “That boy is smart.” with one word per line, either of the formats below could be used to achieve this. However, using the first format, the sentence will be translated as a complete sentence, while using second format; each individual word will be translated by itself (which will not result in an acceptable translation).

```
<p>That<br>boy<br>is<br>smart.</p>
<p>That</p><p>boy</p><p>is</p><p>smart.</p>
```



Re-Translating a Modified Template: When a template form is modified (in the default language), you will need to provide translations for any new phrases introduced and you can also retire translations of any phrases that are no longer in use. If you select the language as described earlier and look at the form, you will see the new phrases in read. Follow the same steps as shown previously to provide translations for the new phrases. On the phrases translation screen, you will find that the phrases are generally sorted into up to four categories: not translated, translated but not proofread, already proofread, and unused. There is a button to eliminate the unused phrases.

Translating Goals & Objectives: If you are utilizing a curriculum (bank of goals and objectives), you will also need to translate that. Translating curriculums is covered in the “Curriculum Development Guide”. You should also review the material on translating documents in the “Special Programs Users Guide”.

Localizing: At the beginning of this section, it was explained that “localizing” allows documents to be written directly in one or more languages other than the default language. To allow this with a document template, you need to do the following three steps:

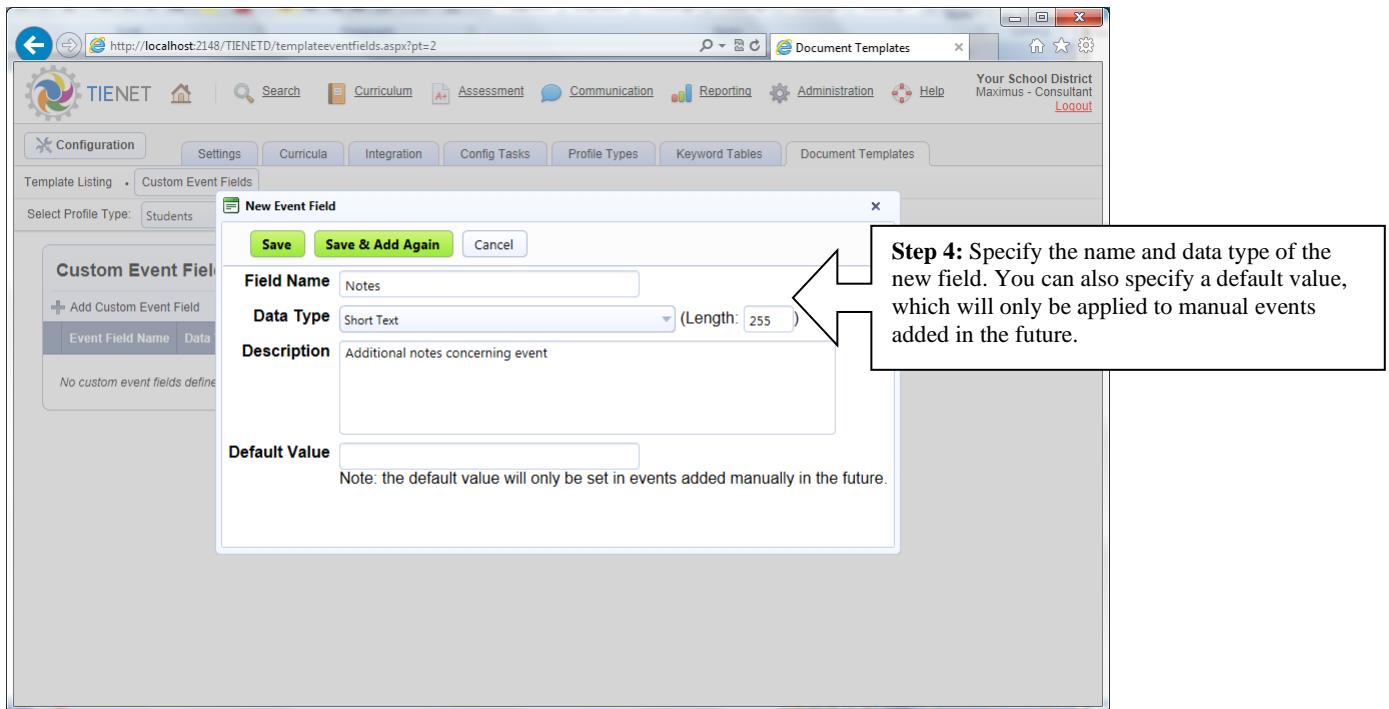
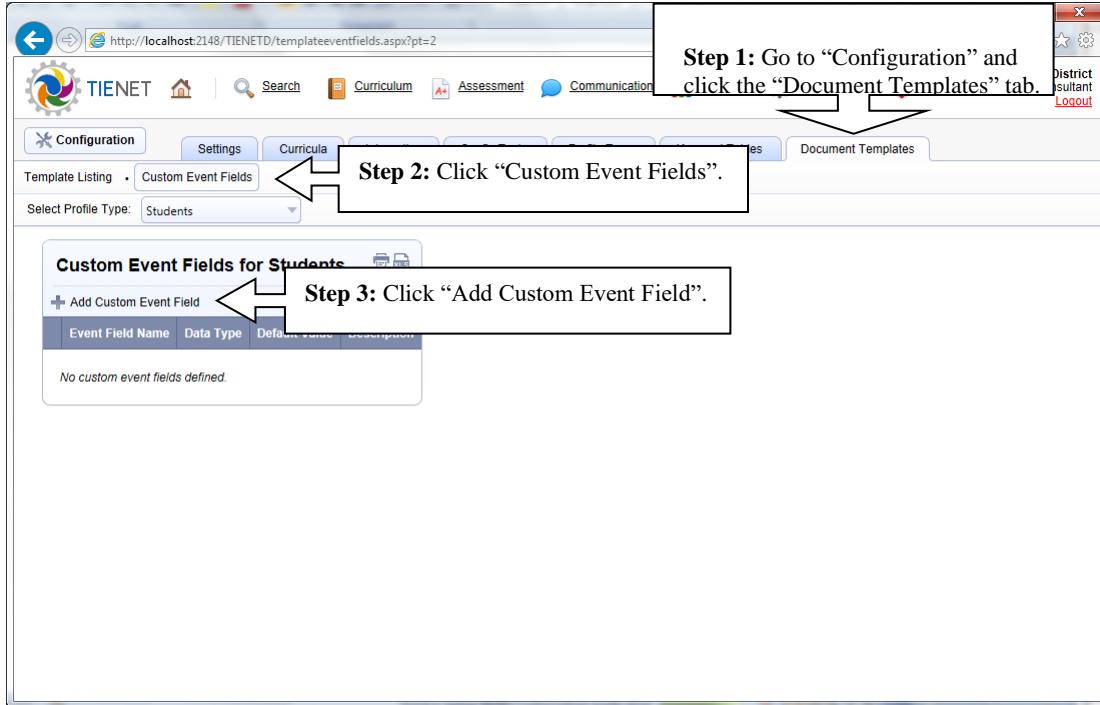
Translate the document template as described previously.

Go the document template properties, scroll down to “Language Options” and enable the “Allow writing documents directly in language(s) into which template has been translated?” option.

You will also need to provide translations for the following other aspects of the document template configuration: document actions, section actions, child template name, and the data dictionary (optional). It is also important not to forget to provide translations for the keyword tables as described earlier.

Customizing Document Events

Custom fields can be added to document events as follows:



FYI – Accessing event data from PowerSchool Special Programs formulas: You can apply the aggregate functions (EXISTS, COUNT, SUMOF, MINOF, MAXOF, TOPONEVALUEOF) to event data from several different contexts as identified below. The filter formulas for the aggregate functions can reference built-in fields (Student, EventDateTime, SourceUserIDName, Subject, Description, Automatic) as well as custom fields.

To access student event data from a student formula, you can use a format like EXISTS(events_, Automated=false and CallType=Phone). Note that events_ (with an underscore suffix) is a special keyword that refers to the event data in this context. This particular example would select students for whom there exist events that are manual (Automated=false) and for which CallType (a hypothetical custom field) is equal to Phone.

To access event data from a document-based formula, you can use a similar format. In this case, you can use “events_” to refer to any events for the student the document is for. Or you can use docevents_ to refer to only events linked to the document.

Customizing the Documents List View with Summary Extracts

The documents list view (i.e. Student Documents List) can be customized by configuring “summary extracts” for document templates that pull a key data elements out of the documents and into the student documents list view. Such customized views are linked to specific document template categories, and so the views appear when the student documents list is filtered to the corresponding document template category.

Since this feature is linked to a specific document template category, you will want to follow the steps below for each document template in the category that will have the customized view.

1. On the document template properties screen, enable the “Show Summary Extract” option.
2. Create a new section named exactly “SummaryExtract”. Enable the “Sub-section of ‘Other’”, “Is Hidden” and “Section is summary extract” options, and link in any data fields that you want to appear in the customized view. It is only practical to include a small number of key fields from the document.

DOCUMENT TEMPLATE CUSTOMIZATION (ADVANCED)

The screenshot shows the 'Template Section Properties' window with the 'Section Properties' tab selected. The 'Section Name' is set to 'SummaryExtract'. Under 'Section Options', the 'Other' Section, Hidden Section, and Summary Extract Section options are checked. A callout box points to these three options with the text: 'Check the "Other Section", "Hidden Section" and "Summary Extract Section" options. The name must be "SummaryExtract".' Other options like Letter Section and Editable in Final Document are unchecked.

Section Properties:

Section Name: SummaryExtract

Text Color: #008000;">#008000 **Background Color:** #e0e0e0

Section Options:

Letter Section 'Other' Section Editable in Final Document Hidden Section 'Summary Extract' Section

Section Behavior:

- Include by Default in New Documents
- Add Remove Via Workflow Only
- Prevent Removal from Document
- Require This Section in Final Document
- Prevent Copying from This Section
- Preset as Completed in New Documents
- Prevent End-User from Force Completing This Section
- Require Completion Before Review
- Suppress Page Break Before This Section
- Suppress Water Mark for This Section
- Prevent Printing This Section
- Always Regenerate When Not Final
- Set Incomplete in New Revision Document

Section Status: Active

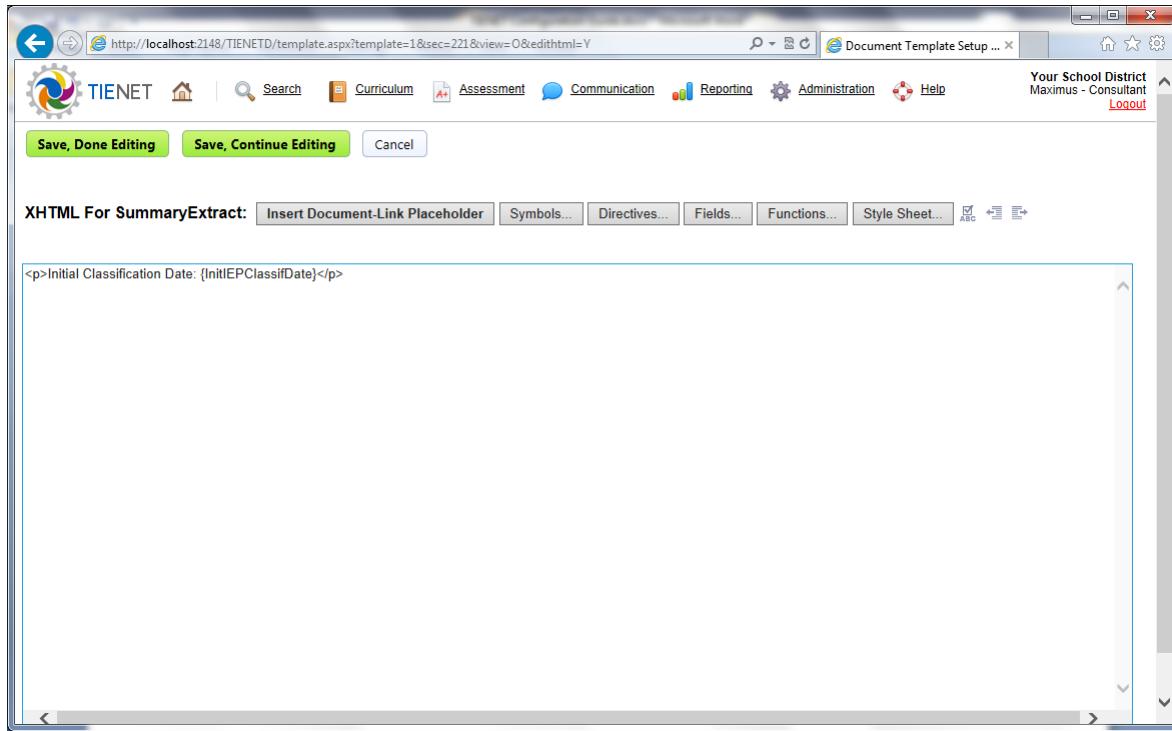
Section Mapped to Curriculum? Curriculum:

Insert Section Where? Between 'Transition Services Plan' and 'Hello There'

Additional Actions: Insert Section Into All Non-Final Documents

3. Define the HTML layout that will be included in the customize documents list view. Note that the layout should be relatively small in size, otherwise it won't be practical to display inside the documents list.

DOCUMENT TEMPLATE CUSTOMIZATION (ADVANCED)



4. Test out the customized view by bringing up a (student) documents list with documents of this category, and then filter the documents list to that category.

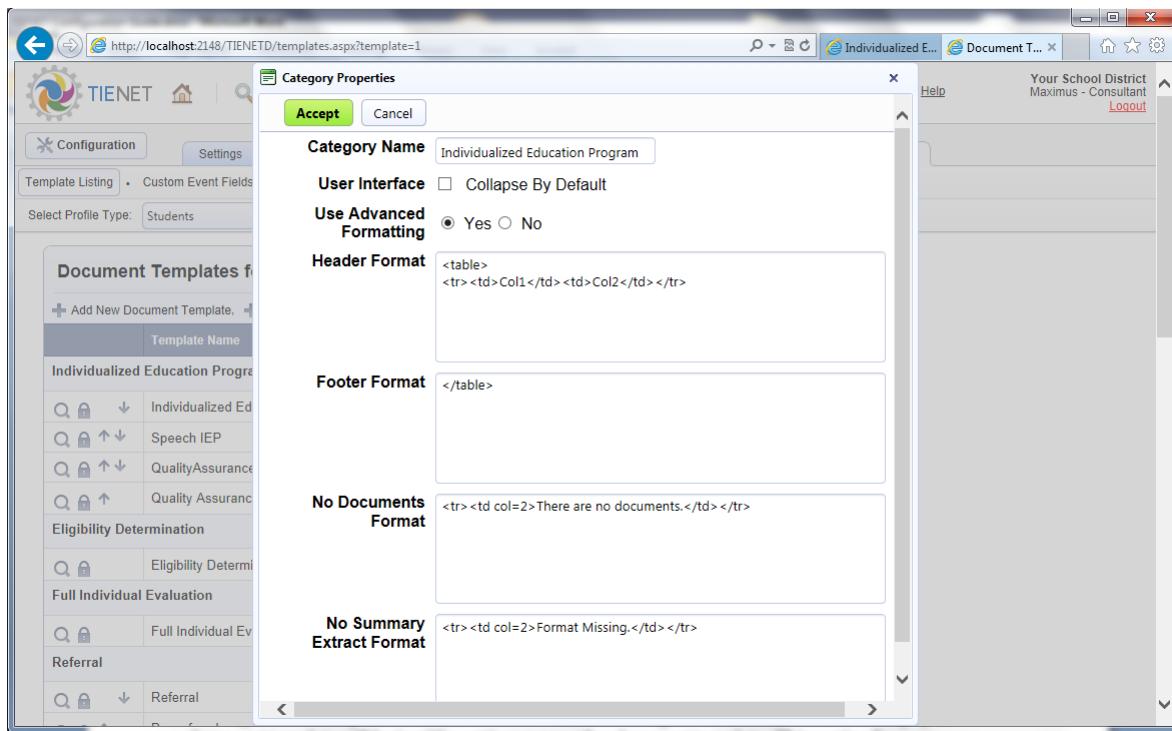
The screenshot shows a web browser window for 'Documents f...' at the URL <http://localhost:2148/TIENET/profileddocuments.aspx?pt=2&profile=91050&origin=Q&cyrs=n&ccts=n>. The page title is 'Document Tem...'. The top navigation bar and user information are identical to the previous screenshot. The main content area shows a list of documents for 'Alfred Daddy (31012705)'. The list includes columns for Profile, Documents, Events, Assessment History, Portfolio, and Security. Filter options for 'By Year' (2009-10) and 'By Category' (Individualized Education Program) are present. A callout box points to the 'By Category' dropdown with the text: 'When the list of documents is filtered to a particular year and category here....'. Another callout box points to the 'Initial Classification Date' field in the table with the text: '...the summary extract from each document, if available, is displayed here.'

Documents for 2009/10	Status	Creation Date	Modification Date	Finalization Date
Individualized Education Program	Draft	07/20/2010 Tue, 07:03 PM	07/20/2010 Tue, 07:03 PM	--
Individualized Education Program				

Advanced Customization: In the example above, the customized layout is included in the existing document list view. It is also possible to take over the formatting of the documents list entirely. The approach is very similar to that used to configure the layout for a repeating rows grid. The main steps are:

Define the summary extract section as previous, but with advanced customization, be sure to include the pseudo-tag {&DocumentLink} as a placeholder for where the document link will appear.

Edit the properties of the document template category, turn on advanced formatting, and define a format for the header, footer, and placeholder formats for when the category has no documents or when a specific document template in the category has no summary extract. A tabular view can be implemented by putting “`<table><tr><th>header</td></tr>`” in the header, `</table>` in the footer, and making the summary extract for each document template an HTML table row.



RtI Intervention Plan Document Templates

The functionality described here is only available if the database is licensed for use with the Response to Intervention (RtI) module. With the assumption that this license exists, this section describes how to configure an intervention plan document for RtI. Generally the document template will be aligned with existing intervention plan forms that the school district is using and most of the work in setting up the document template will be just like setting up any other document template. This section focuses specifically on how to enable progress monitoring and charting to occur directly within the context of the intervention plan document.

DOCUMENT TEMPLATE CUSTOMIZATION (ADVANCED)

Make sure that the template properties described below (that are required for progress monitoring) are set.

Template Properties

Template ID: T1Math (1-10 characters)

Template Name: Tier One -- Classroom Intervention Math

Status: Active

Template Category: RTI Process

Pre-canned document comments:

When creating a new document, the end-user enters a 30-character comment from those you specify below. Enter suggested comments separated by commas.

Template Options:

- Require previous document to be finalized before new one created
- Allow manual copying from other documents from this same template
- Allow manual copying from other documents from other templates
- Allow manual copying from other documents from other profiles
- Automatically update documents from profile without prompting
- Do not prompt user to update documents from profiles
- Disable private statements in this template
- Enable comparison with previous documents
- Allow file attachments
- Use document setup for new documents
- Allow initiation from screening groups
- Equip For Delta Export
- Show Summary Extract
- Print draft/review status in page header

It is helpful to place all the intervention plan document templates in one template category named something like "RTI Process" or "Intervention Plans".

Enable "Allow initiation from screening groups" if you wish this plan to be initiated directly from screening groups.

Event Settings:

Default user comments added to event produced when document set to review status:

Default user comment added to event produced when document set to final status:

Enable for progress monitoring?

Yes No
If yes, monitor progress in: Main Document Progress Report
 Stop Monitoring When Document Finalized
 Allow Intervention to Change (within document)

Revision Documents Enabled?

Yes No

Integrated Meetings Enabled?

Yes No
Meeting Type Name:

Enable progress monitoring here by selecting "Yes" and then choose the "Monitor Progress in Main Document" option. The checkbox options depend on the desired workflow. See the note below for more details.

Click the green "Accept" button to save your options.

About progress monitoring options: When enabling progress monitoring as in the figure above, there are several checkbox options that determine the user process flow. Typically a decision about the student will be made as the result of progress monitoring, and finalizing the document should lock in that decision,

potentially triggering conditional document actions that launch the next step in the process. In this case, check the “Stop Monitoring When Document Finalized” option to prevent the user from entering more data points once the document is finalized. Another choice is whether to allow intervention changes within the same intervention plan, or whether to alternatively require a new intervention plan document to be created when there is a change in intervention. The “Allow Intervention to Change (within document)” controls this choice.

Enabling progress monitoring automatically creates five critical fields that should be linked to a section, namely BaselineDate (date), BaselineScore (score), TargetDate (date), TargetScore (score) and UseProgressMonitoring (logical). The dates and scores define the goal line. The UserProgressMonitoring field allows progress monitoring to be turned on/off in individual documents since it may be desired that anecdotal/subjective approaches be used in certain situations. The “UseProgressMonitoring” field can be made editable in every document, set to a default, or set by a guided action. Configure the section in which you want to place these fields. Typical HTML formatting for these fields is shown below.

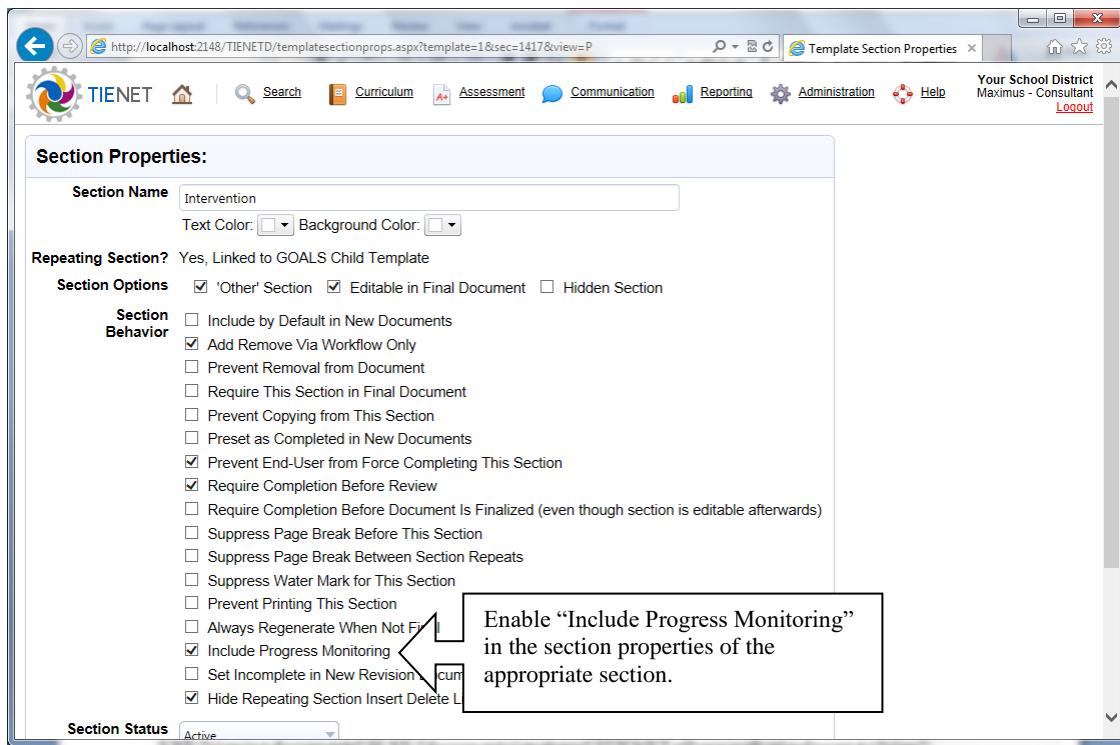
```
<tr><td><b>Start Date:</b> {BaselineDate} &nbsp;&nbsp;&nbsp; <b>End Date</b> {TargetDate}</td></tr>{#ENDIF}{#IF ImprovementMath=True}<tr><td> <b>Extended End Date:</b> {ExtendedEndDateMath}{#ENDIF}</td></tr>

<tr> <td><b>Baseline Data Point:</b> {BaselineScore} &nbsp;&nbsp; <b>Goal:</b> {TargetScore}</td></tr>

{#IFEDIT}<tr><td><b>Initiate Progress Monitoring:</b> {UseProgressMonitoring:A}<br><br></td></tr>{#ENDIF}

{#IFVIEWAND UseProgressMonitoring=TRUE}<tr><td><b>Progress Monitoring Results:</b> {&ProgressMonitoring}</td></tr>{#ENDIF}
```

Indicate which document template section will contain the progress monitoring data points and chart. Typically this is the same section that contains the critical fields described in the previous step. To specify the section, go to the section, click Edit Section Security and enable the “Include progress monitoring” option. By default, the progress monitoring data points and chart will be appended to the end of the section. However, you can optionally define precisely where the progress monitoring data points and chart will appear by inserting a {&ProgressMonitoring} placeholder in the HTML format (as in the last line of the sample HTML above).



Configuring Bulk Operations on Documents

This section describes how to configure scenarios where users can process multiple documents at once (create, print, apply pre-defined modifications).

- There is a new document template option labeled ‘Allow bulk document creation’. If enabled via Document Template Properties, users authorized to create such documents can go to Students Search > Utilities > Bulk Document Creation to access a utility that allows them to select the document template and a set of student profiles for which to bulk-create documents. Note that the ‘Allow bulk document creation’ option cannot be enabled for a document template if the ‘Use Document Setup for New Documents’ option is enabled.
 - Note that if you want to link this to a bulk printing operation, add a date or date/time field named "DocumentPrintedOn" to the document template. PowerSchool Special Programs will automatically populate a field with this name when the document is rendered for printing. Then you can create a list report of documents that need printing using the formula "DocumentPrintedOn Is Empty" and users can bulk-print from that report.
 - Also, note that it is possible now to set up section actions that trigger upon being rendered for printing, and therefore can queue the document to be set to final.

- It is now possible to enable users to apply pre-defined data modification operations on documents via a list report. This can be used to implement specialized scenarios such as a bulk document approval process. To configure this, follow these steps:
 - Configure a document action with a “triggering event” of bulk operation and that performs the desired data modification on the document.
 - Set up a list report that presents the profiles or documents for which users should be able to apply the pre-defined data modification operation. Then go to the report properties of the list report and enable the “Report if under ‘Configuration Management’ option.” Then look for the “Enable Users to Apply Data Modification Constraints via This Report” option (under ‘Output Options’) and enable it too. This second option will only appear if an eligible document action has been created.
 - Make sure that the security for the list report only grants access to staff users who should be able to execute the bulk action(s). This is your primary way of controlling who can execute bulk actions. If you link the document action to a security group, that will also restrict who can execute that particular bulk operation.

Ownership/Category Public Report -> Optional Category: Case Manager Reports
 Report Is Under "Configuration Management" Control

Data Refresh Options Auto Refresh System-Wide Report View Overnight, Also At: N/A
 Data can be as old as: 1 Hour (Not applied to auto-refreshed system-wide report view)
 Allow Manual Refreshing
 Execute on Mirror Database

Output Options Allow Generate As PDF
 Allow Users to Render Report Directly On Their Home Page
 Highlight Report Name in Report Menus Using Color:
 Enable Users to Apply Data Modification Operations via This Report
 Disable Floating Column Headers

Editing Control Options Hide Data When Editing Report
 Allow Users To Make Private Copies of Report

Accept **Cancel**

Master/Detail Documents

This feature enables the configuration of a special relationship between a “master” child template and a “detail” document template. An end-user editing a “master” child document through repeating rows or repeating sections can initiate the creation of a “detail” document by clicking a button (configured in the HTML format for the repeating row or section), or alternatively this can be automated programmatically through a document or section action. The “detail” document is bound to the “master” child document, enabling data flow back and forth between them. If the master document is deleted or undeleted, then corresponding detail documents are deleted or undeleted at the same time.

At the configuration level, the constraints are as follows:

- Only one “detail” document template can be created for a given “master” child template.

- Creation of the “detail” document template is initiated through the configuration of the “master” child template, although from that point forward, the configuration of the “detail” document template can proceed just as with any other document template.
- PowerSchool Special Programs will not allow the “master” child template to be deleted unless the “detail” document template is deleted first.

At the data level, the constraints are as follows:

- At most one detail document can be created for each master child document, and therefore a master child document will have zero or one detail documents bound to it.
- If the master child document is deleted for any reason, the detail document is automatically deleted. However, the user interface prevents master child documents with detail documents from being manually deleted through the repeating section or repeating row user interface. If the detail document is deleted directly, the corresponding master child document is no longer bound to a detail document and therefore can be deleted through the repeating row or section user interface.

The master child template will support special button directives in repeating row or section layouts to create or view the corresponding detail documents. The directives can be placed inside of #IF conditional directives for additional control over when they appear.

- The create button for a child document will only be visible if the corresponding detail document does not exist yet. The directive must reference a “pseudo-action” named “CREATEDETAIL” and will generally take the form {*CREATEDETAIL:L "Create Detail”}. Note that “CREATEDETAIL” is a pseudo-action and not an actual section action or document action that you need to configure. The directive also supports CSS class and other attributes just like a button directive. Creation of the detail document via this button will operate even if the user has no editing rights to the detail document, although such a user would not be able to edit the detail document that got created.
- The view button for a child document will only be visible if the corresponding detail document exists. The directive must reference a pseudo-action named “VIEWDETAIL” and will generally take the form {*VIEWDETAIL:L "View Detail”}. It also supports CSS class and other attributes just like a button directive. The button assumes the user has at least view access to the detail document given that the user has access to the master document. This means that the view button does not currently support the scenario where a user can access the master document but has no view rights to the detail document.

When a detail document is created, data will flow to the detail document using the normal data flow options that are available when a workflow document is created. In addition to this, a detail document template supports several specialized data modification methods (in document and section actions) that can pass data back and forth between the master child document and the detail document. Note that these special modification methods are configured in the detail document and therefore take the perspective of the detail document.

- **Set Fields from Master Child Document:** Writes fields in the detail document using values from master child document. The trigger for this is typically the creation of the detail document.
- **Update Master Child Document:** Writes fields in the master child document using values from the detail document.
- **Insert Master Child Document:** Inserts a new master child document using values taken from the detail document. Note that the new master child document will initially not have a detail document linked to it, so once the document or section action is executed, there is no link between the new master child document and the current detail document that initiated its creation.

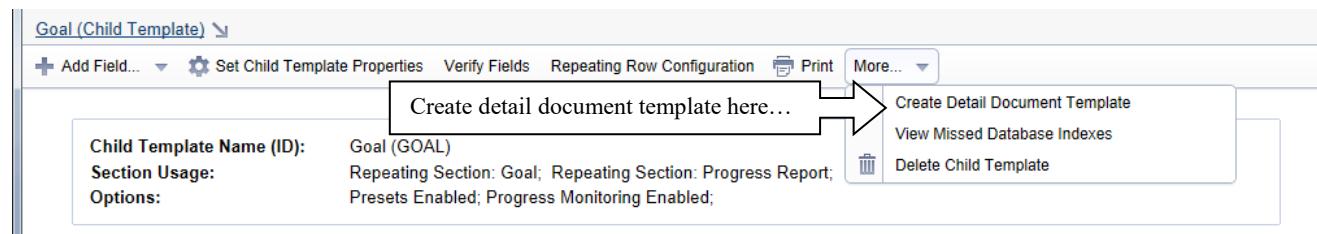
Additionally there is a special modification method that can be configured in the master document template as follows:

- **Create Detail Documents:** Creates detail documents for the master child documents that meet the specified criteria (and that do not already have a detail document).

Since detail documents are not created manually, it may be useful to automatically set the comment of detail documents using a document action.

An example scenario where the master/detail document feature is very useful is when activities need to be assigned to various staff in a master document, and then detail documents are launched for each activity so that the assigned staff member can record details related to that activity. The “document owner security” model can be used to make sure staff can only see their own detail documents and not those assigned to other staff. The document and section actions described previously can capture data entered into the detail documents and move them back into the parent.

To get started, go to the configuration of the document template, select the child template from the section flyout menu, and then select “Create Detail Document Template” from the “More” dropdown menu.



Language Localization

Localization, by definition, is the process of preparing a system to appear in a non-default language (e.g. French, Spanish, etc). Localizing PowerSchool Special Programs requires consulting with PowerSchool to make sure the user interface itself is localized for the desired language, and then translating all the relevant configuration: forms, templates, document actions, section actions data dictionary, etc. Translation of the configuration can be done through the configuration user interface by an authorized user.

The “System Administration Guide” describes how to translate document templates into a non-default language so that documents originally written in the default language (i.e. English) can then be translated into that language. Although “localization” also requires document templates to be translated, “localization” as described here assumes documents will be written directly in the non-default language instead of being translated.

To localize the configuration of PowerSchool Special Programs, perform the following high level steps:

For each keyword table used for keyword selection fields or multiple value fields, make sure there is either a ‘Name’ or ‘Description’ character column containing default language text. Then add an additional character column with the name ‘Name_languagecode’ or ‘Description_languagecode’ where *languagecode* is the two-character ISO code for the language (e.g. es for Spanish, fr for French, etc). Populate this column with the translations.

Translate the document templates as described in the “System Administration Guide”. Then for full localization, do the following:

1. Enable the ‘Allow writing documents directly in language(s) into which template has been translated?’ language checkbox option in the template options for each document template.
2. Translate the name of each document template and document template category.
3. Translate the name of each section within each document template.
4. Translate all document and section actions, and optionally the captions for the document data fields.
5. For each profile type, translate all section names, HTML formats, constraint messages, profile field captions and the caption for the profile type itself.
6. Translate relevant standard reports (list reports and multi-dimensional reports). Translating advanced reports is not supported at this time, but a workaround is to duplicate the desired sections inside of the existing advanced report and localize just those duplicate sections into the target language.

Configuring DocuSign Integration

PowerSchool Special Programs allows configuration of DocuSign integration for specific document templates that require signatures, such as parental consent documents. DocuSign configuration for document templates can lie dormant in a customer’s database until a DocuSign account is configured by the customer. This allows a state/province model database to be delivered fully ready for DocuSign, and then activating the integration requires only establishing a DocuSign account.

Only finalized documents can be sent to DocuSign for e-signature by signers. At the time of submission, each signer is identified by name and email address. The signer data is supplied to the DocuSign service via its API along with the finalized document in the form of a PDF file containing embedded “signing

tabs". Note that DocuSign itself refers to the package containing the signer data and PDF as an "envelope". The document (or "envelope" as referred to by DocuSign) initially has a status of "sent". But the status will ultimately change to "voided" if canceled by the sender, "declined" if rejected by any one signer, or "completed" when all signers have signed the document. The completed PDF (signed by all parties) is stored back in PowerSchool Special Programs as a file attachment to the document for easy access.

One can configure document templates to be integrated with DocuSign with or without a DocuSign account in place. The DocuSign account is only needed for testing the configuration, but since testing is always recommended, a DocuSign account can be established by selecting "Configuratn" from the "Administration" menu, clicking the "Integration" tab and then "DocuSign Integration".

Step 1: To configure a particular document template for DocuSign integration, log in and access that document template's main configuration screen. Select a configuration task and then select "Configure DocuSign E-Signature" from the upper "More" dropdown menu. As shown in the screen shot below, you are now prompted for DocuSign email boilerplate text that will go out to signers, and an optional number of days from sending to expiration after which pending documents sent out for signatures will automatically be voided if signatures are not completed by then. Accepting this configuration is technically sufficient for the system to recognize the document template as "DocuSign integrated" and, if a DocuSign account is also configured, to show authorized end-users the option to submit documents for e-signature. However, more steps are needed to make the integration fully functional, and these steps are covered next.

The screenshot shows the 'Configure DocuSign E-Signature: Individualized Education Program' screen. It includes fields for Subject (containing placeholder text like '{FirstName} {LastName}') and DocuSign Email (containing a placeholder for a Blurb). A large text area for the Blurb is highlighted with a blue border. Below these, there is a 'Days Until Expiration' field set to 20 and a 'DocuSign Options' checkbox labeled 'Allow Submit Review Document'.

Step 2 - Configure staff signer role(s): Each "staff signer role" is based on a staff field in the document. When in operation, the DocuSign integration identifies who the potential staff signers are for a document by extracting staff from any staff fields in the document template, or any child template of the document template, that are marked with the 'Require Signature' field property. So a necessary step is to set this property in the appropriate staff fields. If the field property/option is not visible, verify that you have completed Step 1

Step 3 - Configure "Other Signer Role(s)": Typically these roles are used for students, parents and guardians; but they can also be used for non-lookup staff roles for which there is not a staff profile field. If other signer role(s) are necessary for the document template, select "Configure DocuSign E-Signature" from the upper "More" dropdown menu (as per step 1) and observe that you can add other signer roles as shown in the screen shot below.

Configure DocuSign E-Signature: Individualized Education Program

(Boilerplate email subject/blurb to be received by signers.
Example: Please review and e-sign this Individualized Education Program document.)

DocuSign Email

Subject: Please review and e-sign this Individual Education Program docun
{FirstName} {LastName}

Blurb
(Optional)

Days Until Expiration: 20 (Optional number of days from sending to expiration if signatures not complete)

DocuSign Options Allow Submit Review Document

Staff Signer Fields CaseManager, Sign_BilingualSpecial

Other Signer Roles									
	Role ID	Role Caption	Source	Name Expression	Email Address Expression	Omit If Name Missing	Allow Prompting End-User for Missing Name/Email	Route with Staff	
		Parent1	Parent/Guardian	Profile	Parent1Name	Parent1EmailAddress	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

For each such signer role, it will be necessary to query either the profile or the document for the name and email address of the person who will be the signer. If you are expecting the queried name or email address to potentially have empty values, you can optionally set a checkbox option for that signer role to allow the system to prompt the end-user for the missing information at the time of submitting the document for e-signature. The following specific configuration elements are needed for each of the other signer roles:

- **Role ID:** This is a unique identifier that is used to associate e-signature tabs with the role.
- **Role Caption:** Identifies a label for the signer role, such as “Parent”. This caption is only displayed in the user interface for submitting documents for e-signature and monitoring their status. The role caption will never be shown within DocuSign to the actual signers of the document.
- **Source:** Identifies where the name and email address of the signer will be queried from. At this time, “profile” and “document” are the possible sources.
- **Name Expression:** A formula expression in the context of the source (profile or document) that produces the name of the signer in First Last format.
- **Email Address Expression:** A formula expression in the context of the source (profile or document) that produces the email address of the signer. This is typically just the name of a field containing email addresses.
- **Allow Prompt for Missing Name/Email:** If checked, and the name and/or email address values are blank in the context of submitting a particular document for e-signature, the end user is prompted and can supply those missing values at the time of submission. If this option is not

checked, the missing information will be called out, but the user will be prevented from submitting the document for e-signature.

- **Route with Staff:** If checked, then the corresponding signer role will always be routed at the same time as signers based on staff profile fields.

Note that is also possible to configure other signer roles in child templates to support repeating signer data. From the document template configuration, choose the child template from the fly out menu, and then select “Set Child Template DocuSign Properties”. Field mapping is not supported in child templates at this time.

Step 4 - Configure E-signature tab directives: Each staff field-based and other signer role should have one or more associated “signer tab” directives in the document template section designed for signature collection. The table below delineates all the variations in the syntax of signer tab directives. Note that all names used in the directive syntax are required to be alphanumeric (underscore allowed).

Directives	Explanation
{'StaffFieldName:SignHere'} {'RoleId:SignHere'}	Creates a “sign here” tab associated with the specified staff field or other signer role.
{'StaffFieldName:SignHere[instancename]'} {'RoleId:SignHere[instancename]'}	If there is a need to have the same person sign a document in more than one place, you can have more than one sign here directive for the same person, but each one must be identified with an instance name.
{'StaffFieldName:InitialHere'} {'RoleId: InitialHere'} {'StaffFieldName:InitialHereLO'} {'RoleId: InitialHere:O'}	Creates an “initial here” tab associated with the specified staff field or other signer role. An “O” modifier can be used here to make a particular “onital here” field optional for the signer.
{'StaffFieldName: InitialHere[instancename]'} {'RoleId: InitialHere[instancename]'}	If there is a need to have the same person initial a document in more than one place, you can have more than one initial here directive for the same person, but each one must be identified with an instance name.
{'StaffFieldName: DateSigned'} {'RoleId: DateSigned'}	Creates a “date signed” tab associated with the specified staff field or other signer role. Note that DocuSign automatically populates the tab with the date of signing.
{'StaffFieldName: DateSigned[=datefield]'} {'RoleId: DateSigned[=datefield]'}	Creates a “date signed” tab that is mapped to a date or date/time field in the document template. When the signers have all completed their signatures, the date or date/time will be written back to the document field.
{'StaffFieldName:CheckBox'} {'RoleId:Checkbox'}	Creates a “checkbox” tab associated with the specified staff field or other signer role. This syntax should only be used when there is only one checkbox for the signer. If there will be more than one checkbox for the signer, the “instance name” syntax in the next row should be used instead. In DocuSign, a signer is not required to set the value of the checkbox, and may leave it unchecked. To require a signer to make an explicit choice (i.e. give consent or deny consent), use “radio” tabs instead. See the notes below the table for more information on checkboxes.
{'StaffFieldName:CheckBox[instancename]'} {'RoleId:Checkbox[instancename]'}	If there will be more than one checkbox for a signer, a unique instance name must be assigned to each checkbox using this syntax. Note that the instance name only needs to be unique for checkboxes associated with the given signer. Two different signers may have a checkbox with the same instance name.
{'StaffFieldName:CheckBox[=logicalfield]'} {'RoleId:Checkbox[=logicalfield]'}	Creates a checkbox tab that is mapped to a logical field in the document template. When the signers have all completed their signatures, the value of the DocuSign checkbox will be written back to the document logical field.
{'StaffFieldName:Radio(groupname)[instancename]'} {'RoleId:Radio(groupname)[instancename]'}	Creates a “radio” tab associated with the specified staff field or other signer role. Radios are often preferable to checkboxes because the signer is required to make an explicit choice. The “group name” is used to group radios together into mutually exclusive groups. The instance name is required to uniquely identify each

	individual radio, and the instance name must be unique within the same group for the same signer. See the notes below this table for more information on radios.
{'StaffFieldName:Radio(groupname)[=logicalfield.true]}' {'StaffFieldName:Radio(groupname)[=logicalfield.false]}' {'RoleId:Radio(groupname)[=logicalfield.true]}' {'RoleId:Radio(groupname)[=logicalfield.false]}'	Creates a “radio” tab that is mapped to a document template logical field and a specific value or either true or false. When the signers have all completed their signatures, the specified logical value will be written back to the document logical field. For best results, the document logical field should allow empty values and default to empty so that it is unambiguous whether or not a value has been written back. See the notes below the table for more information on radios.
{'StaffFieldName:Radio(groupname)[=keywordfield.keyword]}' {'RoleId:Radio(groupname)[=keywordfield.keyword]}'	Creates a “radio” tab that is mapped to a document template keyword field and a specific keyword value. When the signers have all completed their signatures, the specified keyword value will be written back to the document keyword field.
{'StaffFieldName:Text:Wwidth}' {'RoleId:Text:Wwidth}'	Creates a “textbox” tab of the specified width, for example, {'Parent1:Text:W600}. By default, the textbox is optional for the signer, but one can add an additional R modifier to make the textbox required, as in {'Parent1:Text:W600R}. Multi-line word wrapping can be enabled on the textbox by specifying its height using an additional H modifier, as in {'Parent1:Text:W600H200}.
{'StaffFieldName:Text[instancename]:Wwidth}' {'RoleId:Text[instancename]:Wwidth}'	If there will be more than one textbox for a signer, a unique instance name must be assigned to each checkbox using this syntax. Note that the instance name only needs to be unique for textboxes associated with the given signer. Two different signers may have a textbox with the same instance name.
{'StaffFieldName:Text[=characterfield]:Wwidth}' {'RoleId:Text[=characterfield]:Wwidth}'	Creates a “textbox” tab that is mapped to a document template character fieldvalue. When the signers have all completed their signatures, the specified character value will be written back to the document keyword field.

Radios and Checkboxes: A “radio” or “checkbox” directive will both render a checkbox whether or not DocuSign is active. When a document is sent to DocuSign for signing, DocuSign will actually render its own checkboxes and it is helpful to leave plenty of space for DocuSign to do that. In DocuSign, a signer is not required to set the value of a checkbox, and may leave it unchecked. Radios are often preferable to checkboxes because the signer is required to make an explicit choice. The example formatting below implements a requirement for the signer to explicitly choose giving consent or not giving consent.

```
<div>Do you consent to the plan outlined in this document? &nbsp; <label>{'ParentSigner:Radio(Consent)[=DocuSignConsent.true]}' I give consent</label> &nbsp; <label>{'ParentSigner:Radio(Consent)[=DocuSignConsent.false]}' I do not give consent</div>
```

Note that if the fields being written to (i.e. DocuSignConsent in the example above) are not explicitly displayed in a document section, it is a good idea to mark them in the data dictionary as “Internal Values” so that they do not mistakenly get deleted.

In some cases, you will want to have a standard checkbox data field that effectively gets swapped with a DocuSign checkbox tab when the PDF is generated for DocuSign. In this case, have the logical field directive right next to the tab directive and apply the built-in “HIDEIFDOCUSIGN” CSS class to the logical field directive. Additionally, you will want to modify the tab directive so that it does not automatically render a checkbox outside of Docusign. To do that, modify the tab directive to add an “I” (for invisible) modifier such as at the end of the following syntax:

```
{'StaffFieldName:Radio(groupname)[?keywordfield.keyword]:I}'
```

FYI – Important: A document will only be sent to a particular signing role if one or more signing tabs associated with that role are found in sections included in the finalized document. This enables signing roles to be conditionally included or excluded using #IF logic, or conditional inclusion of sections.

To trouble shoot issues, it is helpful to understand how the signing tabs work “under the hood”. A directive like {CaseManager:SignHere} is rendered as the HTML shown below. The “ESIGNTAB” CSS style is designed to keep the “[CaseManager:SignHere]” text invisible in the document.

```
<span class="ESIGNTAB">[CaseManager:SignHere]</span>
```

The HTML coding above is embedded in the PDF document to DocuSign. DocuSign recognizes the coding and positions the tabs accordingly. PowerSchool Special Programs injects the appropriate coding into final documents whether or not a DocuSign account is in place, which enables past documents to be potentially signed once an account is in place. However, documents finalized before the signer tab directives were configured in the section HTML format will not be sent to DocuSign successfully because they will not have the HTML coding.

Step 5 (define reporting and workflow): There are a number of functions documented in the “Formula Reference” document that are specifically related to DocuSign E-signature. These functions can be used in the context of a report definition to pull out documents that have a particular state in the signing process.

With regard to workflow, it is important to be aware of the distinction between users authorized to “submit” documents for E-Signature versus users authorized to “send” documents to DocuSign. Most pricing plans for DocuSign limit the number of users who can send documents via DocuSign. To overcome this limitation, users who are not provisioned as DocuSign users can nonetheless be authorized with a “Submit to DocuSign” document template security right, which authorizes them to submit the document to another user who has been provisioned as a DocuSign user and can therefore send documents via DocuSign. DocuSign users are provisioned within the DocuSign administrative dashboard. The corresponding users in PowerSchool Special Programs must be granted the “Send Documents via DocuSign” student privilege. The “Send Documents via DocuSign” privilege can be granted at different organizationa levels to support, for example, having one sender per school.

It is helpful here to have an understanding of the distinct states that a document goes through in the DocuSign process:

- **Submitted:** The document has been submitted to an authorized sender (user provisioned within DocuSign) but has not actually been sent yet via DocuSign.
- **Sent and Pending:** This status means that an authorized user, provisioned as a DocuSign user and with the “Send Documents via DocuSign” privilege, has sent the document to DocuSign for signing. DocuSign manages the signing process forward. Pending refers to the fact the document has not reached one of the three end states yet.
- **Voided:** This is one of three possible end states. This means that the sender has voided/canceled the signing of a document. Signers are notified and processing stops. A document is also voided if it times out by exceeding the allowed amount of time for the sending process. Note that a reason for voiding is captured along with this state.

- **Declined:** This is one of three possible end states. If any signer declines the document, the document as a whole is now declined and processing stops. Note that a reason for declining is captured from that signer.
- **Completed:** If all signers sign the document, the process is complete and the document is considered signed.

When processing of a document is complete and the state changes to either voided, declined or completed, then the submitter and/or sender are automatically notified with a message that contains a link back to the document. If additional workflow is needed, a document action can be created with a “Modify Data When Electronic Signature Status Changes” trigger, which is further qualified to be triggered when the status changes to one or more specific values (see below). It can be assumed that any document field values mapped to signing tabs will be written back before any document action triggers are evaluated, which allows the workflow logic to be based on such mapped field values.

Omitting certain document sections from DocuSign [New in 20.11]: There is a document template section option named “Omit from Electronic Signature”, which when enabled, will result in the section being omitted from the document sent to DocuSign for signing. This option is overrideable by the system administrator.

DocuSign Conditional Logic Directives [New In 20.11.1.0]: The following directives are helpful in controlling certain section content designed for DocuSign:

- `{#IF_DOCUSIGN_ACCOUNT_EXISTS}content{#ENDIF}` includes the content only if a DocuSign account has been configured by the customer.
- `{#IF_NOT_DOCUSIGN_ACCOUNT_EXISTS}content{#ENDIF}` includes the content only if the customer has not configured a DocuSign account.
- `{#IF_DOCUSIGN_STAFF_SIGNING_ENABLED}content{#ENDIF}` includes the content only if the customer has not disabled staff signing for the document template.

- {#IF_DOCUSIGN_STAFF_SIGNING_DISABLED}content{#ENDIF} includes the content only if the customer has disabled staff signing for the document template.

Configuring Digital Signature

PowerSchool Special Programs allows the configuration of Digital Signature for specific document templates that require signatures, such as parental consent documents. Digital Signature configuration for document templates can lie dormant in a customer's database until a Digital Signature license is configured by the customer. This allows a state/province model database to be delivered fully ready for Digital Signature, and then activating the functionality only requires licensing and enabling Digital Signature.

The Digital Signature configuration process was built on three principles: 1) configurators should not need to completely refactor a document to add Digital Signature to it, 2) customers using the model but not using Digital Signature should not see any change to how their document functions, and 3) the configuration process should align with existing signature functionality as much as possible.

As a result, configuring Staff and Other Signers is done through the same configuration that is needed for DocuSign. In addition, reporting on Digital Signature is done using the same methods as DocuSign. Configuring which fields need to be filled out by which signers follows a different configuration method.

Activating Digital Signature: Digital Signature is active when it is both licensed and enabled. Templates can be configured for Digital Signature even when Digital Signature is not active, however the Digital Signature configuration cannot be tested without activating Digital Signature.

Licensing Digital Signature: Digital Signature can be licensed using the Manage Client Licenses tool. Once the license “e Signature” has been activated, admin users and staff users who can view configuration will gain access to the Digital Signature Settings tab.

Enabling Digital Signature: Digital Signature can be enabled from the Digital Signature Settings page, located at Administration -> Configuration -> Settings -> Digital Signature Settings. Admins and Consultants can enable Digital Signature. Staff users who can view configuration can view the Digital Signature settings but cannot modify the Digital Signature settings. Once enabled, users are able to submit documents for signing and sign documents.

When Digital Signature is active (i.e. licensed and enabled), templates with Digital Signature configured will behave differently. Staff users filling out a document will still be able to enter data into signature areas, however any required fields in signature areas will not be required. Once a signature request is submitted, signature areas will only be editable when accessed through the Signature Wizard by the right signer. When signing, required fields will be required again. The Signature Wizard is used to guide users through the Signing process and collect the signature data. When all signers have completed signing, the collected signature data is saved to the document.

When Digital Signature is not active (i.e. unlicensed or disabled), the document will continue to function like it did prior to the introduction of Digital Signature.

Configuration Step 1: A Digital Signature configuration must be created for templates that are being configured for Digital Signature. This works the same as DocuSign configuration, **Step 1**.

Step 2 - Configure staff signer role(s): Staff Signer roles must be created for templates that have staff signers. This works the same as DocuSign configuration, **Step 2**, except that a Staff Profile Reference field's Description value is used as the Role Caption if it exists. Signers added through this method can sign through the application, as well as through the signature request email.

Step 3 - Configure “Other Signer Role(s)”: Other Signer roles must be created for templates that have non-staff signers. This works the same as DocuSign configuration, **Step 3**.

Step 4 - Configure Digital Signature areas: Digital Signature is configured using the #IFSIGN directive and the ‘G’, ‘S’, ‘I’, and ‘L’ directive modifiers. These tools can be used to define certain areas of the document as signature areas associated with a signer role. When Digital Signature is enabled, fields in these areas can only be modified by the signer with the specified signer role. These areas cannot be edited by the staff member filling out the rest of the document or other signers. When Digital Signature is disabled, the fields in these areas act normally.

Directive	Explanation
{#IFSIGN RoleID} ... {#ENDIF}	The RoleID must match the role ID of one of the staff signers associated with the document, or one of the other signers associated with the document.
{#IFSIGN RoleID1, RoleID2, RoleID3, ...} ... {#ENDIF}	When signers are signing the document, the content within the #IFSIGN directive is only editable for the signer with a matching Role ID. When staff are editing (not signing) the document <u>before a signature request is submitted</u> , fields that appear within the #IFSIGN directive will be editable to allow staff to pre-fill certain values that the signers will be able to edit while signing.
	Multiple RoleIDs can be associated with a single #IFSIGN directive. When multiple roles are listed, each matching role can sign the area. If multiple signers sign the same area, all responses are displayed on the document and signed PDF, but only the first response will be saved to the document's data. Warning: RoleID is case sensitive and must exactly match the configured RoleID.
{#IF_ESIGNATURE_ENABLED} ... {#ENDIF}	Includes the content if Digital Signature is enabled. Alternatively, formula field, ESignEnabled, can be used. Eg. {#IF . ESignEnabled} {#ENDIF}
{#IF_NOT_ESIGNATURE_ENABLED} ... {#ENDIF}	Includes the content if Digital Signature is not enabled.
{#IF_STAFF_SIGNING_ENABLED} ... {#ENDIF}	Includes the content if staff signing is currently enabled for this document template.
{#IF_STAFF_SIGNING_DISABLED} ... {#ENDIF}	Includes the content if staff signing is currently disabled for this document template.

{#IFVIEW} ... {#ENDIF}	The behavior of #IFVIEW directives (including #IFVIEWAND, #IFVIEWOR, etc.) changes when the document is accessed through the signature wizard. Includes the content if it is not in a signature area with a RoleID matching that of the current signer.
{#IFEDIT} ... {#ENDIF}	The behavior of #IFEDIT directives (including #IFEDITAND, #IFEDITOR, etc.) changes when the document is accessed through the signature wizard. Includes the content if it is a signature area with a RoleID matching that of the current signer.

The G field directive modifier can be used to control the behavior of a field in an #IFSIGN directive:

Directive Modifier	Explanation
{LogicalValue:S} {LogicalValue:SL} {LogicalValue:SL"Label"}	When rendering the document in the signature wizard, the logical value field will be rendered as a Sign button. When clicked, the Sign button will be replaced by the signer's signature. When rendering the signed PDF, a value of True will display the signer's signature. A value of False will not display anything. When rendering for anything else, nothing will be displayed. When rendering with an 'L' directive modifier, a line will be placed underneath the sign button, signer's signature, or empty area. When rendering with an 'L:"text here"' directive modifier, a line and label, with the wording "text here", will be placed underneath the sign button, signer's signature, or empty area.
{LogicalValue:I} {LogicalValue:IL} {LogicalValue:SI"Label"}	When rendering the document in the signature wizard, the logical value field will be rendered as a Initial button. When clicked, the Initial button will be replaced by the signer's initials. When rendering the signed PDF, a value of True will display the signer's initials. A value of False will not display anything. When rendering for anything else, nothing will be displayed. When rendering with an 'L' directive modifier, a line will be placed underneath the initials button, signer's initials, or empty area. When rendering with an 'L:"text here"' directive modifier, a line and label, with the wording "text here", will be placed underneath the initials button, signer's initials, or empty area.
{FieldName:G}	Removes the field from the signature area. In other words, this field will be editable by the staff user filling out the document, and will not be editable by the signer whose RoleID matches that of the #IFSIGN directive.
{TextFieldName:G"system:FullName"} {TextFieldName:G"system:Initials"} {DateTimeFieldName:G"system:SignatureDate"}	Associates the field with one of the system's signature fields. When the system field is updated by the Signature Wizard, this field's value will also be updated. This field will otherwise be read-only in all circumstances.
{FieldName:Modifiers:G"OtherFieldName:Modifiers"}	When Digital Signature is active, OtherFieldName will be rendered with the specified modifiers.

	<p>When Digital Signature is not active, FieldName will be rendered with any other specified modifiers.</p> <p>This directive modifier is typically used to provide an alternate Logical Value field for a signature when the form already contains something like a text field for the signature.</p> <p>This directive modifier is equivalent to</p> <pre>{#IF_NOT_ESIGNATURE_ENABLED} {FieldName:Modifiers} {#ELSE} {OtherFieldName:Modifiers} {#ENDIF}</pre>
--	--

Step 5 (define reporting and workflow): You can report on Digital Signature data using the same methods outlined in DocuSign configuration's **Step 5**.

Other Digital Signature Settings: The Digital Signature Settings page, accessed through Administration -> Configuration -> Settings -> Digital Signature Settings, contains a couple of other settings that will control Digital Signature behavior.

The **Document Delivery Options** can be used to determine what is sent to signers after the signing process has been completed. If no options are checked, signers are not notified when the signing process has been completed.

The **Default Routing Order** can be used to determine the standard routing order preferred by the district.

The **PDF Creation Options** can be used to tell the system how to handle #IFSIGN areas with data from multiple signers. “One PDF with all signer’ responses” will always generate one PDF, with any areas with data from multiple signers duplicated for each signer. “Separate PDFs for each non-staff signer when responses differ” will generate separate PDFs when there are any areas with data from multiple signers, otherwise it will generate one PDF.

The **Notify staff when a signed document is modified** setting can be used to control the info message to staff signers when they access a document that has been modified after it was signed. When checked, the info message is displayed. When unchecked, the info message is not displayed.

The **Staff Email Address Field** can be used to determine what Staff profile field contains the email address of staff members.

The **Digital Signature-Ready Document Templates** table can be used to disable Digital Signature for staff users on specific document templates.

In addition, specific sections of document templates can be **Omitted from Digital Signature** by selecting the “Omit from Electronic Signature” property for that section.

Tips and Tricks

When using the #IFSIGN directive with multiple RoleIDs, make sure that the HTML content of the directive includes the entire element, not just the opening or closing tag since the area may be repeated.

For example, this will have unexpected results:

```
{#IFSIGN RoleID1, RoleID2}
<table><tr><td>Label</td><td>{SignatureField}</td></tr>
{#ELSE}
<table><tr><td>Other non-signature text</td></tr>
{#ENDIF}
</table>
```

Instead, the closing tag should be included in the #IFSIGN area.

```
{#IFSIGN RoleID1, RoleID2}
<table><tr><td>Label</td><td>{SignatureField}</td></tr></table>
{#ELSE}
<table><tr><td>Other non-signature text</td></tr></table>
{#ENDIF}
```

#IFSIGN areas should be editable while in draft or review so that the data can be ‘pre-filled’ before signers sign.

The response priority for signers who have already signed an active signature request will not be modified when this expression is modified. If there are multiple possible priorities, be sure to leave gaps between them so you can easily add priorities between existing priorities later (i.e. use 0, 100, 200, 300 instead of 0, 1, 2, 3).

Other Configuration for Digital Signature

ESignature Configuration

For the most part, a document’s ESignature Configuration acts the same whether it’s used with Digital Signature or DocuSign. However, there are a couple of settings that will behave differently.

The **eSignature Email Subject** and **Blurb** are both optional. If either are not specified, the system will provide a default subject/blurb for Digital Signature emails.

The **Allow Submit Review Document** option allows Digital Signature submission while the document is in Review. When used with Digital Signature, this setting also allows Digital Signature submission while the document is in Draft, although the document will need to be promoted to review before the submission process can be completed.

Other Signer roles can have a **Response Priority Expression** specified. This expression is only used with the #IFSIGN digital signature configuration method. When multiple signers have entered data for the same fields, the first signer to submit their response is the one whose data will be used on the final document. This expression can be used to override that default behavior. The data from the signer with the highest response priority will be used on the final document. If multiple signers have the same response priority then the data from the first signer among that group will be used on the final document. This expression is run when the signer's data changes, and the result is saved to the signer's ResponsePriority field.

Section Actions

The **Prevent Section Completion** section action has some options that relate to Digital Signature:

- “Only when editing document” means that the section action will not prevent section completion while signing the document and will only prevent section completion while editing the document.
- “Only when signing document” means that the section action will not prevent section completion while editing the document and will only prevent section completion while signing the document.
- “When editing and signing document” means that the section action will always prevent section completion - both while editing the document and while signing the document.
- “Only when signing if Digital Signature is enabled; otherwise, when editing” means the section action will behave differently depending on whether Digital Signature is enabled. If Digital Signature is enabled, then the section action will function like “Only when signing document”, and if Digital Signature is disabled, then the section action will function like “Only when editing document”. This option can be used to create a section action that will be useful both for customers using Digital Signature, and those opting not to use it at all.

In addition, the system will only run section actions where one or more fields referenced by the trigger criteria are editable by the signer. If none of the fields referenced by the trigger criteria are editable by the signer, then the section action is not checked for that signer. This prevents actions meant for one signer role from preventing the completion of the signing process by another signer role.

Document Actions

The **Modify Data when Electronic Signature Status Changes** document action applies to both DocuSign and Digital Signature. When the document's signature request is updated to one of the specified statuses, the rest of the document action will be run.

Section Properties

The **Omit from Electronic Signature** section property applies to both DocuSign and Digital Signature. When selected, the section will be excluded from the Signature Wizard and signed PDF.

Using DocuSign Configuration with Digital Signature

Models configured for DocuSign are also compatible with Digital Signature. No configuration changes should be needed for DocuSign documents, although there may be edge cases where the two systems are incompatible – see the **Tips and Tricks** section below.

Submitting a document configured for DocuSign to Digital Signature follows the same process as submitting a DocuSign document to DocuSign. Once submitted, the document switches to the Digital Signature signing process.

Transitioning to Digital Signature from DocuSign: If a DocuSign account exists and Digital Signature is unlicensed or disabled, the system will submit new signature requests to DocuSign. Once Digital Signature has been licensed and enabled, the system will submit new signature requests to Digital Signature, even if the DocuSign account still exists. Existing signature requests that were made to DocuSign and have not been completed will continue to communicate with DocuSign.

Signed Documents: As each signer signs the document, the document will be updated to include that signer's responses. These responses will be seen in view, edit, and print mode. Signatures will only be included on the final signed PDF.

Tips and Tricks

- Repeating rows for child templates that contain DocuSign fields must not be set to View Only. The child template must at least be updatable so that the DocuSign fields can be updated from the Signature Wizard.
- Since DocuSign documents signed with Digital Signature will display all tab responses on the signed document, some bindings that are mapped to document fields may display duplicate values. For example:

```
{PGExcusalParent1Date}{ 'ParentSign:DateSigned[=PGExcusalParent1Date]}
```

Will display the date twice on a signed document – once from the standard binding, and once from the DocuSign binding. If the binding is changed to something like:

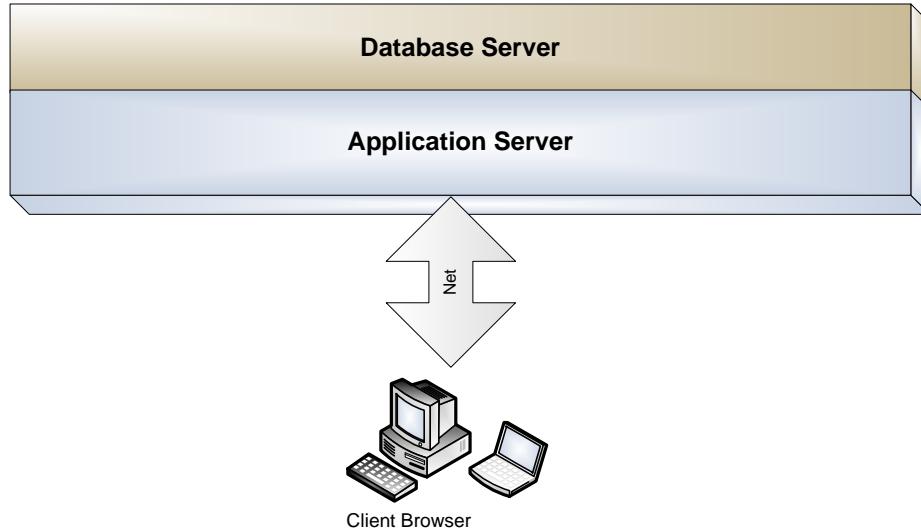
```
{#IF_NOT_ESIGNATURE_ENABLED}{PGExcusalParent1Date}{#ENDIF}
{'ParentSign:DateSigned[=PGExcusalParent1Date]}
```

Then the value will display once.

- Fields directives that directly reference the profile (such as `Profile.Parent1GetLetters`) will use the latest data from the profile when rendering a document for signing. If the profile is changed after a signing request is initiated, then a document that is being signed will also change to match the profile. If this behavior is not desirable, a document field that has a data flow from the original profile field should be used in place of the direct profile field reference.

Optimizing Document Speed and User Responsiveness

To optimize the speed and responsiveness of documents, it is helpful to be aware of the level of the system architecture in which various configuration features operate. Below is a simplified system architecture diagram that is helpful for this purpose. Configuration features that interact with the database server are the slowest. Configuration features that execute in the client browser will be very fast. Therefore, whenever it is possible to switch from features that interact with the database with those that execute in the client browser (or the application server), the user experience will be faster.



The following approaches can be used to replace configuration features that interact with the database server with those that execute in the client browser or on the application server:

- Wherever possible, replace “auto-postback”, which interacts with the database server and is noticeably slow, with JSIF which executes immediately in the client browser or on the application server.
- Where possible, replace #IF statements with JSIF plus “simulate-if” keyword. A section that has many #IF statements, especially in a repeating row layout, can slow down the user experience. But JSIF plus the “simulate-if” keyword exactly simulates the behavior of an “{#IF formula;” directive without any extra interaction with the database server. With the “simulate-if” keyword, JSIF does not render any enclosing tags nor any JavaScript triggering behavior, and therefore it behaves identically to the #IF directive. Simple instances of {#IF formula} can often be converted directly to {#JSIF expression, simulate-if} with no other changes required. A section that has many #IF statements, especially in repeating rows, can run noticeably faster if they are converted to JSIF with simulate-if. Note that “simulate-ifviewor” and “simulate-ifeditor” are also supported and simulate #IFVIEWOR and #IFEDITOR respectively.

- Where possible, switch to the “fast add” feature for repeating row layouts that allow addition of rows. This will speed up the user experience greatly.

SQL Timeout Errors: The practices described below can correct certain situations where there are SQL timeout errors, deadlock errors, or sluggish performance in documents.

- **Minimize use of “dot” operator:** A lot of references to Profile.field in document/section actions and document forms can lead to poor performance (including SQL timeout errors and deadlock errors). The best practice is to move the values of such profile fields into the document using one-way dataflow (with automatic updating). Then these values can be referenced directly in the document. This approach has a secondary historical benefit in that it captures the original profile value fields at the time the document was finalized.
 - **Optimize use of aggregate functions:** Child template calculated fields that use aggregate functions (EXISTS, COUNT, etc) can be a source of performance degradation. The system will load the value of every data and calculated field in the child template prior to rendering the repeating rows/sections. So if such calculated fields are not displayed in every single repeating row or section, they are being loaded whether or not particular values are displayed. To optimize this, such calculated fields can be marked as “Internal Value Only” which prevents the preloading of their values. Then in the HTML section, reference the calculated field using a formula directive `{=CalculatedField}` instead of a field directive `{CalculatedField}`. Note that this optimization will not yield any benefit if the calculated field is a simple calculation (no aggregates) or if it is displayed on every repeating row or section.
 - **Optimize Indexes:** ClickConfiguration > Profile Types > More... > View Index Suggestions to access a screen that suggests additional quick search indexes that would make a difference based on the recent actual user workload in the database. Specifically the screen suggests specific fields where performance could be improved by assigning the quick search index property and/or the “Include with Quick Search Indexes” property. Thw “Include with Quick Search Indexes” property enables the field’s value to be stored in quick search indexes without acting as the main index key.

Printing Special Document Content for Student/Parent/Guardian

A document template can be configured to allow a user to print a version for the document for a student, parent or guardian with special content added or removed. To do this, in the HTML format for any section, use the following CSS classes to mark content to be visible to staff only, students only, or parents/guardians only. Note that “staff only” is the default mode of printing and viewing for all users who are not students, parents or guardians.

- STUDENTONLY for example, <div class="STUDENTONLY">content</div>
 - STAFFONLY for example, <div class="STAFFONLY">content</div>
 - PARENTGUARDIANONLY for example, <div class="PARENTGUARDIANONLY">content</div>

Note that applying the above classes is enough to have the document appear as expected if student, parent, guardians are able to log into their respective portal and view the document. But one additional step is needed to allow staff to print the document for a student, parent or guardian (since when staff print documents, by default they will print the staff version of any document). In document template properties for the document template, you will need to check “Enable Print for Parents/Guardians” and/or “Enable Print for Students” under print options. Once enabled, when staff use “Print Selected Sections”, the popup will have a “Print For?” option that defaults to staff.

Configuring Forms

Chapter

8

HTML

The following requirements must be met to be XHTML compliant:

- All tag and attribute names must be in lower case.
- All open tags must have a corresponding close tag, or if the tag is self-contained such as
, the self-contained tag should end in />.
- Attribute values should be enclosed in quotes.
- Tags should be well formed, i.e. <label>Name</label> is not well-formed but <label>Name</label> is.

One of the most important best practices of modern web design is separation of semantic content (HTML) from presentation (CSS). Semantic content refers to the information and its meaning as stored in a particular document, and presentation refers to how that information is rendered and presented in the browser. The best practice is to remove as much presentation information as possible from the HTML document so that it focuses purely on content and meaning, and use separate CSS styles to instruct the browser how to present that information in a particular context. Following this best practice yields the following benefits:

- The volume and complexity of HTML can be significantly reduced, and so the maintenance effort is reduced as well.
- It becomes much easier to apply a consistent look and feel across documents and sections of documents.
- There will be much fewer problems resulting from multiple authors with different markup styles if the best practices are followed, and the resulting HTML markup is much leaner.

CSS

All document template sections and other configurable HTML formats occur in the larger context of a PowerSchool Special Programs page. As a result, they automatically benefit from PowerSchool Special Programs CSS defaults that are established in its master style sheet to help ensure consistency across browsers. For example, all tags are set to have zero margins with a font of “Arial, Helvetica, sans-serif”. Many other commonly used tags have a padding of zero.

```
* {margin: 0; font-family: Arial,Helvetica,sans-serif; }
p, ul, li, body, form {padding: 0; }
hr {color:#999999; }
```

```
img {border:0;}
em {font-style:italic; font-weight:normal; }
strong {font-style: normal; font-weight: bold; }
table {font-weight:normal}
```

It is possible to override the above defaults or to establish entirely new tag defaults for just the configurable portion of PowerSchool Special Programs pages that is rendered from your HTML content. This is possible because PowerSchool Special Programs always renders profile form section content within a <div id="PROFILEFORMCONTENT">....</div> tag pair and similarly always renders all document template content inside a <div id="DOCCONTENT">....</div> tag pair. With this knowledge, you can set and/or override default styles in several ways.

If you want to apply styling to ALL tags inside of the DOCCONTENT division, you can create the following style.

```
#DOCCONTENT * {font-size: 12pt;}
```

The asterisk ensures that the style is applied to all tags within content rendered from HTML formats. Note that if you omitted the asterisk, the styling would be applied to the division tag and inherited by most tags inside it, but with the notable exception of table tags; hence the asterisk is critical to ensuring that everything is affected.

You can also apply the style to certain tags anywhere they appear inside the DOCCONTENT division, as follows below.

```
#DOCCONTENT p, ul, li {font-size: 12pt;}
```

If you wish to make sure content is not included in printed output, you can use the “DISPLAYONLY” class built into PowerSchool Special Programs.

It is important to note that when a document is finalized, unlike the HTML format of the sections, the style sheet at the time of finalization is not preserved. Therefore, when changing the style sheet, changes must be made in a way that ensures backward compatibility. The style sheet is not preserved because, if every final document effectively had its own style sheet, certain functionality like bulk printing would not work correctly.

HTML Best Practices

These practices focus on separation of content from presentation.

Tags like , <i> and <u> have no semantic meaning and are only used to determine appearance. Consider using in place of , and in place of italics. Actually, the semantic meaning of is to “emphasize text”, and that of is to “strongly emphasize text”. However, PowerSchool Special Programs uses CSS defaults to make display in bold-face, and in italics.

Use `` and `` to create lists of information in a semantic way that assists screen readers and is better for the visually impaired. If you don't like the bullets, you can control that with CSS (`list-style-type:none`).

Avoid using tag attributes intended to control appearance, and avoid placing styling information directly in tags except for one-off exceptions. Even class attributes can be avoided to an extent by using techniques described earlier based on id. The more presentation information that stays in the style sheet, the leaner your HTML formats will be.

In many cases, it is possible to avoid using tables to control layout by instead using something like the following:

```
<div style="float:left">left-column content</div>
<div style="float:right">right-column content</div>
<div style="float:clear">Continued content occupying both columns</div>
```

Accessibility

Be mindful of best practices to enhance accessibility of forms, and at the same time, this will ensure that the forms are compliant with governmental legislation like Section 508 in the USA. One important best practice is to use semantic HTML for content and CSS for presentation as described in the previous section. When markup is laden with tags, attributes and hacks designed to have it look a particular way for current browsers, it will be much less intelligible to screen readers and other aids for people with visual challenges.

Another important practice is to test that all fields have associated labels. A simple way to test that an existing form meets this best practice is to click on all the field labels in edit mode. When you click a field label, the field should receive the input focus. If the field does not receive the input focus, the field has no associated label (or possibly you clicked the wrong text and some other text is actually the label).

In some cases, PowerSchool Special Programs automatically renders labels according to best practice. This is the case with the `{field:T"label"}` or `{field:L"directives"}` directives and when dual Yes/No checkboxes are rendered for a logical field that allows empty values (checkboxes are labeled Yes and No respectively). In other cases, you may need to supply your own label, and in order to meet accessibility standards, you must make sure your label is explicitly associated with the field. There are two ways to do this:

1. Simply wrap a label around the field, as follows:
`<label>Field Label: {fieldname}</label>`
2. Make an explicit association as follows:
`<label for="$fieldname">Field Label:</label> {fieldname}`

In the second approach above, note the '\$' in front of the field name in the label tag. The '\$' instructs the system to make any necessary adjustments to the field name that are needed to complete the association. For example, if the field is in repeating rows, the \$ will instruct the system to adjust the reference to the field as needed for each repeating row. The '\$' also validates the field name, such that if it not valid, it will show as a misconfiguration. Finally, it future proofs the implementation so that it will work as expected in

future platform versions. Note also that a few special cases are also supported. To label the No checkbox of a logical field that allows empty values, use the `<label for="$LogicalField!">` syntax (note the exclamation point). To label the time field of a date/time field, use the `<label for="$DatetimeField#TIME">` syntax.

In a profile form section, if you want to reference a particular value of a multiple value field, then use markup like that below. If you omit the \$, the association will not be made.

```
<label for="$fieldname[value]">Field Label:</label> {fieldname[value]}
```

Best Practices to Support Translations

When configuring forms, one must be mindful to avoid certain practices that prevent the translation engine from working the way that it should. Typically, the patterns to avoid involve forcing the system to use untranslated values in translated forms. Here are things to avoid:

1. `{=KeywordField.Description}` forces the system to always display the untranslated value. But administrators will add additional columns like `Description_es` to hold the translated descriptions that should be displayed instead in translation mode.
`{=KeywordField}` allows the system to choose the best column based on the current language. However `{=KeywordField}` should be replaced with `{KeywordField}` or `{KeywordField:R}` (to force read only) for efficiency reasons.
2. `{=KeywordField.OtherKeywordField.Description}` should be replaced with `{=KeywordField.OtherKeywordField}` to allow the system to display the appropriate description column based on the language.
3. `{=SELECTBYVALUE(SortOrder, 1:"Special Education Services", 2: "Career and Technology Education Services", 3: "Related Services")}` prevents translation because it hard codes English within a formula. This should be replaced with something that allows the phrases to be extracted for translation into multiple languages, such as:
`{#IF SortOrder =1 }Special Education Services{#ELSEIF SortOrder =2 }Career and Technology Education Services{#ELSEIF SortOrder =3 }Related Services{#ENDIF}`
4. `{=IFELSE(UseFieldA=true,FieldA,FieldB}` (where `FieldA` and `FieldB` are translatable long text fields) prevents translation by hardwiring the English versions of the field values into the directive. The formula language does not reference translated values of fields. Therefore this directive should be written as `{#IF UseFieldA }{FieldA }{#ELSE }{FieldB }{#ENDIF}`.

Field Directives & Validation Capabilities by Data Type

From a design standpoint, it is best to choose the data type for each field (e.g. character, numeric, logical, etc) based strictly on the nature of the data rather than how it will be presented to the user on a form (e.g. checkbox, textbox, dropdown, etc). However, when designing a form, control over the presentation of each field is desirable. PowerSchool Special Programs typically supports several alternative presentations for each data type thereby allowing the ideal data type to be chosen without being overly concerned with presentation limitations. Below are details on the presentation and validation capabilities for each data type. All editable fields and data types support the basic field labeling options, but the information below focuses on what is different about each data type.

- **Character:** Character fields are always presented as a single-line text box in edit mode. The width of the text box can be specified with the “W” modifier. You can specify a width in characters, for example `{fieldname:W20}`, or as a percentage of the available width; for example `{fieldname:W99%}`. In a document template, the “S” modifier specifies that the character field be spell-checked, whereas by default it is not spell checked. When not in edit mode, the character field is generally displayed as text with a few exceptions. If the character field has either the “URL” or “Email Address” field property option, it will be displayed as web or email hyperlink respectively.

From a data validation standpoint, the following field properties will automatically enable validation of the field and rejection of invalid values: “URL”, “Email Address”, “Require Numeric”, and “Require Alphanumeric”. Other types of validation checks can be implemented by creating a section action with a triggering event of “Rollback Saving Section”, and which have the “Anchor user message to first referenced field” option checked.

- **Logical:** Logical fields that do not allow empty values are always presented as checkboxes. Logical fields that allow empty values are presented, by default, as two checkboxes labeled “Yes” and “No”, but can alternatively be presented as a dropdown menu by introducing the “D” field modifier. Either way, the labels associated with true and false values can optionally be configured as follows: `{logicalfield:+"labelforyes"- "labelforno"}`. The “Yes” value is normally displayed before the “No” value, but that order can be reversed by including the “N” modifier.

When the logical field allows empty values, the two checkboxes can be positioned in two separate places on the form by using two separate directives. Use `{logicalfield:+"labelforyes"}` to position the checkbox for the true value, and `{logicalfield:-"labelforno"}` to position the checkbox for the false value.

A group of related logical fields can be laid out in a way that enforces a requirement that the user must check at least one checkbox with the overall behavior consistent with them being a single required field. The red/yellow colors are applied and change as expected. This is configured as follows:

1. Arrange the logical fields on the form inside of a common parent element such as `<div>`, `<p>` or `<fieldset>`. Note that this common element will change color when the first checkbox is checked.

2. In the directive for each of the logical fields, include an O modifier to specify a unique group name. For example, {fieldname:L"label"O"groupname"}. The group name is simply an identifier that is used to group the checkboxes together.
- **Date and Date/Time:** A date field is presented as a textbox with an associated calendar popup when in edit mode. The “W” width modifier is supported as with the character field. The “O” modifier can be used to specify a precise output format to be used when not in edit mode. An example of an output format is “ddd, dd/mm/yyyy” which would output something like “Mon, 02/28/2012”. The following character placeholders can be used in the overall format:
 - d (day, 1-31), dd (day, 01-31)
 - ddd (day of week, Mon-Fri), dddd (day of week, Monday-Friday)
 - h (hour, 1-12), hh (hour, 01-12), H (hour, 0-23), HH (hour, 00-23)
 - m (minutes, 0-59), mm (00-59)
 - M (month, 1-12), MM (month, 01-12)
 - MMM (Jan-Dec), MMMM (January-December)
 - s (seconds, 0-59), ss (seconds, 00-59)
 - tt (AM/PM)
 - yy (year with 2 digits), yyyy (year with 4 digits)
 - Non-letter characters are copied to the output as is, except that \ is an escape character. For example, {MyDateTime:O"HH\h\mm"} would output a time that looks like 23h12.

A date/time field will include an additional textbox for time. The time portion can optionally be positioned separately from the date portion using two separate directives. The {*datetimefield:D*} directive (upper-case “D” modifier) will position the date portion and {*datetimefield:d*} directive (lower-case “d” modifier) will position the time portion. Note that you cannot have an editable time component without an editable date component, although you can make the date component invisible using the “@” modifier as long as date component will always be filled with a default value.

From a data validation standpoint, the following field properties will automatically enable validation of the field and rejection of invalid values: “No Past Dates in Draft”, “No Future Dates”, and “Time Required” (date/time fields only). Due to the way the future moves into the past over time, the “No Past Dates in Draft” is only enforced when the document is in draft mode (as the label suggests). The “Time Required” option is only for date/time fields. If this option is not enabled, the user is allowed to omit the time portion. Other types of validation checks can be implemented by creating a section action

with a triggering event of “Rollback Saving Section”, and which have the “Anchor user message to first referenced field” option checked.

- **Numeric and Integer:** Numeric and integer fields are presented, by default, as a textbox in edit mode, and support the width modifier as with the character field. The “O” modifier can be used to specify a precise output format to be used when not in edit mode. Specific examples are given below.

▪ O"c"	14.1 => \$14.10, £14.10, ... (currency, server culture)
▪ O"X2"	15 => 0F (hexadecimal from integer only)
▪ O"x4"	254 => 00ffe (hexadecimal from integer only)
▪ O"n0"	integer 9999001 => 9,999,001
▪ O"n2"	numeric 9999001.1 => 9,999,001.10
▪ O"##."	0.3678 => .37, 0.3 => .3
▪ O"0.00"	0.3678 => 0.37, 0.3 => 0.30
▪ O"p"	.371 => 37.1% (percentage)
▪ O"p1"	.37 => 37.0% (percentage)

Numeric fields can have a “Currency” field property option that modifies the behavior of the numeric field in two ways: 1) expands capacity to support 10 digits left of the decimal point, 2) displayed as currency (e.g. \$9.10) on forms.

An integer field that has a minimum value and a maximum value can alternatively be presented as a dropdown menu of all integer values between and including the minimum and maximum. Simply include the “D” modifier. You can also specify a numeric interval along with the “D” modifier. For example, if a field has a minimum value of 5 and a maximum value of 25, you can present {5, 10, 15, 20, 25} in the dropdown using the syntax `{myinteger:D"5"}` where 5 is the interval.

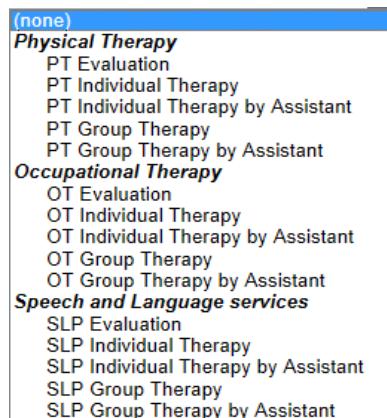
From a validation standpoint, the minimum and maximum values optionally specified in the field properties are enforced whether or not the field is presented in the form of a dropdown menu. Other types of validation checks can be implemented by creating a section action with a triggering event of “Rollback Saving Section”, and which have the “Anchor user message to first referenced field” option checked.

- **Short Text, Long Text:** You can specify a width modifier just like with the character field. But in addition, you can specify a height (in lines of text) using the “H” modifier, for example `{longtextfield:W99%H9}`. Depending on the context, the “S” modifier can also be used to modify the behavior of the text box as follows:

- If used in a profile form with a short-text field, a spell-check link will be included next to and for the field when in editing mode.
- If used in conjunction with a long text field in a document template, an “Insert Statements” link will appear even if the field presently has no public statements.
- **School Year:** Fields of type “school year” are always presented as dropdown menus. If used in conjunction with a school year type data field, the *filterspec* will be a range of years relative to the current school year. For example F"-1,+1" will include last year, the current year, and the next school year. F"-10,0" will include the last ten years and the current year, but not future years.
- **Keyword Selection:** Keyword selection fields by default are presented as dropdown menus in edit mode. If the keyword table has either a Name or Description column, values from that column are displayed in the dropdown; otherwise, the values from the Keyword column are displayed. Normally, when values from a Name or Description column are used and those values are longer than 25 characters, the description may be truncated at the first left parenthesis character to remove excessive detail. Including the N modifier prevents this truncation.

A keyword selection field can also be presented as mutually exclusive checkboxes using the “D” modifier. The checkboxes can be configured to be stacked horizontally, vertically or in a grid. Simply specify the directive as {keywordfield:DW##} where ## should be replaced by the desired number of columns. If this number is equal to or greater than the number of keywords, this will result in a horizontal layout. If the W modifier is omitted, the default number of columns is one resulting in a default vertical layout. Any number between results in a grid layout.

The “F” modifier is used to filter the keywords by specifying a filter column in the keyword table. The general format is {KeywordField:F”FilterColumn”}. The dropdown will omit keywords for which the filter column is EMPTY or FALSE. If the filter column is itself a keyword field referencing a second keyword table, in addition to omitting keywords with an empty value in the filter column, the dropdown menu is automatically presented in a hierarchical format where the filter column is assumed to identify the categories for the keywords. In the example shown below, “diagnostic code” keywords are categorized by service type.



It is also possible to filter a “dependent” keyword field dropdown based on the value selected from a “master” keyword field dropdown (also known as cascading dropdown menus) using the general format as follow:

```
{MasterField:T} {DependentField:O"MasterField" F"FilterColumn"}
```

Note that there are two keyword tables involved: the one for the master field and the one for the dependent field. For the above to work properly, two conditions must be met: 1) verify that if the O modifier is omitted, the dependent field is presented hierarchically as shown in the previous screen image, 2) the filter column must point to the keyword table used for the master field (specified with the O modifier).

The F and O modifier combination will work in other situations as well. Assume that there is a “ServiceType” field (keyword dropdown) and a “DiagnosticCode” field (keyword dropdown). The keyword table containing the service types has a column named “Category” that identifies the service category corresponding to each service type. The keyword table containing the diagnostic codes also has a column named “ServiceCategory” that identifies the service category corresponding to each diagnostic code. Then directives like those shown below can be used to link the DiagnosticCode field such that the DiagnosticCode field is dynamically filtered based on the ServiceType field.

```
{Service:LF"Category" } {DiagnosticCode:LF"ServiceCategory" O"ServiceType"}
```

- **Profile Reference:** By default, profile reference fields appear as a textbox with a lookup link next to it in edit mode. As with all text boxes, the “W” modifier can be used to change the width. When not in edit mode, the name of the referenced profile is displayed. If people (students, staff) are being referenced, the name is displayed in Last, First format by default. However, the “N” modifier can be added to the directive to override this and display the name in First Last format.

When not in edit mode, the name of the referenced profile can be displayed as a hyperlink to the actual profile by enabling the “Display as Hyperlink” field property option. When this option is enabled, the field will be displayed on any forms as a hyperlink that the user can click to open a popup window showing the corresponding profile. For example if this option is enabled for a staff profile reference field, the field will be displayed as a hyperlink that the user can click to view the staff profile.

The “F” modifier can be used to specify a filter for the profiles displayed in the lookup popup window. The format is `{profileref:reference:F"formula"}`. The formula is a PowerSchool Special Programs formula the confines the results available in the lookup popup window. The formula should be in the context of the profile type being looked up. When a filter is used within a document, in some circumstances, it is possible to embed information from the document in the formula using a placeholder enclosed with \$ characters. The placeholder can be a name of a document field linked to the same section. Alternatively, it can be the symbol “Profile” which embeds a reference to the profile of the document. For example, in a student document, the directive below filters the lookup window to any teachers of any classes the student is in.

```
{Teacher:LF"EXISTS(ClassStaffRoster,
    EXISTS(ClassStudentRoster,Student=$Profile$))"}
```

Note that the profile filter formula only applies to the popup selection window. Because the end user can enter an ID directly, it is also necessary to have a section action with a triggering event of “Rollback Saving Section” to validate the value.

Using JavaScript and the DataFormAPI

There are two basic methods of embedding JavaScript functions in document and profile forms (edit mode only): 1) form level events and 2) field level events. The JavaScript functions you write can call the built in “Data Form API” to interact with the form in a way that is largely protected against “breaking” in future versions of PowerSchool Special Programs. When configuring document templates, the JavaScript code for any given template section is maintained in the JavaScript configuration tab for that section. In profile forms, you must surround any JavaScript functions in script tags and embed that in the HTML of a section (in Version 15.0, this will be transitioned to a JavaScript tab like in document templates). In the remainder of this section, there are three main topics: form level events, field level events, and the Data Form API.

Form Level Events: If you define the functions listed below, PowerSchool Special Programs will automatically recognize and call them at the appropriate time when the user is editing documents and profiles.

- OnFormLoad() gets called once the form is fully loaded.
- OnFormSubmit() gets called when the form is being submitted for saving. The function can conditionally stop the submission of the form by returning a false value.
- OnFormGetCurRoot() is a special function that can be defined for a repeating document section for which the user will select statements from the curriculum. This function is called when the user clicks the “Select from Curriculum” button. It can return the desired curriculum root or an empty string to revert to the default behavior. It can also return a string in the format “*-message*” to display the user an alert message rather than opening the curriculum (perhaps telling the user to take some other action first). Finally, it can return “*=curroot*” which has the effect of directing the user to the specified curriculum, and hiding any curriculum dropdown which might allow the user to switch to another curriculum.

Below is a very simple example of how form submission can be stopped after prompting by the user. Once the Data Form API is covered, you will see how you can implement richer examples.

```
function OnFormSubmit()
{
    return confirm('Are you sure you want to submit?');
}
```

OnFormLoad and OnFormSubmit are also supported in service capture. For a profile form section used in service capture, the OnFormSubmit function can receive a bSchedulingServices parameter, which will be true only when the user is scheduling new services in the future (see below). When an HTML layout with this kind of OnFormSubmit is used outside of the service capture context, the bSchedulingServices parameter will be undefined.

```
function OnFormSubmit(bSchedulingServices)
{
    if (bSchedulingServices) {
        return confirm('Are you sure you want to submit scheduled services?');
    }
    else {
        return true;
    }
}
```

Field Level Events: To implement field level events, the “J” modifier is used in a field directive to specify a JavaScript function be called when the field’s value changes during editing. For fields involving a textbox, the JavaScript function will be called when the field loses the user focus (e.g. user tabs out of field). In the field directive, you specify the JavaScript function as in the example {Goal:W100H5J"api:OnChangeGoal"} where OnChangeGoal is the name of the function (note the “api:” prefix which will be discussed later). The name of the field that changed is passed as a parameter to the function so that the new value of the field can readily be obtained using the Data Form API (see example function below). Also, if the field in question has an auto-postback, the JavaScript function will be called just before the auto-postback occurs.

```
function OnChangeGoal(strFieldName) {
    alert('New value is: ' + DataFormAPI.getFieldValue(strFieldName));
}
```

Field level events can also be used in any repeating row scenario. In this case, the notification function receives a second parameter identifying the specific row of the field that changed, and in turn this “row ID” can be passed to API functions as illustrated below.

```
function OnChangeGoal(strFieldName, strRowID) {
    alert('New value is: ' + DataFormAPI.getFieldValue(strFieldName, strRowID));
}
```

In the example directive {Goal:W100H5J"api:OnChangeGoal"}, the name of the JavaScript function has an “api:” prefix. If the “api:” prefix is omitted, the function will be called with a very different parameter, namely a reference to the DOM object that triggered the change. This is a legacy approach that will not be

supported in future versions of PowerSchool Special Programs since it requires internal knowledge of details that may change in future versions. So it is recommended that the “api:” prefix always be used.

Data Form API: The API is a set of built-in functions that your JavaScript functions can call to interact with the form in a way that is unlikely to break in future versions of PowerSchool Special Programs. The API also enables many interesting scenarios that are not possible without it. It is important to note, however, that the API functions are designed to work with editable fields on the form. If you wish to use the API to write to a field but not expose the field as editable to the end user, the field directive modifiers ^ and @ provide alternative methods for that (see the list of field directive modifiers in Appendix B).

Before getting to the API functions, it is helpful to define certain common parameters that appear across many of the build in functions. These parameters are defined below:

- **strFieldName:** You pass in the name of a field using this parameter. If you are working with a multiple-value field in the context of a profile, specify strFieldName using the format *fieldname[valuename]*.
- **strRowID:** When applying the API to repeating rows, you will supply this parameter to identify a particular repeating row. If not applying the API to repeating rows, this parameter should be omitted. Note that it is generally the last parameter in each function so that it can be omitted without affecting other parameters. When using the J modifier with a repeating row field, the row ID is passed to the notification function as a second parameter making it available for use in API functions. Another way of obtaining a row ID is to add a special purpose button directive that calls a javascript function you specify that receives the row ID as its only parameter. The directive will have the format {*<ROWBUTTON>:J’*FunctionName*’L’*Label*’}. Supply the function in the JavaScript tab such that it takes a single parameter, for example: function MyRowFunction(rowID).
- **strChildTemplateID** and **strParentRowID:** Some functions operate on a repeating row layout (multiple repeating rows), and such functions include strChildTemplateID as a parameter to identify the underlying child template. In most situations strChildTemplateID is the same Child Template ID seen in the child template properties. The one exception to this is the scenario of repeating rows within repeating sections. In this case only, strChildTemplateID should be specified using a format of *OuterChildTemplateID\$InnerChildTemplateID*. This format should not be used in the context of repeating rows nested in repeating rows. If the intent is to operate on the inner repeating row layout of nested repeating rows, such functions only operate on the inner repeating rows of a single specified parent row. So in this case, the parent row ID is specified in the strParentRowID parameter. In all other situations, strParentRowID should be omitted.

The API functions are listed below. It should be noted that JavaScript is a case sensitive language so generally it is advisable to match the case of configured names in PowerSchool Special Programs.

- **DataFormAPI.getParentRowID(strRowID)**
In a nested repeating row situation, this returns the row ID of the outer row corresponding to the inner row specified by strRowID.

- **DataFormAPI.fieldExists(strFieldName, strRowID)**
Returns true/false depending on whether the specified field exists on the form as an editable field.
- **DataFormAPI.fieldIsVisible(strFieldName, strRowID)**
Returns true/false depending on whether the specified field exists on the form as an editable field that is also visible on the form. The typical reason a field might exist but not be visible is that the field is currently hidden by a JSIF directive.
- **DataFormAPI.setFocus(strFieldName, strRowID)**
Sets the user focus to the specified editable field, and if necessary, scrolls the field into view.
- **DataFormAPI.scrollFieldIntoView(strFieldName, bAnimate, strRowID)**
Scrolls the specified editable field into view. If bAnimate is true, the field will be flashed to call attention to it.
- **DataFormAPI.getFieldDataType(strFieldName, strRowID)**
Returns a string indicating the PowerSchool Special Programs data type of the specified editable field. This can be one of the following: ID, CHAR, LOGICAL, DATE, DATETIME, NUMERIC, INTEGER, SHORTTEXT, LONGTEXT, SYEAR, IMAGE, FILE, KEYWORD, or PROFILE.
- **DataFormAPI.getFieldValue(strFieldName, strRowID)**
Returns the value of the specified editable field as a JavaScript value. A table of information on how PowerSchool Special Programs data types are represented as JavaScript values is found below this list of functions. As explained in the table, some data types return a question mark string “?” if the field contains an invalid value (integer, numeric, date), and other data types require a “J” modifier to equip the field for access by getFieldvalue and setFieldValue. Also note that if the field is a lookup in combination with a non-lookup, this function will get the value of either the lookup or non-lookup depending on what is selected.
- **DataFormAPI.getIsNonLookupSelected(strFieldName, strRowID)**
When the directive specifies a lookup/non-lookup combination as in the directive below, this function returns whether the non-lookup is selected. Be sure to specify the lookup field name and not the non-lookup field name.
{LookupField:O"NonLookupField"J"api:onChangeFunction"}.
- **DataFormAPI.setFieldValue(newValue, strFieldName, strRowID)**
Sets the value of the specified editable field with a JavaScript value. You can always pass null in the newValue parameter to clear a field. A table of information on how PowerSchool Special Programs data types are represented as JavaScript values is found below this list of functions. As explained in the table, some data types require a “J” modifier to equip the field for access by getFieldvalue and setFieldValue.
- **DataFormAPI.getFieldKeywordTableValues(strFieldName, callback_function, strRowID)**
Works with a keyword fields only and produces a JavaScript object with all the field values in the

keyword table row associated with the specified field's current value. For the callback function parameter, pass the name of another function that will receive the keyword table row values once the values are retrieved. This is an important point. The getFieldKeywordTableValues method does not immediately return with the keyword table row values. Rather the request to obtain the keyword table row values is initiated. When the results are ready, the callback function is called and passed three parameters as follows: strFieldName, keyword table row values (as a single object), and strRowID if provided. For example, if the keyword table has field/column named "Category" and "keywordrow" is the parameter containing the results, then keywordrow.Category will reference the value of the category field.

For example, the following line of Javascript, obtains the Description associated with the keyword selected in the keyword field named 'KeywordField', and writes that to a character field named 'TextField'.

```
DataFormAPI.getFieldKeywordTableValues('KeywordField',
    function(strFieldName, keywordRow) {
        DataFormAPI.setFieldValue(keywordRow.Description, 'TextField');
    }
)
```

- **DataFormAPI.getChildTemplateID(strRowID)**
Returns the child template ID of the specified row. This can be used to obtain an appropriate strChildTemplateID parameter value for the functions that follow.
- **DataFormAPI.getRowIDsWithVisibleField(strChildTemplateID, strFieldName, strParentRowID)**
Returns an array of row IDs for all repeating rows for which the specified field exists and is visible. strChildTemplateID identifies the repeating rows, and strParentRowID identifies the parent row in a nested repeating row situation (see the information on parameters prior to this list of functions).
- **DataFormAPI.getVisibleFieldNamesInRow(strRowID)**
Returns an array of field names for fields that are visible in the specified row.
- **DataFormAPI.applyNotificationFunctionToFieldsInRows(strChildTemplateID, strFieldName, notificationFunction, strParentRowID)**
If you have written a field notification function for use with field level notifications in a repeating row scenario, you can use this function to call that notification function for all instances of that field across all rows in which the field exists and is visible. Typically, this is called from OnFormLoad to make sure the function is called once initially for each repeating row. The notificationFunction parameter is simply the exact name of the function (no quotes), which is case sensitive.
- **DataFormAPI.addRepeatingRow(strChildTemplateID, strParentRowID)**
Use this function to add a repeating row to a repeating row layout. "Fast Add" must be enabled for this to work. Note that the function returns the row ID of the newly added row so that the API can subsequently be used to write values to its fields.

- `DataFormAPI.selectMsgGroup(strCallbackFunctionName)`

Calling this function pops up the screen that allows the user to select a user messaging group. You pass the function the name of a “call back” JavaScript function that you write. When the user selects a messaging group, the call back function is automatically called and passed a single parameter, namely an array of User IDs that are in the message group. Note that the array of User IDs may contain User IDs for students, parents, guardians, etc. Your method can focus on staff User IDs by ignoring any User ID that does not start with the letter U. At the end of this section, you will find example functions that illustrate an interesting scenario using this method.

- `DataFormAPI.invokeLookup(strFieldName, strRowID)`

If the specified field has a lookup link, this programmatically invokes the lookup link.

- `DataFormAPI.invokeNonLookup(strFieldName, strRowID)`

If the specified field has a non-lookup link, this programmatically invokes the lookup link.

- `DataFormAPI.validateAddress(searchString, providedAddress)`

Calling this function pops up the screen that allows the user to select address. You pass address string to be searched against the MAR API and providedAddress in given format to be displayed on the modal popup.

```
var providedAddress= {
    "StreetName": "CHESAPEAKE STREET",
    "StreetName2": "",
    "Quadrant": "SE",
    "City": "WA",
    "State": "DC",
    "Zipcode": "20032",
    "Ward": "8"
}
```

Pop up contains two buttons ‘Use recommended address’ and ‘Use the address I provided’. Click events for both these buttons should be handled in ‘model’. If user selects a recommended address it will be available in a global variable called selectedAddress.

```
SelectedAddress:
{
    "MarId": "24445",
    "FullAddress": "425 CHESAPEAKE STREET SE",
    "AddressNumber": "425",
    "AddressNumberSuffix": null,
    "StreetName": "CHESAPEAKE",
    "Quadrant": "SE",
    "Zipcode": "20032",
    "Status": "ACTIVE",
    "Ward": 8,
    "StateCode": "11",
    "StateAbbreviation": "DC"
}
```

Signature of the click events:

```
UseRecommendedAddress_OnClick = function() {
```

```
//Set address fields from SelectedAddress object.  
//Operations to be done on click of 'Use recommended address' button.  
DataFormAPI.closeModalPopup();  
    return false;  
};  
  
UseProvidedAddress_OnClick = function() {  
//Operations to be done on click of 'Use the address I provided' button.  
DataFormAPI.closeModalPopup();  
    return false;  
};
```

Field Value Representation in JavaScript

When using the API, it is important to know how PowerSchool Special Programs field values are represented as JavaScript values in the API. These are the types of values that will be returned from getFieldValue and that you will pass to setFieldValue.

PowerSchool Special Programs Data Type	JavaScript Value
ID, Character, Short Text, Long Text	JavaScript string
Logical	JavaScript bool (true/false) or null for empty value
Date, Date / Time	JavaScript Date object or null for empty value (see http://www.w3schools.com/jsref/jsref_obj_date.asp). Note: getFieldValue will return “?” string if field contains an entry that is not a valid date value.
Numeric	JavaScript float or null for empty value. Note: getFieldValue will return “?” string if field contains an entry that is not a valid numeric value.
Integer	JavaScript int or null for empty value. Note: getFieldValue will return “?” string if field contains an entry that is not a valid integer value.
School Year	JavaScript int (containing the initial year) or null for empty value
Image	Javascript string containing binary data in base 64 format or null for empty value.
File	Not supported by API.
Keyword	Javascript string containing the keyword, or null for empty value. To use a keyword field with getFieldValue and setFieldValue, the field must have a “J” modifier. Although the “J” modifier is normally used to specify a JavaScript notification function, that is not necessary. A plain “J” modifier is adequate.

Profile	Javascript string containing the profile ID or null for empty value. Such values may also be in the format <i>ID (Name)</i> . The name part is treated as informational and is ignored, for example, by setFieldValue. To use a profile reference field that is presented as a dropdown menu with getFieldValue and setFieldValue, the field must have a “J” modifier. Although the “J” modifier is normally used to specify a JavaScript notification function, that is not necessary. A plain “J” modifier is adequate.
---------	---

Exposing Value of PowerSchool Special Programs Formulas to the API: If you wish to access the value of a PowerSchool Special Programs formula from JavaScript, you can use the `{=jsvarname=formula}` variation of the basic formula directive. This directive exposes the value of the formula to JavaScript in a format consistent with the Data Form API. The value is accessed from JavaScript code via a reference to `window.form$jsvarname`. You can also configure a directive with a formula in the context of the Globals profile using the format `{=@$jsvarname=globalsformula}`.

Example Scenario: Consider a scenario where a document has repeating rows that allow the user to select any number of staff. But you also want to provide the user a button that brings up the user message group selection window and allows the user to select a message group. Upon selection, the message group staff users are inserted into the repeating rows, but in a way that avoids duplicate staff.

Team Staff

			CASE (Frank Casemanager)	(ID) lookup
			GREENWOOD (Susan Greenwood)	(ID) lookup
Insert Staff From Messaging Group				

The following assumptions are in place:

- The child template has a child template ID of ‘StaffCT’.
- The staff profile reference field within the child template is named ‘StaffRef’.
- The repeating row layout utilizes the “fast add” approach.

The following HTML markup reproduces the editable list of staff shown above including the button.

```
<p><b>Team Staff</b></p>
<br />
```

```
{^StaffCT}
{#IFEDIT}
<button onclick="DataFormAPI.selectMsgGroup('InsertStaff');return false;">Insert Staff From Messaging Group</button>
{#ENDIF}
```

Note that the button calls the API function with a callback function named ‘InsertStaff’. If a button is specified this way in the HTML, it is important that the onclick method returns false at the end, and that it only appear in edit mode. The definition of the InsertStaff method is shown below. Note that this also uses a number of API calls:

```
function InsertStaff(astrUserID) {
    var strChildTemplateID = 'StaffCT';
    var strFieldName = 'StaffRef';
    var astrRowID = DataFormAPI.getRowIDsWithVisibleField(strChildTemplateID, strFieldName);
    // Process each user id, but only staff users beginning with letter U
    for (var idxUser = 0; idxUser < astrUserID.length; idxUser++) {
        var strUserID = astrUserID[idxUser];
        var idxOpenParen = strUserID.indexOf('(');
        if (idxOpenParen > 0) strUserID = strUserID.substr(0, idxOpenParen);
        strUserID = strUserID.trim().toUpperCase();
        var bFound = false;
        var strRowID_Empty = null;
        // Search for staff ID in current rows, but also find first empty row in case
        //      it can be used if staff ID is not found.
        for (var idxRow = 0; idxRow < astrRowID.length; idxRow++) {
            var strUserID_Check = DataFormAPI.getFieldValue(strFieldName, astrRowID[idxRow]);
            idxOpenParen = strUserID_Check.indexOf('(');
            if (idxOpenParen > 0) strUserID_Check = strUserID_Check.substr(0, idxOpenParen);
            strUserID_Check = strUserID_Check.trim().toUpperCase();
            if (strUserID == strUserID_Check) {
                bFound = true;
                break;
            }
            else if (strUserID_Check == '' && !strRowID_Empty) {
                strRowID_Empty = astrRowID[idxRow];
            }
        }
        if (!bFound) {
            // If staff ID was not found, then use empty row if available, otherwise insert new row.
            if (!!strRowID_Empty) {
                DataFormAPI.setFieldValue(astrUserID[idxUser], strFieldName, strRowID_Empty);
            }
            else {
                var strRowID = DataFormAPI.addRepeatingRow(strChildTemplateID);
                DataFormAPI.setFieldValue(astrUserID[idxUser], strFieldName, strRowID);
            }
        }
    }
}
```

In the context of documents only, several functions are available that are not part of the API but which nonetheless can be called to invoke one of the save or cancel editing buttons. These functions are as follows:

- doSaveDoneEditing(): This function is equivalent to clicking the “Save, Done Editing” button.

- `doSaveContinueEditing()` or `doSaveContinueEditing(bSkipAttemptComplete)`: This function is equivalent to clicking the “Save, Continue Editing” button. Normally, clicking the “Save, Continue Editing” button also checks whether the section meets all completion requirements, and if so, marks it as complete. To avoid checking for section completion, the `doSaveContinueEditing` function takes an optional `bSkipAttemptComplete` parameter for which you can pass true to skip the behavior (e.g. `doSaveContinueEditing(true)`).
- `doCancelEditing()`: This function is equivalent to clicking the “Cancel, Editing” button.

Conditional Form Logic: #JSIF Directive

The #JSIF (javascript if) directive is designed to conditionally include or exclude regions of a form using a method that is very fast and responsive to end users. #JSIF supports a conditional expression syntax that is lean and performant on both the server and in the browser. There is a similar #IF directive, described in the next section, that can evaluate much more complex conditions than #JSIF, but for performance reasons, should only be used when #JSIF cannot meet the logical requirements.

The basic syntax of #JSIF is given below. Nesting and #ELSE is supported, but #ELSEIF is not supported at the time of writing.

```
{#JSIF fieldlogic, optionname1=value1, optionname2=value2, ... }
```

Content

```
{#ENDIF}
```

As the simplest case, “*fieldlogic*” in the syntax above can be the name of a single field. If the field is a logical field, the content is shown if and only if the logical field is true. The logical field name can be prefixed with a ~ character to reverse the logic and show the content only if the logical field is false. For other data types, the content is shown if and only if the field has a value, or you can precede the field name with a ~ character to show the content if the field is EMPTY. You can also reference a keyword field with the format *keywordfield.keyword* to show the content only if a specific keyword is selected. To specify multiple keywords, use the format *keywordfield.keyword1/keyword2/keyword3*. Alternatively, use the format *keywordfield.logicalfield* where *logicalfield* is the name of a logical field in the keyword table that identifies those keywords with true values in the column. If the logical field name is coincidentally also the name of a keyword, the logical field name takes precedence.

[New in 21.11.1.0] In documents, you can reference DocStatus using formats such as DocStatus.Review, DocStatus.Draft/Review, or ~DocStatus.Final.

“*fieldreference(s)*” can also be a list of field references separated by either an ampersand (&) to designate “AND” logic or a pipe (|) to designate “OR” logic. Any field reference can be preceded with a ~ character to negate the logic for that one field. For example {#JSIF checkbox1&checkbox2&~checkbox3} will show the content if checkbox1 and checkbox2 are checked and checkbox2 is not checked. If you need to mix “AND” and “OR” logic, you can use parentheses to nest the logic.

As shown in the syntax, the JSIF directive can include comma-separated options, as follows:

`collapse=true` (By default, the HTML formatting is rendered invisible using the CSS visibility style, but it still takes up space on the form. Enabling this option causes the HTML formatting to collapse when not enable using the CSS display=none style).

`cssclass=somecssclass` (If specified, this applies a css class to the enclosing tag as described for the tag option.)

`editonly=true` (If this option is enabled, the HTML formatting will only be visible in edit mode.)

`not=true` (Obsolete, but still supported for backwards compatibility)

`simulate=if` (Specifies that the JSIF directive should simulate the behavior of an “{#IF formula}” directive. In this case, the JSIF will render neither any enclosing tags nor any javascript triggering behavior, such that it behaves identically to the #IF directive. In fact, suitable instances of {#IF formula} can be converted directly to {#JSIF expression, simulate=if} with no other changes required. A section that has many #IF statements, especially in repeating rows, can run noticeably faster if they are converted to JSIF with simulate=if. Note that “simulate=ifviewor” and “simulate=ifeditor” are also supported and simulate #IFVIEWOR and #IFEDITOR respectively.

`static=true` (This specifies that the content will not change state based on immediate user input. In effect, this is an optimization that instructs the system to not generate any javascript triggering behavior).

`tag=span` (By default the HTML formatting is enclosed in a div tag which is used to control the visibility of the content. This option allows a different tag to be specified, such as span or p.

`whitespace=true` (When this option is enabled, the space that the content normally occupies when shown will remain as white space when the content is not shown. This is particularly use in conjunction with the span tag to prevent the form from shifting around when the content is displayed or hidden.)

`insertafter=keyword` (This can be used immediately below a keyword field presented as a set of mutually exclusive checkboxes. The JSIF content will then be moved below the checkbox for specified keyword. The entire JSIF statement would typically be something like {#JSIF SomeKeywordField.SomeKeyword,insertafter=SomeKeyword} so that the content appears below a particular checkbox after clicking it.

The directive `{-field1,field2,field3...}` can be very useful in conjunction with JSIF. This directive, available only in documents, resets the specified fields (comma-separated list of field names) to their default values, but only if the fields are not included on the section/form at the time the user submits the form. This directive is compatible with the #IF and #JSIF directives. In the example below, the

{-OtherText} directive ensures that the OtherText field is reset to its default value if the Service field is on the form but the OtherText field is not:

```
{#IF StudentNeedsService}
    <label>Service:{TypeOfService}</label>
    {#IF JSIF Service.Other,tag=span}
        <label>Other: {OtherText}</label>
    {#ENDIF}
    {-OtherText}
{#ENDIF}
```

Conditional Form Logic: #IF Directive(s)

In addition to #JSIF, which is designed to operate dynamically both in the browser as well as on the server, there are a number of #IF style conditional directives that are executed strictly on the server. Below are the most commonly used #IF Directives(s), however, there are additional ones documented in appendix B.

Conditional Directives	Explanation
{#IFEDIT} <i>HTML Formatting</i> {#ENDIF}	The HTML formatting between the {#IFEDIT} and {#ENDIF} directives is only included when the form is used in edit mode.
{#IFVIEW} <i>HTML Formatting</i> {#ENDIF}	The HTML formatting between the {#IFVIEW} and {#ENDIF} directives is only included when the form is used in view mode.
{#IF formula} HTML Formatting {#ENDIF}	The HTML formatting between the {#IF <i>formula</i> } and {#ENDIF} directives is only included if the formula evaluates to true. If this is used in a profile form section, the formula is expressed in terms of the profile fields. If this is used in a document template section, the formula is expressed in terms of the document fields.
{#IFVIEWAND formula} HTML Formatting {#ENDIF}	The HTML formatting between the {#IFVIEWAND <i>formula</i> } and {#ENDIF} directives is included if the form is in view mode and if the formula evaluates to true. [New in 20.11.4.0] There is also a {#IFVIEWAND [^] <i>formula</i> } version of this directive

	(i.e. with a caret character) that can be used in a child profile form to allow specification of the formula in the context of the parent profile. This allows the directive to behave as expected when the form is used to add a new child profile.
{#IFVIEWOR formula} HTML Formatting {#ENDIF}	The HTML formatting between the {#IFVIEWOR <i>formula</i> } and {#ENDIF} directives is included if the form is in view mode (formula ignored in this case), or if the form is in edit mode and the formula evaluates to true. [New in 20.11.4.0] There is also a {#IFVIEWOR^ <i>formula</i> } version of this directive (i.e. with a caret character) that can be used in a child profile form to allow specification of the formula in the context of the parent profile. This allows the directive to behave as expected when the form is used to add a new child profile.
{#IFEDITAND <i>formula</i> } HTML Formatting {#ENDIF}	The HTML formatting between the {#IFEDITAND <i>formula</i> } and {#ENDIF} directives is included if the form is in edit mode and if the formula evaluates to true. [New in 20.11.4.0] There is also a {#IFEDITAND^ <i>formula</i> } version of this directive (i.e. with a caret character) that can be used in a child profile form to allow specification of the formula in the context of the parent profile. This allows the directive to behave as expected when the form is used to add a new child profile.
{#IFEDITOR <i>formula</i> } HTML Formatting {#ENDIF}	The HTML formatting between the {#IFEDITOR <i>formula</i> } and {#ENDIF} directives is included if the form is in edit mode (formula ignored in this case), or if the form is in view mode and the formula evaluates to true. [New in 20.11.4.0] There is also a {#IFEDITOR^ <i>formula</i> } version of this directive (i.e. with a caret character) that can be used in a child profile form to allow specification of the formula in the context of the parent profile. This allows the directive to

	behave as expected when the form is used to add a new child profile.
{#IF_RO formula} HTML Formatting {#ENDIF}	If the formula evaluates to true, then any fields in the content are forced to be read only. [New in 20.11.4.0] There is also a {#IF_RO^ formula} version of this directive (i.e. with a caret character) that can be used in a child profile form to allow specification of the formula in the context of the parent profile. This allows the directive to behave as expected when the form is used to add a new child profile. Tip: If you want a portion of a section to be read-only for all staff users (versus ADMIN/CONSULTANT), use {#IF_RO User IS NOT EMPTY). Note that the User keyword is not empty only for staff users.
{#IF_USER <i>staff_formula</i> } HTML Formatting {#ENDIF}	If the current user meets the criteria defined in the staff-based formula (or if the user is the ADMIN or CONSULTANT), the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents). Note that the "User" keyword is unnecessary and not supported in user formulas like this.
{#IF_EDITANDUSER <i>staff_formula</i> } HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user meets the criteria defined in the staff-based formula (or if the user is the ADMIN or CONSULTANT), the HTML formatting is included.
{#IF_USER_RO <i>staff-formula</i> } HTML Formatting {#ENDIF}	If the current user meets the criteria expressed in the staff-based formula, the contents will be read-only.
{#IF_ADMINCONSULTANT} *HTML Formatting {#ENDIF}	If the current user is an ADMIN user, CONSULTANT user or staff user with the 'Access Admin Form Content' privilege, the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents).

{#IF_ADMIN} HTML Formatting {#ENDIF}	If the current user is an ADMIN user or a staff user with the ‘Access Admin Form Content’ privilege, the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents).
{#IF_CONSULTANT} HTML Formatting {#ENDIF}	If the current user is a CONSULTANT user, the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents).
{#IF_EDITANDADMINCONSULTANT} HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user is an ADMIN user, CONSULTANT user or staff user with the ‘Access Admin Form Content’ privilege, the HTML formatting is included.
{#IF_EDITANDADMIN} HTML Formatting {#ENDIF}	If the form is in edit mode and the current user is an ADMIN user or staff user with the ‘Access Admin Form Content’ privilege, the HTML formatting is included.
{#IF_EDITANDCONSULTANT} HTML Formatting {#ENDIF}	If the form is in edit mode and the current user is a CONSULTANT user, the HTML formatting is included.
{#IF_EDITANDSTAFFUSER} HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user is a staff user, the HTML formatting is included.
{#IF_USERORFINAL <i>staff-formula</i> } HTML Formatting {#ENDIF}	If the current user meets the criteria defined in the staff-based formula (or if the user is the ADMIN or CONSULTANT), the HTML formatting is included. This is only available for document templates and only when the current section has the “Always Regenerate When Not Final” property.

<pre>{#IFFIELD <i>fieldname</i>} HTML Formatting {#ENDIF}</pre>	This directive is available only in profile forms and is intended to provide support for field level security. If you wish to exclude part of the form section markup if a particular field is unavailable due to field level security, enclose that part of the markup with this directive. This will conditionally include the markup only if the specified field is available.
<pre>{#IF_NO_PREVIEW} Html formatting to hide {#ENDIF}</pre>	The user caseload customize column screen displays student profile forms with as many fields as possible in the form of checkboxes. In some cases, this may result in unwanted content appearing on that screen. The new directive syntax below can be used to suppress such content on that screen and similar screens
<pre>{#IF_STAFFUSER} HTML Formatting {#ENDIF}</pre>	[New In 21.11.2.0] This directive, which is available only in profiles (not documents), includes the HTML formatting if and only if the current user is a staff user.

Each {#IF...} style directive has a corresponding {#ENDIF} directive and may also have {#ELSE} or {#ELSEIF...} in between. In all of these directives, the keyword immediately after the pound sign (e.g. “IFEDIT” in {#IFEDIT}) must be in upper case. Any word(s) immediately after #ENDIF are ignored so that you can document what the #ENDIF corresponds to, for example {#ENDIF-IFVIEW}.

If is useful to compare the #IF style directives with #JSIF. In fact, you will want to use #JSIF instead of #IF whenever possible because #JSIF is more performant than #IF. But it is not always possible. There are three main differences to consider:

- The #JSIF syntax for logical expressions is much simpler than the PowerSchool Special Programs formulas used in #IF directives. If the logic you need is too complex for #JSIF and requires a full PowerSchool Special Programs formula, then you will need to use one of the #IF variants.
- The #JSIF directive renders a specified tag around the content (e.g. div, p, span, etc) whereas #IF directives do not.
- Evaluating #JSIF expressions is more performant than evaluating PowerSchool Special Programs formulas for #IF directives. Use #JSIF wherever possible. The simulate keyword can be used to specify that the #JSIF directive should simulate the behavior of an “{#IF formula}” directive. In this case, the #JSIF directive will render neither any enclosing tags nor any javascript triggering behavior, such that it behaves identically to the #IF directive. In fact, suitable instances of {#IF formula} can be converted directly to {#JSIF expression, simulate-if} with no other changes

required. A section that has many #IF statements, especially in repeating rows, can run noticeably faster if they are converted to JSIF with simulate=if. Note that “simulate=ifviewor” and “simulate=ifeditor” are also supported and simulate #IFVIEWOR and #IFEDITOR respectively.

Auto-Postback: If you have determined that #JSIF cannot be used for a situation, but still want the form to exhibit dynamic changes as soon as the user changes the value of a field, such as a checkbox, you can trigger an auto-postback. Auto-postback is exactly equivalent to the user clicking “Save & Continue” immediately after changing the value of a particular field (e.g. clicking a checkbox). To configure a field to initiate auto-postback, introduce the “A” modifier into the directive for that field. There are several considerations:

- Auto-postback is much slower than #JSIF. The form displays a “Please wait...” message to the user, but nonetheless, too many auto-postbacks in a document template can make the user experience clunky. This is why #JSIF is preferred wherever possible.
- Unlike #JSIF, auto-postback actually saves the data entered so far. The positive side of this is that frequent saving makes it less likely the user will ever lose data.

Pop-up Help Definitions and Guides

To set up pop-up help definitions or guides in a document template, first you must add a style and two JavaScript functions to the style sheet:

First add the following style definition to the style sheet:

```
.DOC_GUIDE {position:absolute; display:inline; padding:5px; border:4px ridge #0000DD;
background:#EEEEEE; font-size:10pt; z-index:9; border-radius: 10px;}
```

Next add the following javascript block to the end of the style sheet below any style block(s):

```
<script language=javascript>
function ToggleGuide( idGuide)
{
    var objGuide = document.getElementById(idGuide);
    objGuide.style.display=objGuide.style.display == 'none' ? "block" : 'none';
    return false;
}
function HideThisGuide(idGuide)
{
    var objGuide = document.getElementById(idGuide);
    if (objGuide) {
        objGuide.style.display='none';
    }
    else {
        alert(idGuide + ' not found.');
    }
}</script>
```

```

        return true;
    }
</script>

```

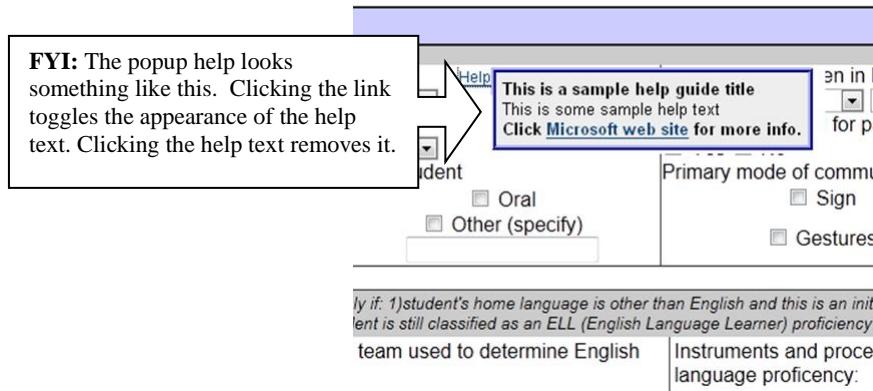
Next add a block like the following for each help popup:

```

{#IFEDIT}
<a href="#" onclick="return ToggleGuide('GUIDE_XXXX')>Help</a>
<p id=GUIDE_XXXX class=DOC_GUIDE style="display:none" onclick="return HideThisGuide(this.id)">
<b>This is a sample help guide:</b><br/>
This is some help text<br/>
<b>Click <a href="http://www.microsoft.com" target=new>Microsoft web site</a> for more info:</b><br/>
</p>
{#ENDIF}

```

But be sure to replace “GUIDE_XXXX” in each block with a unique identifier (e.g. GUIDE_PresentLevels) and customize the help content.



Note that it is very helpful to use style sheet entries to ensure a consistent appearance to all of your help guides.

Associating a Help Guide with a Field: If you want a help guide to disappear when the user changes the value of the field, first make sure you assign the guide identifier to use the format GUIDE_FieldName. Next introduce an additional JavaScript function to the style sheet as follows:

```

<script language=javascript>
function HideThisFieldGuide(strFieldName)
{
    HideThisGuide ('GUIDE_' + strFieldName);
}
</script>

```

Finally, reference the HideThisFieldGuide function in the field directive as in the following example:

```
{FieldName:J"api:HideThisFieldGuide"}
```

Using JSIF to Automatically Invoke a Help Guide: You can place a help guide inside of a JSIF block to cause the help guide to automatically appear when the conditions are met. Note the use of editonly=true in the JSIF directive which ensures the help guide only appears in edit mode. Also note the use of tag=span which ensures that the help guide appears inline with the content.

```
{#JSIF formula,editonly=true,tag=span}
<span id="GUIDE_FieldHelp" class="DOC_GUIDE" onclick="return HideThisGuide(this.id)">
    Help Guide Text Goes Here
</span>
{#ENDIF}
```

Assessment Directives

Assessment directives are preceded with an exclamation point, and allow scores and information from the assessment repository and/or curriculum to be included in a form. These directives have a general format as follows:

```
{!CurRoot=SAMPLE;name1=value1;name2=value2}
```

To include a curriculum statement description in a form, use a directive with a format like the following:

```
{!CurRoot=SAMPLE;Label=10101}
```

To include assessment total scores of various types, use a directive with a format like the following:

```
{!CurRoot=SAMPLE;AssessAdminID=100;ScoreType=Percentage}
```

```
{!CurRoot=SAMPLE;AssessmentName=Math Test Form B;ScoreType=Percentage,HistoryYear=2005}
```

AssessAdminID is the numeric ID of the assessment administration. Alternatively you can use AssessmentName, in which case, the system determines the latest assessment administration for the named assessment that the student participated in. In this case, the HistoryYear parameter can be included to limit it to particular year (more on this below). ScoreType is one of {RawPoints, MaxRawPoints, Percentage, RubricLevel, RubricSymbol, Percentile, NCE, GE, Stanine, Scaled, GAP and Other}. It can also be a combination of one of the numeric score types and either RubricLevel or RubricSymbol (example: ScoreType=Percentage/RubricLevel}.

The HistoryYear parameter is very useful in conjunction with the AssessmentName parameter. If the HistoryYear parameter is omitted, the latest score is retrieved even if it is from years ago. The HistoryYear parameter can be absolute, for example, 2005 refers to school year 2005/2006. Or if the HistoryYear parameter is zero or negative, it is assumed to be relative to the current year (e.g. 0, -1, -2, -4, -5).

There is an additional “Filter” parameter that can be used in conjunction with the AssessmentName parameter to selectively select scores based on assessment repository filter/analytical fields. This takes the form “Filter=FieldName=Value” where FieldName is the name of an filter field in the assessment

repository and Value is an appropriate value for that field. The example below only pulls out a score for which the student was in grade 05 (at the time of taking the assessment).

```
{!CurRoot=SAMPLE;AssessmentName=Math Test Form B;ScoreType=Percentage, Filter=Grade=05}
```

To include assessment section scores of various types, use a directive with a format like the following:

```
{!CurRoot=SAMPLE;AssessAdminID=100;SectionID=Reading;ScoreType=Percentage}
```

```
{!CurRoot=SAMPLE; AssessmentName=Math Test Form B;SectionID=Reading;ScoreType=Percentage}
```

In this case, SectionID is the exact user-defined ID of the section, and ScoreType can be the same types as for total scores.

In some cases, it might be useful to have the score and its label formatting not appear at all if the score is not available for the student. In this case, you can specify a format in the directive, such as Format=<p>Score:\$</p>. The \$ character is a placeholder for where the score appears in the format. If \$ is omitted, the score appears after the label. Note that the format is only displayed if the specified score or value is actually found for a student; therefore, the format is very useful for making sure there are no labels with empty score values.

To include assessment curriculum statement scores of various types, use a directive with a format like the following:

```
{!CurRoot=SAMPLE;AssessAdminID=100;Label=10101;ScoreType=Percentage}
```

```
{!CurRoot=SAMPLE; AssessmentName=Math Test Form B;Label=10101;ScoreType=Percentage}
```

In this case, SectionName is the exact name of the section, and ScoreType is one of {RawPoints, MaxRawPoints, Percentage, RubricLevel}.

To include the date when an assessment was administered or taken, use a directive with a format like the following:

```
{!CurRoot=SAMPLE;AssessAdminID=100;Date=Y}
```

```
{!CurRoot=SAMPLE;AssessmentName=Math Test Form B;Date=Y}
```

When displaying numeric scores, you can also use the “Translate” parameter to include an arbitrary translation of score values into alphanumeric labels. The example below sets up a letter grade translation. Note that the labels must be listed from lowest score to highest score.

```
{!CurRoot=SAMPLE;AssessmentName=Math Test Form B; Translate=F,50:D,70:C,80:B,90:A}
```

The following brings all this together with some examples:

Score Category	Examples
Total Score	{!CurRoot=SAMPLE;AssessAdminID=100;ScoreType=Percentage} {!CurRoot=SAMPLE;AssessAdminID=100;ScoreType=Rubric} *Requires assessment rubric
Section Score	{!CurRoot=SAMPLE;AssessAdminID=100;SectionID=Reading;ScoreType=Percentage} {!CurRoot=SAMPLE;AssessAdminID=100;SectionID=Reading;ScoreType=Rubric} *Requires assessment rubric
Curriculum Statement Score	{!CurRoot=SAMPLE; AssessmentName=Math Test Form B;Label=10101;ScoreType=Percentage} {!CurRoot=SAMPLE; AssessmentName=Math Test Form B;Label=10101;ScoreType=Rubric} *Uses curriculum rubric

Profile Grid Directives

Profile grid directives allow you to display child profiles of the current profile being viewed in a table/grid format. The directive can be used from within a profile form section or within a document template section. The display is always view only. When used in a document template, the information will always reflect the last time the section was saved or finalized since the output of the section is generally rendered and stored in the database. The general syntax is given below:

```
{%childprofiletypename: column1_expr:"column1 title", column2_expr: "column2 title",...(logicalfilter)[sort1_expr, sort2_expr, ...]-modifiers}
```

The *childprofiletypename* can be the name of either a standard child profile type or a many-to-many (M-M) child profile type. The column expression designates what data should appear in the table, and what the column names should be. The other two major components – logical filter, and sort value list – are optional.

In the case that the child profile type is a M-M profile type, the context for the columns, filter and sort will be the other parent profile type. For example, when displaying the class roster in a student profile, the fields that you would show would be from the classes that the student is enrolled in. However, since M-M profile types can also have data fields, it is also possible to force the “context” profile type to be the actual M-M profile type. This is done by prefixing the child profile type name with =, as shown below:

```
{%=<childprofiletypename: column1_expr:"column1 title", column2_expr: "column2 title",...(logicalfilter)[sort1_expr, sort2_expr, ...]}
```

As a special case, it is also possible to apply the profile grid directive to another top-level profile type that is related to the current profile type via a profile reference field that is marked as a quick search field. The syntax for this is shown below:

```
{%=<toplevelprofiletypename.profilereference: column1_expr:"column1 title", column2_expr: "column2 title",...(logicalfilter)[sort1_expr, sort2_expr, ...]}
```

Sort expressions can be appended with **DESC** to indicate that records should be sorted in descending order. Note that a DESC can appear in each sort expression (if it is absent then ascending sort for that expressions will be used) and must appear within the square brackets.

Valid modifiers are:

- V – Indicates that view icons should be presented when applicable to the user to allow viewing child records. This is not available for M-M child profile types
- E – Indicates that view/add/edit/delete icons should be presented to the user to allow viewing, adding, editing and deleting child records if authorized by user security. This is not available for many-to-many child profile types.
- I – Enables precise control of which icons may show. The “I” modifier may be followed by any of V (view), A (add), E (edit) or D (delete) to show the respective icons if authorized by user security. This is not available for many-to-many child profile types.
- R## – Establishes a maximum number of rows to display (replace ## with the specific number).

For example:

```
{%=Caseload: Student:Student [Profile_Created_On DESC, Student.ID]}
```

The output that is rendered is a generic HTML table with known CSS classes which can be defined in the style sheet for the profile form sections or document template. The format is as follows:

```
<table class="CHILDGRID_TABLE">
<tr class="CHILDGRID_HDRROW">
    <th class="CHILDGRID_HDRCELL">column1_title</th>
    <th class="CHILDGRID_HDRCELL">column2_title</th>
</tr>
<tr class="CHILDGRID_ROW">
    <td class="CHILDGRID_CELL">row1 column1</td>
    <td class="CHILDGRID_CELL">row1 column2</td>
</tr>
<tr class="CHILDGRID_ROW">
    <td class="CHILDGRID_CELL">row2 column1</td>
    <td class="CHILDGRID_CELL">row2 column2</td>
</tr>
</table>
```

Example

```
{%ClassStudentRoster: ID:"Class ID", Name:"Name", Location:"Location", ClassType:"Class Type"(Language=English)[Name]}
```

ClassStudentRoster - this directive references the M-M profile type

ID:"Class ID" – the first column of the grid, will be titled *Class ID* and evaluates to the ID of the related class

Name:"Name" – the second column of the grid, will be titled *Name* and evaluates to the name of the related class

Location:"Location" – the third column of the grid, will be titled *Location* and evaluates to the location of the related class

ClassType:"Class Type" – the forth column of the grid, will be titled *Class Type* and evaluates to the class type of the related class

(Language=English) – filter that limits the list to classes that only have been marked as English
[Name] – sort criteria that indicates that the results should be sorted by class name

The profile grid directive can also be used to present document data in a student profile section. Data from both top level document templates and child templates can be included. The first two examples below illustrate presenting data from a top level document template (identified by DocTemplateID) and from a child template (identified by ChildTemplateID) respectively. You can also present data from group intervention documents linked to the student, and this is shown in the third example. The -V modifier in the examples below specify that view icons should be included to allow the user to drill down into the actual documents.

- { %DocTemplateID: DocHistoryYear:"Year", DocStatus: "Status"
 (DocHistoryYear=HistoryYear) [Document_Created_On]-V }
- { %DocTemplateID.ChildTemplateID: DocHistoryYear:"Year", DocStatus: "Status"
 (DocHistoryYear=History Year) [Document_Created_On]-V }
- { %GroupPlan.Students: DocHistoryYear:"Year", DocStatus: "Status"
 (DocHistoryYear=History Year) [Document_Created_On]-V }

It is also possible to create a profile grid that displays data from events (e.g. Student Events tab), as shown below. The -V modifier is not supported in this case.

- { %Events_ : EventDateTime: "Date/Time", Subject: "Subject", SourceUserIDName : "User ID/Name" }

Integrating profile constraints into profile grid: In some use cases, it may be useful to have a checkbox column in the profile grid with a button in the column header that allows the end user to apply a specified profile constraint to the checked profiles. To configure this, follow the steps below:

1. Set up a profile constraint for the profile type displayed in the profile grid. The profile constraint must be set up to evaluate when the user explicitly clicks a button or as a bulk operation. The constraint must also have a constraint action of “Set Field Values”.
2. The profile grid column must be an expression of type logical (true/false). This will control whether the checkbox is enabled or not for that row, which allows for conditionally enabling or disabling each row checkbox.

3. The column title must embed the name of the constraint using either of the following two formats below. The column title becomes the label of the button. If just the constraint name is provided, the constraint name is used to label the button.

[*ConstraintName*]

Column Title[*ConstraintName*]

Document Forms and Profile Grids: Profile grids can be used in document forms, but there are limitations. Icons are not supported at all.

Configuring Packages

Chapter

9

Frequently Asked Questions

What are packages?

Packages are a pre-defined set of modules that can be licensed in customer databases, specifically: Service Capture, SpecialEducation, RTI, Section 504, ELL, and Gifted Talented. The model configuration team, when developing a standard state or province model, can tag certain content (document templates, reports, etc.) with packages such that only customers licensed to use those packages can access that content.

How does the model configuration team indicate what packages a particular state/province model currently supports?

The model configuration team should license the supported packages (and only the supported packages) in the relevant model database (i.e. XXMODEL, XXMODELCR, XXMODELAPP).

Can the model configuration team define entirely new packages?

No, the package definitions are embedded in the software itself.

What happens when a new customer is instantiated from a model database with respect to packages?

By default, a newly cloned customer database will have the same licensed packages that are set in the model itself. Unlicensed packages should then be removed. Packages should not be licensed to a customer that are not set in the model itself because that will have no effect and will be misleading.

How do I see what packages are licensed in a particular database from the front end?

The ability to see packages from the front end is only possible if configured in a globals profile form, and that is where it would be viewed if configured. The {#PACKAGES} directive in a globals profile form shows the packages licensed in the current database.

What content can be associated with packages?

The model configuration team can associate certain types of content with packages within a standard state/province model. Content not associated with any packages at all are available to all customers using the model. Once content is associated with one or more packages, the content may be hidden for customers that do not have at least one of the associated packages. The following types of content may be associated with packages.

1. **Document Templates:** Set using document template properties for a single document template or, for bulk setting, use Administration > Configuration > Document Templates > More > Set Template Properties.
2. **Keyword Tables:** Set using keyword table properties.
3. **Standard Reports:** Applies only to standard reports that are under configuration management only. Set using report properties for a single report, or for bulk setting, use Reporting > Standard Reports > Utilities > Apply Option to Reports.
4. **Profile Form Sections:** Set using profile form section properties.

What effect do packages have on users?

It is important to note that content (e.g. document templates) not associated with any packages at all are available to all customers using the model. It is only when content is associated with one or more packages that the content may become unlicensed to customers not having those packages.

- Staff users will not see unlicensed content at all, for example, unlicensed document templates will not appear in the new document dropdown menu.
- System administrators will not see the unlicensed content in the administrative area, for example, they will not see unlicensed keyword tables (to view/edit) or document templates (to set security for).
- Consultant users will see the unlicensed content listed in the configuration UI. The unlicensed content will have a “no edit” icon instead of a magnifier icon.

Advanced Reporting

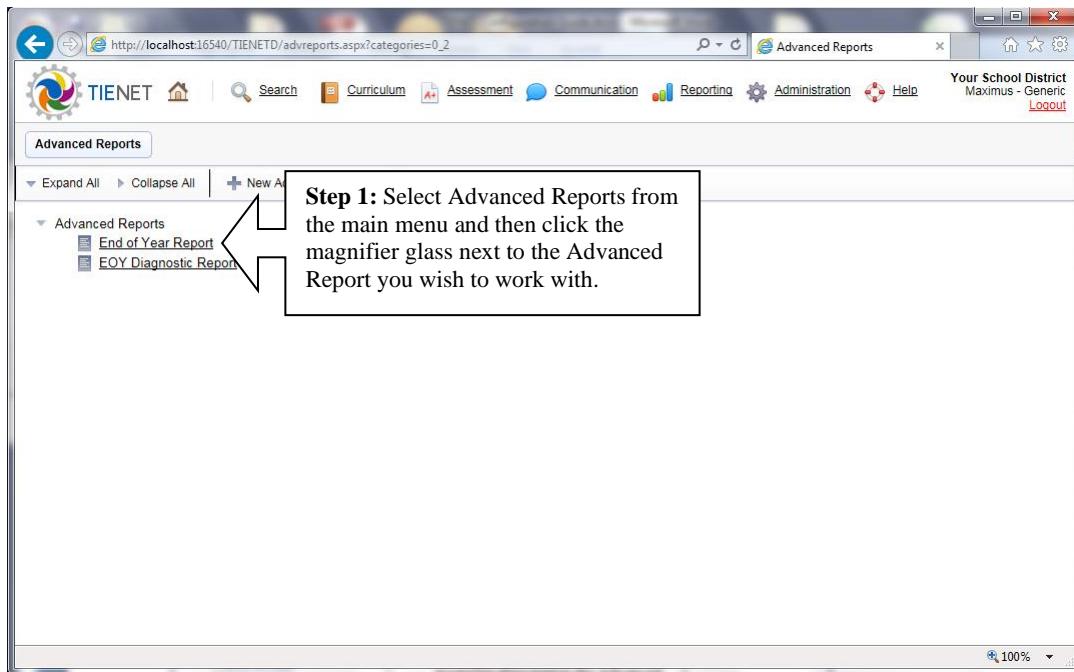
The Advanced Reporting module is designed to produce complex state and federally mandated reports using a rules-based engine that enables straightforward definition and processing of reports.

Chapter

10

Advanced Report Overview

The best way to become familiar with the Advanced Reporting module is to “explore” an existing Advanced Report, as follows:



Step 2: Advanced Reports are divided into one or more sections. Use this fly-out menu to change the section you are currently viewing.

Step 3: Each section is linked to any number of “rules” that determine which student profile or other profiles are included in the various counts and aggregates that appear in the section. Click the “Rules” tab to see the list of rules linked to the selected section. See the explanation of rules below.

Rule ID	Parent Rule	Description	Formula	Students Included
AfricanAmerican	-	The student's ethnicity is African American.	Ethnic = Black	1046 of 2117
Age14Exiting	ExitDateThisYear	The student is 14 and exited this school year.	AgeDecember1 = 14	6 of 151
Age15Exiting	ExitDateThisYear	The student is 15 and exited this school year.	AgeDecember1 = 15	16 of 151

FYI – About “Rules”: A rule is an essential building block in an Advanced Report. Rules are applied to profiles (e.g. student profiles) to determine which profiles should be included or counted in the various numeric cells (counts or aggregates) on the report. A rule is composed of the following:

- **Rule ID:** An alphanumeric identifier that identifies the rule and allows it to be uniquely referenced.

- Formula:** A logical formula that determines which profiles are included in the rule.
- Description:** The description is ideally the formula expressed in plain English.
- Parent Rule:** Each rule can optionally have a parent rule. If a rule has a parent rule, profiles can only be included in the rule if they are included in the parent rule. Parent rules enable a hierarchy of rules to be defined, resulting in better organization and more efficient report processing.

Step 4: Next click the “Layout Mode” to see how the section layouts are defined.

FYI: In the layout, look for purple-colored “directives” enclosed in squiggly brackets. Directives define the dynamic elements in the report such as counts and aggregates.

TABLE 1
Number of Public Students Referred, Initial Classifications, Reevaluations,
Declassifications & Home Instruction By Age Group And Federal Disability Category
("From July 1, "+ CHARACTER(DateYear(FirstDayOfSchool)) +" Through June 30, "+ CHARACTER(DateYear(LastDayOfSchool)))

Enter the number of students referred:
Excluding students referred only for speech-language services

(REGION_ALL)

http://localhost:16540/TIENETD/advreport.aspx?report=6&categories=0_2#

http://localhost:16540/TIENET/advreport.aspx?report=6&sec=8&mode=2&categories=0_2&prdoc=Y

or who were on home instruction for

(REGION-ALL [RULES:REPORTRESIDENT, AGE30ROLDER, AGE21ORYOUNGER, NOTNONPUBLIC, ELIGCATNOTSPEECH, ELIGCATNOTSOCIAL, ONLYUNDER6PSD])=Table1Rules	Line			
(REGION-ALL[RULES:AGEUNDER5]				
(REGION-ALL [RULES:PRESCHOOLDISABLED] =PreschoolDisabled)1		=InitClassifLn1	[RULES:REEVALTHIS =ReEvalLn1](ENDREGION)	
(REGION-ALL [RULES:ELIGCATNOTSPEECH, ELIGCATNOTPSD])=OtherCST2	Other CST	(COUNT-ALL [RULES:INITVALTHISYEAR] =InitClassifLn2)	(REGION (NotInitClasf))COUNT-ALL [RULES:REEVALTHIS =ReEvalLn2](ENDREGION)	XDATE THISYEAR =ReturnGenEdLn1
3	Speech Only			=HomeInstructionLn1 (ENDREGION)
For Ages 6-21				
(REGION-ALL[RULES:AGEOVER5] =Age6to21Region)4	Speech Only			
(REGION-ALL[RULES:AUTISTIC] =Autistic)5	AUT	(COUNT-ALL [RULES:INITVALTHISYEAR] =InitClassifLn3)	(REGIONCONTINUE (NotInitClasf))COUNT-ALL [RULES:REEVALTHISYEAR] =ReEvalLn3)(ENDREGION)	(COUNT-ALL [RULES:RETURNEDTOGENED XIDATE THISYEAR] =ReturnGenEdLn3)
(REGION-ALL[RULES:DEAFBLIND] =DeafBlind)6	DB	(COUNT-ALL [RULES:INITVALTHISYEAR] =InitClassifLn4)	(REGIONCONTINUE (NotInitClasf))COUNT-ALL [RULES:REEVALTHISYEAR] =ReEvalLn4)(ENDREGION)	(COUNT-ALL [RULES:RETURNEDTOGENED XIDATE THISYEAR] =ReturnGenEdLn4)
(REGION-ALL)		(COUNT-ALL)	(REGIONCONTINUE	(COUNT-ALL [RULES:ONHOMEINSTRUCTION, EXITEDTHISYRORINPROG] =HomeInstructionLn4) (ENDREGION)

100%

http://localhost:16540/TIENET/advreport.aspx?report=6&sec=8&mode=2&categories=0_2&prdoc=Y

Step 6: Next click “HTML Mode” and note that the layout is actually defined as HTML markup with embedded Advanced Reporting directives.

Select: Table 1

Report Mode Layout Mode HTML Mode Rules Measures

Unprocess Test Profile Setup Export... Print Section Print All

```
<table ID="Table1" width=100%><tr><td><!--Container Table-->
```

<p>Department of Education</p>

<p>Office of Special Education Program</p>

<p>SPECIAL EDUCATION END OF THE YEAR REPORT</p>

<tr> <td "="" +="" 4))<="" 4,="" <="" <td="" class="RPT_CELLCONTENTS" code:="" name:="" nowrap>("district="" substring(thisdistrict.id,="" td>="" thisdistrict.name)<="" tr=""> </td></tr>	

--

(Last Processed: 12/30/2014 Tue, 11:57 AM)

100%

Basic Advanced Reporting Directives

To become familiar with the various types of directives described below, you can look to see how they are used in the existing Advanced Reports:

{=globalsformula} Specifies a computed value based on global fields. Examples:

- `{=DistrictName}`
- `{=CountyName}`
- `{=SuperintendentName}`

{COUNT-logic[Rules: ruleids]=cellid} Specifies a student count or other count. The “logic” specifier is optional and, if specified, is one of ALL, ANY, NOTALL or NOTANY. If omitted, it is assumed to be ALL. The “ruleids” is a comma delimited list of rule identifiers. Examples are as follows:

- **{COUNT[Rules: GRADE01, GRADE02]}** Count of all student profiles included in all the specified rules, namely GRADE01, GRADE02.
- **{COUNT-ALL[Rules: GRADE01, GRADE02]}** Same as above, but in this case, the ALL logic is explicitly specified.
- **{COUNT-ANY[Rules: GRADE01, GRADE02]}** Count of all student profiles included in ANY of the specified rules.
- **{COUNT-NOTANY[Rules: GRADE01, GRADE02]}** Count of all student profiles NOT included in ANY of the specified rules.
- **{COUNT-NOTALL[Rules: GRADE01, GRADE02]}** Count of all student profiles NOT included in ALL of the specified rules.
- **{COUNT-ALL[Rules: GRADE01, GRADE02]=Grade12Cell}** This version of the directive gives the numeric cell count a Cell ID of Grade12 so that it can be referenced from other directives as described next.

{COUNT-logic[Cells: cellids]=cellid} Specifies a student count or other count based strictly on inclusion in other cells which are referenced by Cell ID.

- **{COUNT-ANY[Cells: GRADE12Cell, GRADE34Cell]}** Count of all student profiles included in any of the other cells identified by cell ID, namely GRADE12Cell, GRADE34Cell.

{REGION-logicfunction[Rules: ruleids]=regionid}content{ENDREGION} or **{REGION-logicfunction[Cells: cellids]=regionid}content{ENDREGION}** Specifies that any other numeric cells included in the content of this region (between REGION and ENDREGION) should be filtered to the student profiles or other profiles included in the region’s rule logic. The syntax of the REGION directive is the same as for COUNT directive. You can optionally specify a Region ID which can be referenced in REGIONCONTINUE directive, described next.

- **{REGION-ALL[Rules: SpeechImpaired, Grade01]}content{ENDREGION}** Any cells between the REGION and ENDREGION directives are filtered to those profiles included in the region. Regions can be nested.

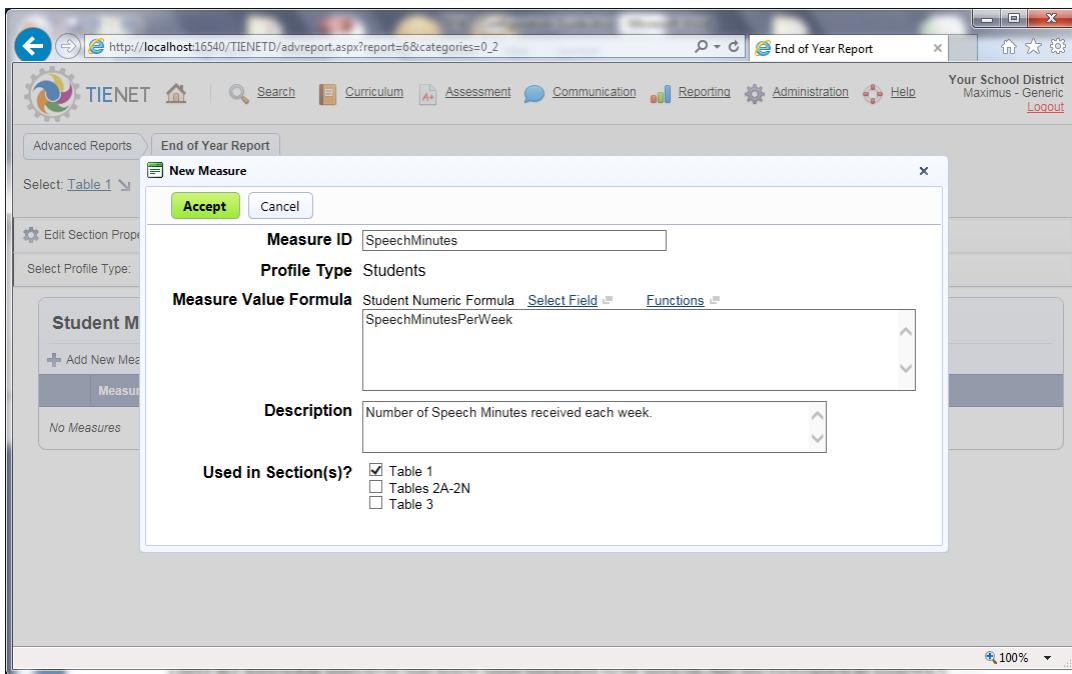
{REGIONCONTINUE(regionid)}content{ENDREGION} This set of directives continues a previous region identified by its region ID. In effect, it allows you to create a discontinuous region.

Numeric cells in an Advanced Report are typically organized into tabular grids. The optimal way to define regions in the HTML formatting for a tabular grid is given below. Note that within the `<table>` `</table>` tags, the Advanced Reporting directives should only appear inside `<th>` or `<td>` tags, otherwise strange display problems can result. Also note that a region is defined for the table, each row, and each column. Each column region is defined as a discontinuous region using REGIONCONTINUE. Each column region is defined in the corresponding column header, but is then re-continued in each row.

```
{REGION[Rules:ruleids]=TABLEA}
<table>
<tr>
    <th>{REGION[Rules:ruleids]=TABLEAC1}Column 1 Header{ENDREGION}</th>
    <th>{REGION[Rules:ruleids]=TABLEAC2}Column 2 Header {ENDREGION}</th>
    <th>{REGION[Rules:ruleids]=TABLEAC3}Column 3 Header {ENDREGION}</th>
</tr>
<tr>
    <td>{REGION[Rules:ruleids]=TABLEAR1}{REGIONCONTINUE(TABLEAC1)r1c1{ENDREGION}}</td>
    <td>{REGIONCONTINUE(TABLEAC2)r1c2{ENDREGION}}</td>
    <td>{REGIONCONTINUE(TABLEAC3)r1c3{ENDREGION}}{ENDREGION}</td>
</tr>
<tr>
    <td>{REGION[Rules:ruleids]=TABLEAR2}{REGIONCONTINUE(TABLEAC1)r2c1{ENDREGION}}</td>
    <td>{REGIONCONTINUE(TABLEAC2)r2c2{ENDREGION}}</td>
    <td>{REGIONCONTINUE(TABLEAC3)r2c3{ENDREGION}}{ENDREGION}</td>
</tr>
</table>
{ENDREGION}
```

Advanced Report Measures

Rules in combination with the directives described above allow you to produce any imaginable student counts or other counts on an Advanced Report. However, there may be cases where you need to incorporate other types of aggregates such as averages, sums, minimums and maximums. To include these types of aggregates on a report, you need one or more measures. A measure is a numeric value defined by a numeric formula and identified by a Measure ID. The screen below shows the screen used to define a measure.



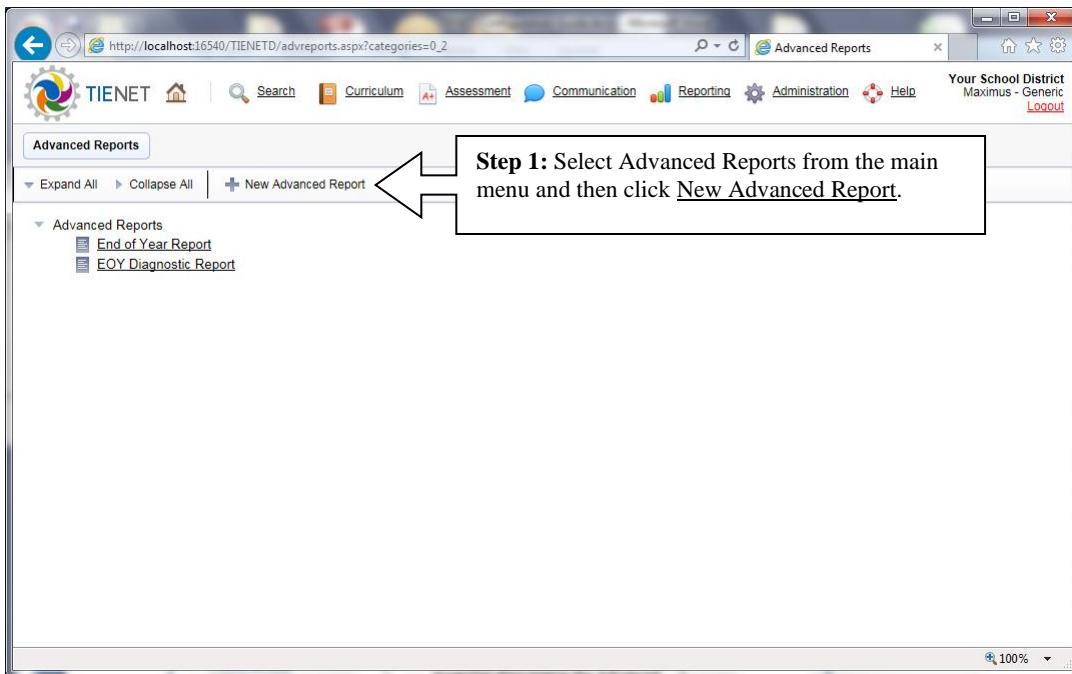
There are additional directives that allow these measures to be used on Advanced Reports as follows:

- **{COUNT(measureid)-logic[Rules: ruleids]=cellid}** or
{COUNT(measureid)-logic[Cells: cellids]=cellid}
- **{SUM(measureid)-logic[Rules: ruleids]=cellid}** or
{SUM(measureid)-logic[Cells: cellids]=cellid}
- **{AVG(measureid)-logic[Rules: ruleids]=cellid}** or
{AVG(measureid)-logic[Cells: cellids]=cellid}
- **{MIN(measureid)-logic[Rules: ruleids]=cellid}** or
{MIN(measureid)-logic[Cells: cellids]=cellid}
- **{MAX(measureid)-logic[Rules: ruleids]=cellid}** or
{MAX(measureid)-logic[Cells: cellids]=cellid}

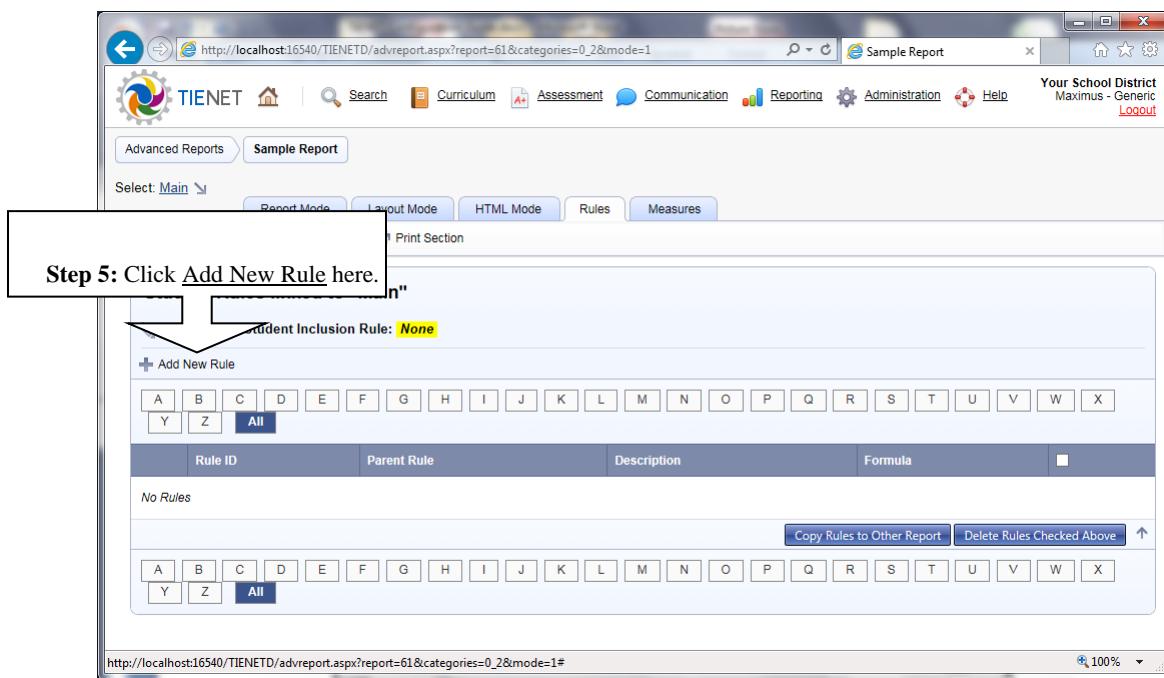
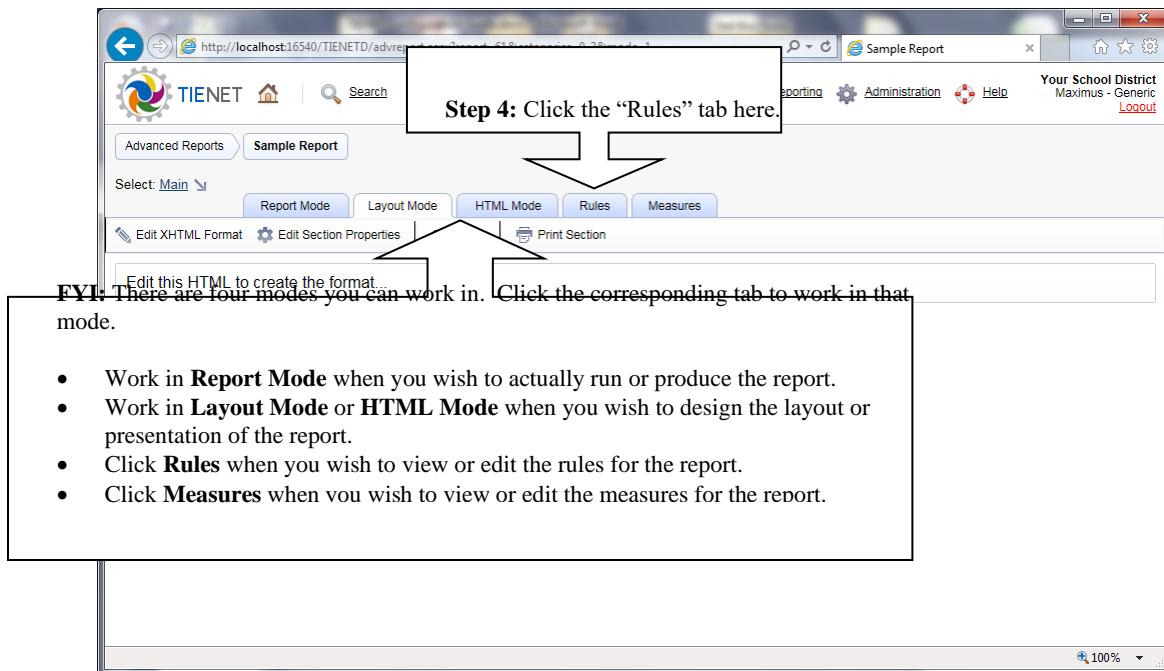
The syntax of these directives is very similar to the syntax for the basic COUNT directive. The new variation of the COUNT directive shown here (with the *measureid* parameter) is like the basic COUNT directive but further filters the count to profiles that have a non-EMPTY measure value. The other directives respectively produce the sum, average, minimum or maximum of the measure value across the included profiles.

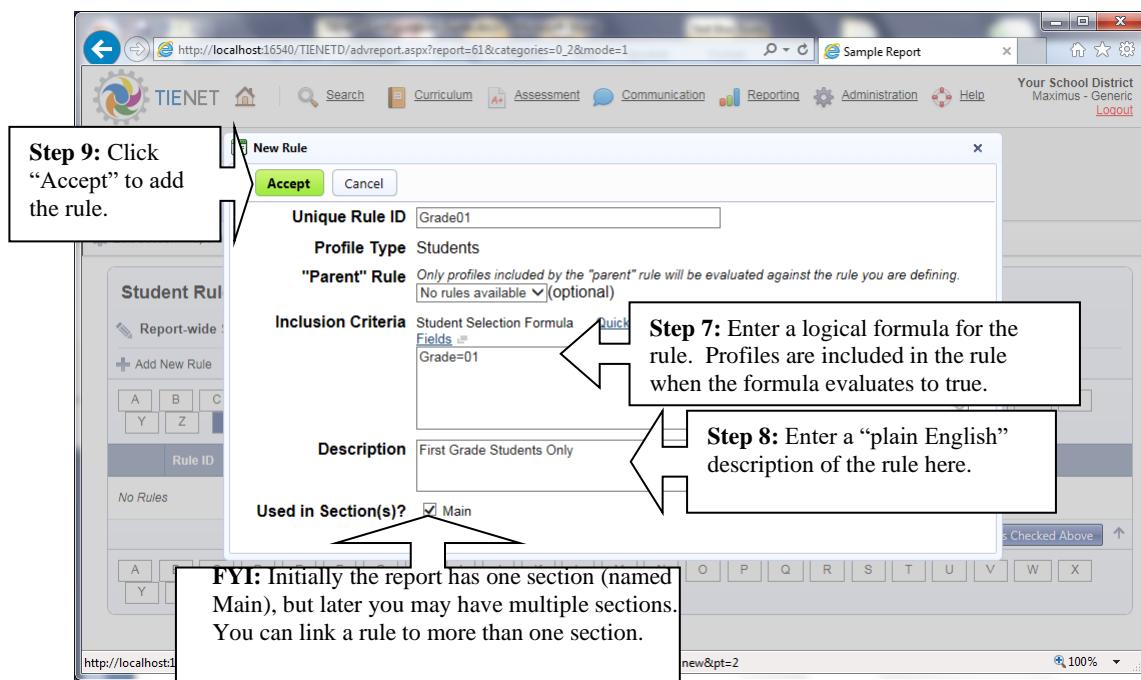
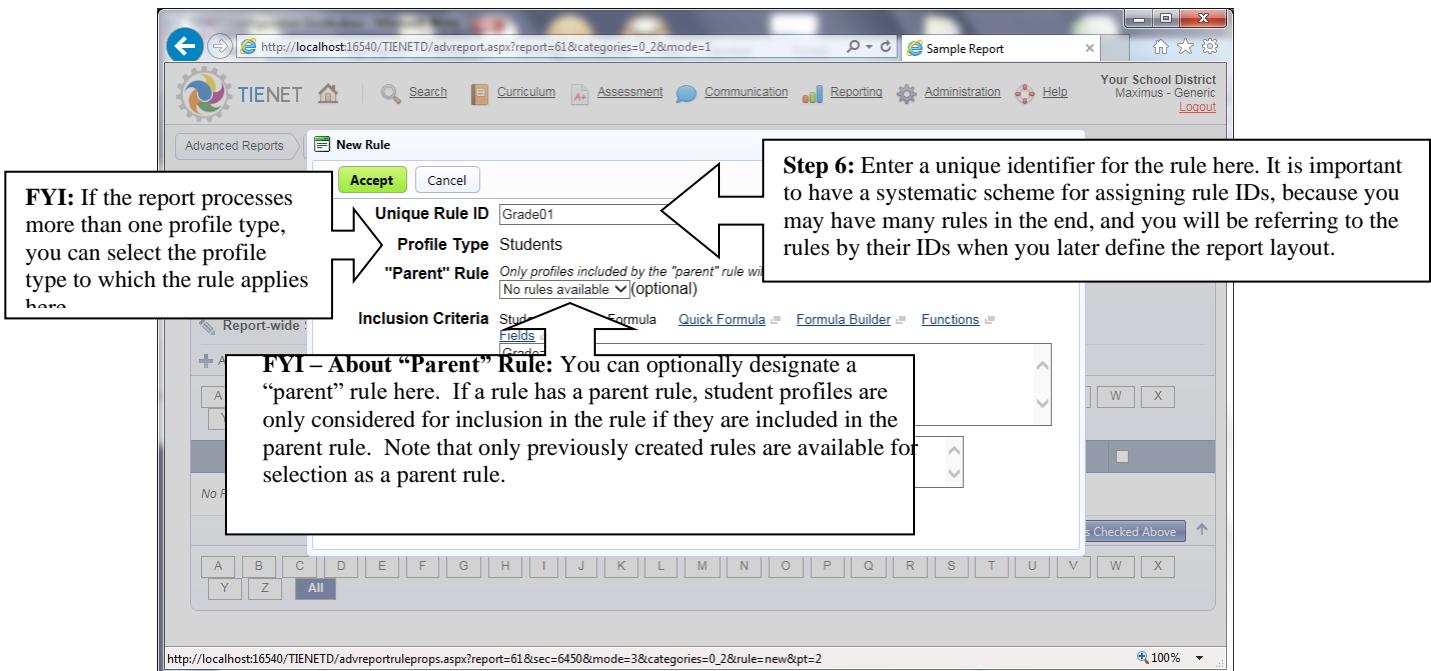
Creating an Advanced Report

This section will lead you through creating a very simple advanced report. In the following sections, more advanced techniques will be introduced.



The screenshot shows a web browser window for 'TIENET' at the URL http://localhost:16540/TIENETD/advreportprops.aspx?report=new&categories=0_2. The page title is 'Advanced Report Properties'. The top navigation bar is identical to the previous screen. The main form is titled 'New Advanced Report' and contains fields for 'Report Name' (Sample Report), 'Unique Report ID' (MYSAMPLE), 'Report Description' (An example Report), and 'Report Category' (none). Below these, a section titled 'Profile Type(s) for Reporting' lists various options with checkboxes. A callout box with an arrow points to the 'Report Name' field with the text: 'Step 2: Enter a descriptive name for the report, a unique 10-character alphanumeric identifier and a description. You can also place the report into an existing or new category.' Another callout box with an arrow points to the 'Profile Type(s) for Reporting' section with the text: 'Step 3: Check the types of profiles that will be directly counted, aggregated and processed on the report. Typical advanced reports process only students.' At the bottom right of the form, there's a note: 'At the bottom, click "Accept" to continue.'





Step 10: The rule is now added. You can click the corresponding edit (pencil) icon to edit it, or the delete (trashcan) icon to delete it. Repeat the previous steps to add any additional rules you know you will need. If you have a series of similar rules, you can click the copy icon (to make a modified copy of the existing rule.

Rule ID	Parent Rule	Description	Formula	checkbox
Grade01	-	First Grade Students Only	Grade=01	<input checked="" type="checkbox"/>

Copy Rules to Other Report | Delete Rules Checked Above

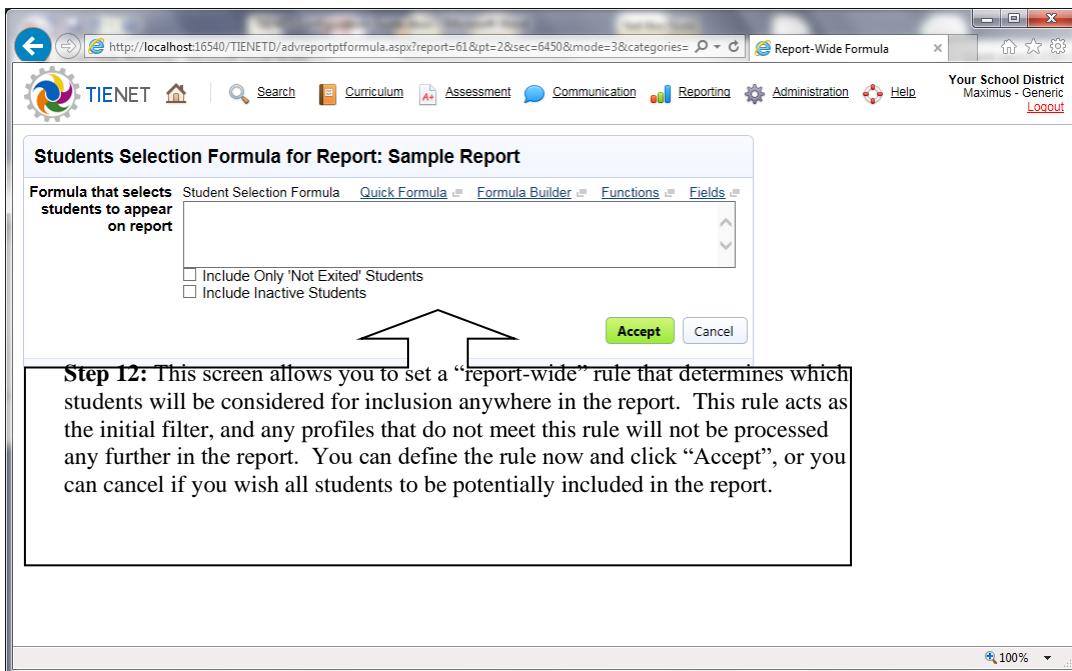
Step 11: Click the edit (pencil) icon here to see how to enter the “report-wide” inclusion rule (in this case, for students).

Rule ID	Parent Rule	Description	Formula	checkbox
Grade01	-	First Grade Students Only	Grade=01	<input type="checkbox"/>
Grade02	-	Second Grade Students Only	Grade=02	<input type="checkbox"/>
Grade03	-	Third Grade Students Only	Grade=03	<input type="checkbox"/>
Grade04	-	Fourth Grade Students Only	Grade=04	<input checked="" type="checkbox"/>

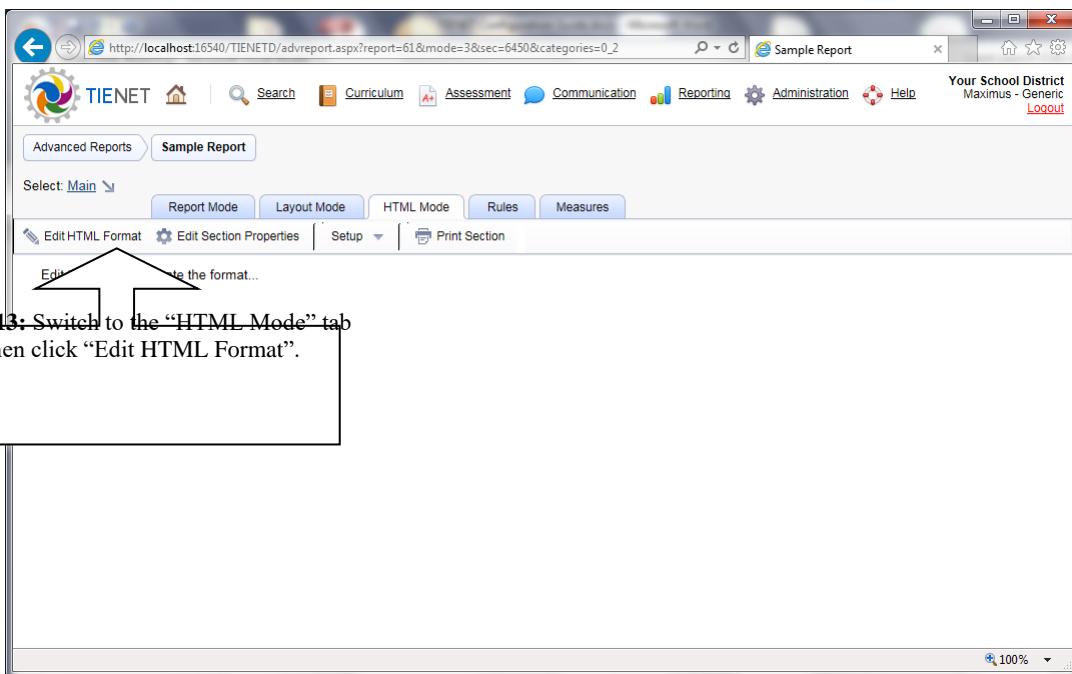
Report-wide Student Inclusion Rule: **None**

Add New Rule

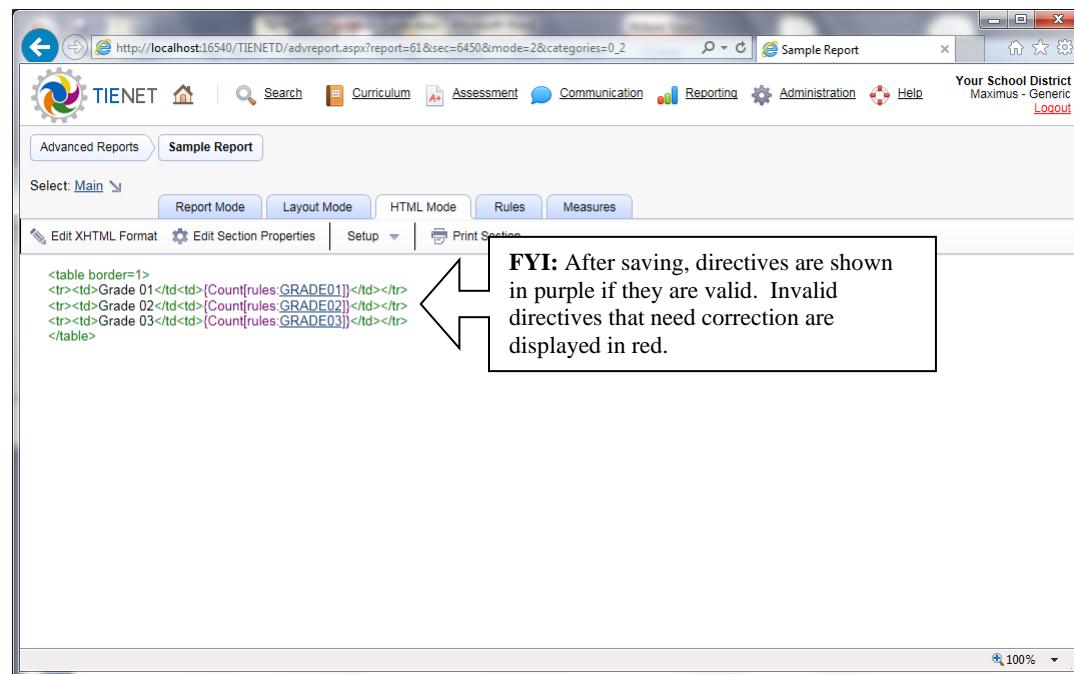
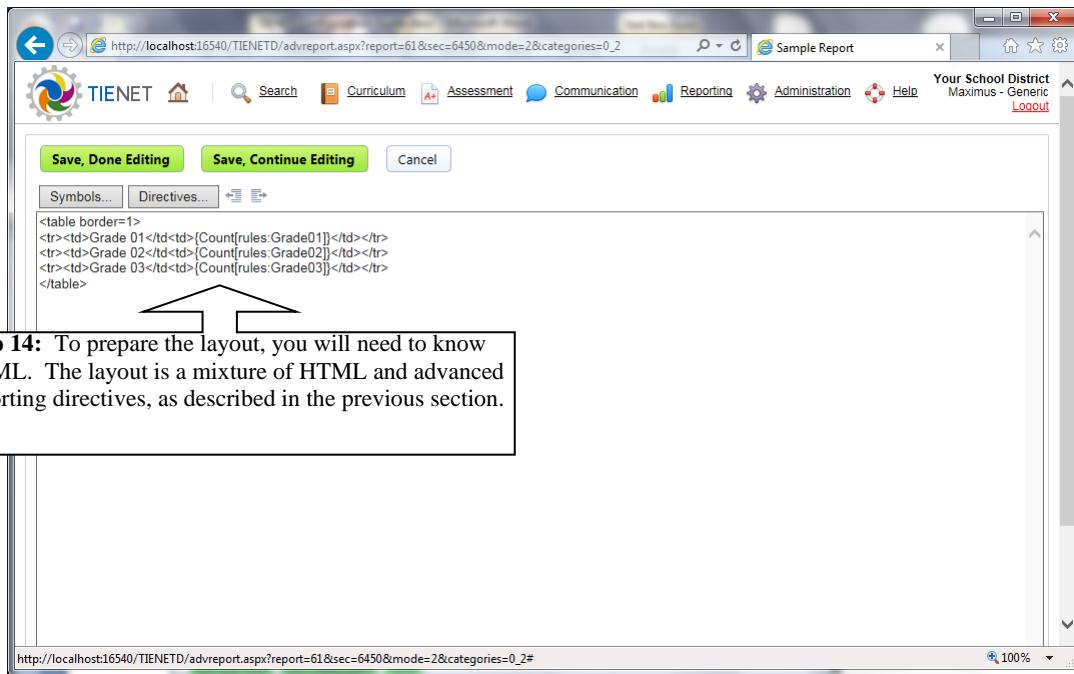
Rules linked to "Main"

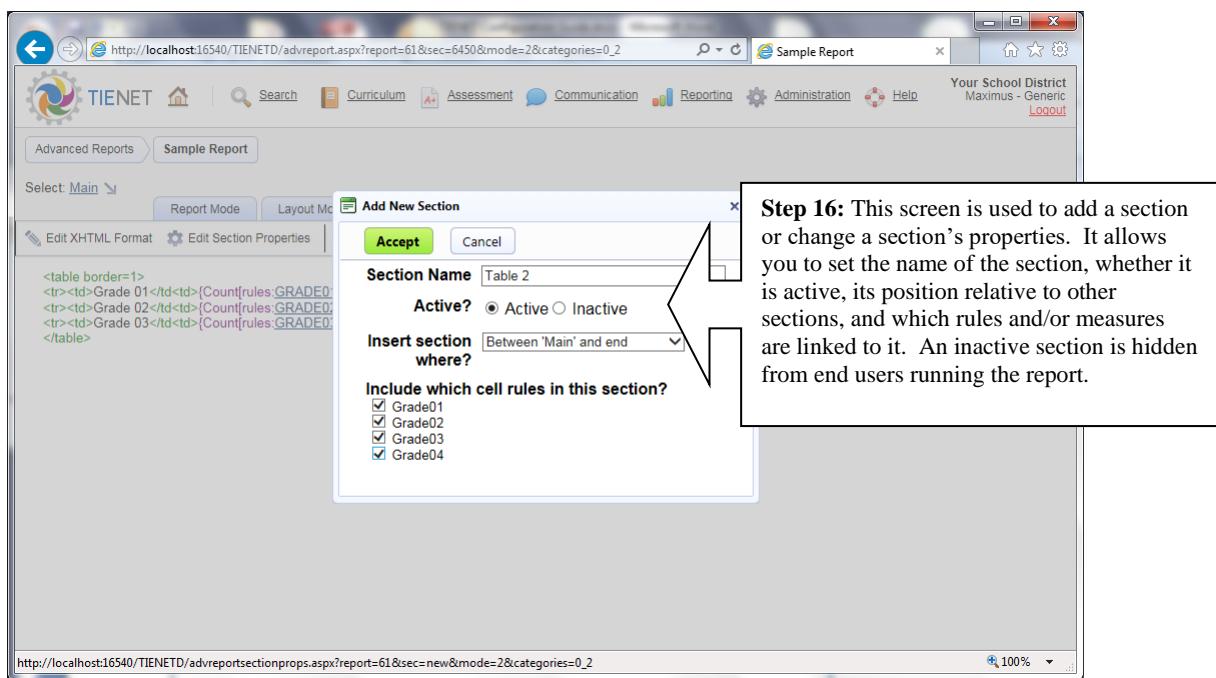
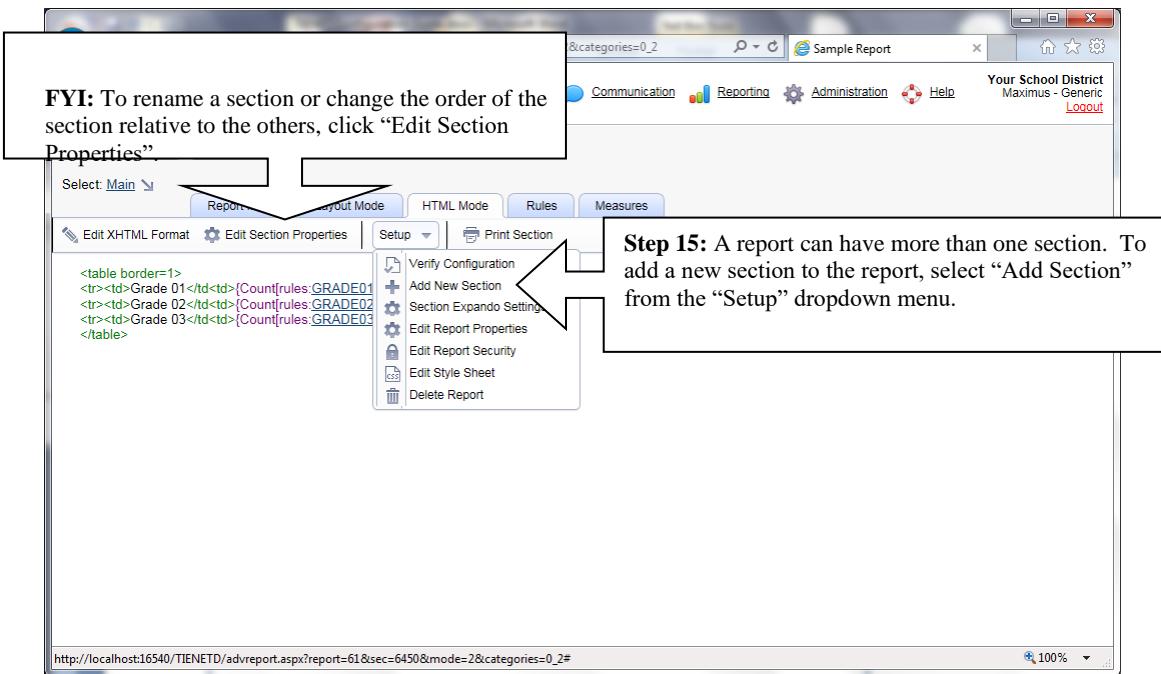


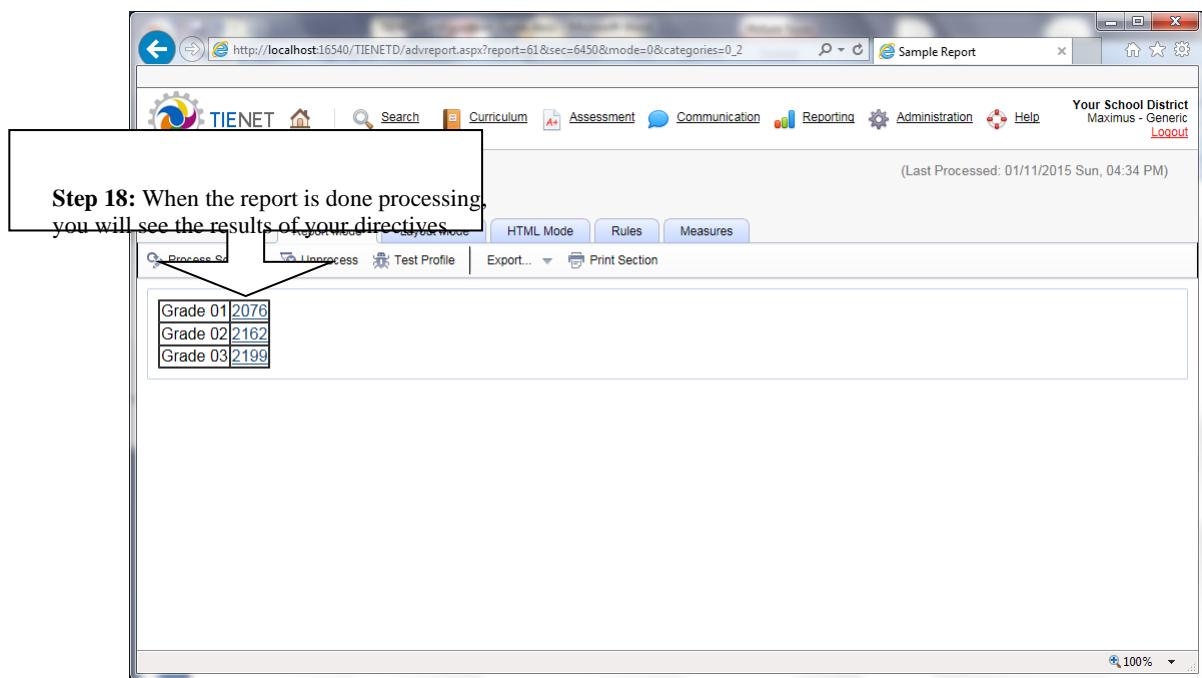
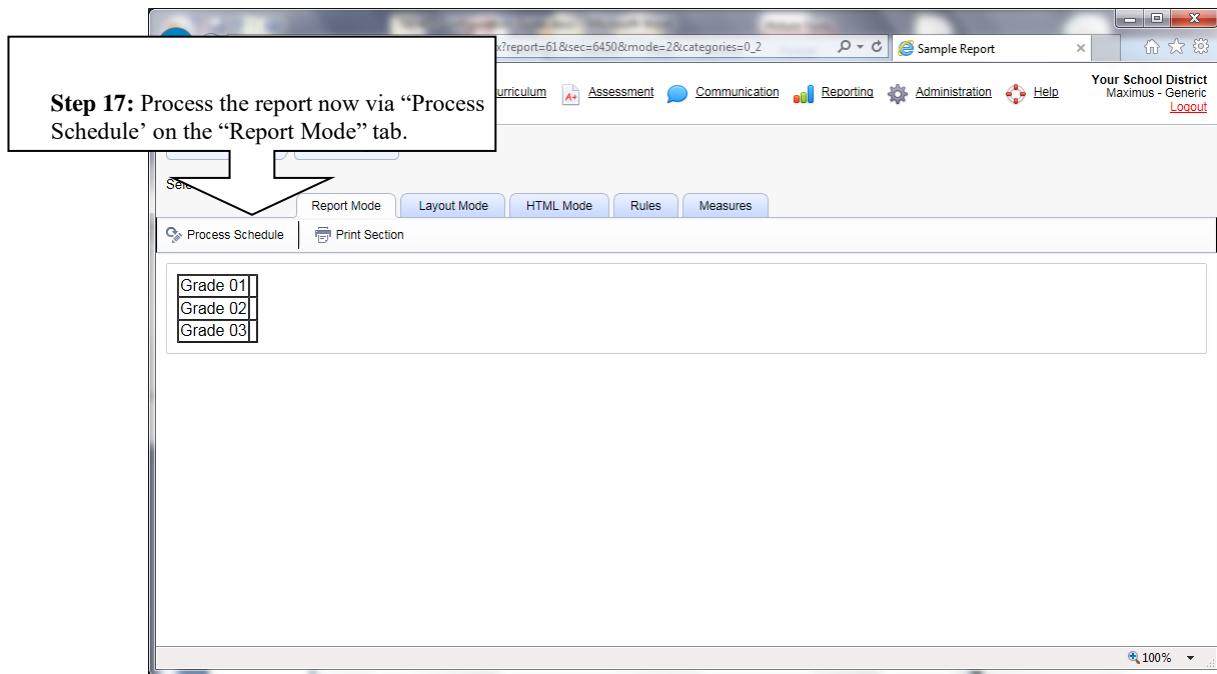
Step 12: This screen allows you to set a “report-wide” rule that determines which students will be considered for inclusion anywhere in the report. This rule acts as the initial filter, and any profiles that do not meet this rule will not be processed any further in the report. You can define the rule now and click “Accept”, or you can cancel if you wish all students to be potentially included in the report.



Step 13: Switch to the “HTML Mode” tab and then click “Edit HTML Format”.







Command (#) Directives

There are some additional directives that may be useful:

{#IF formula}content{#ENDIF} If you wish to include content conditional upon the result of a logical formula, enclose the content between **{#IF formula}** and **{#ENDIF}**. Note that the formula may only reference global fields.

{#EVAL expression} or {#EVAL expression: #.##} This directive is useful for transforming the numeric output of named numeric cells. Simply write an algebraic expression referencing the named cells using operators +, -, *, /, % (modulus) with standard parenthesis as needed. When using division, caution should be used to avoid “divide by zero” errors. Additionally it should be understood that some aggregate values may result in NULL (i.e. SUM of a measure where no actual numeric values are summed). The functions listed below are supported and can assist with these cases. Note that the user will be unable to drill down into an #EVAL expression and so this should only be used when the result cannot be produced by the standard techniques described earlier. The **{#EVAL expression: #.##}** variation allows you to specify the number of decimal places, especially in situations where division is used. For example **{#EVAL expression: #.###}** specifies three decimal places for display purposes.

IIF(logicalexpression, alternative1, alternative2} returns alternative1 if the logicalexpression is true, otherwise it returns alternative2. This can be used to avoid “divide by zero” errors, e.g. IIF(divisor <> 0, numerator / divisor, null). It can also be used to transform numeric cells into text output, i.e. IIF(*cellid* > 1000, ‘HIGH’, ‘LOW’)

ISNULL(expression, alternateifnull) returns expression if it is not null, otherwise it returns alternateifnull.

Adding Charts to Advanced Reports

You can add colorful pie, bar and column charts to your advanced reports provided that the numbers that comprise the numeric series to be charted are already numeric cells on the same section of the advanced report. This is accomplished using chart directives as follow:

```
{$PIE(cellid:"label",...)[title="MyChart",TitleFontSize=14,Height=400,Width=400,DataFontSize=12,Thre
eDee=true]}
```

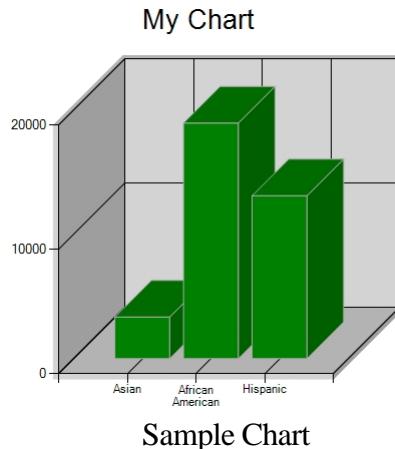
```
{$BAR(cellid:"label",...)[title="MyChart",TitleFontSize=14,Height=400,Width=400,MeasureName="Pop
ulation",DataFontSize=12,ThreeDee=true,Cylinder=true]}
```

```
{$COLUMN(cellid:"label",...)[title="MyChart",TitleFontSize=14,Height=400,Width=400,MeasureName=
"Population",DataFontSize=12,ThreeDee=true,Cylinder=true]}
```

As an example, if you want to produce a pie chart using two aggregates in your advanced report, make sure that each aggregate cell to be charted has an explicit cell ID (described in previous sections). If those cell IDs were Resident and Received, your directive might look like:

```
{$PIE(Resident:"Resident Students", Received:"Students Received From Other Districts") [title="Resident Versus Received", TitleFontSize=14, Height=400, Width=400, DataFontSize=12, ThreeDee=true]}
```

The best way to determine how the additional named properties work is to experiment with them.



Sample Chart

As an example, if you want to produce a pie chart using two aggregates in your advanced report, make sure that each aggregate cell to be charted has an explicit cell ID (described in previous sections). If those cell IDs were Resident and Received, your directive might look like:

```
{$PIE(Resident:"Resident Students", Received:"Students Received From Other Districts") [title="Resident Versus Received", TitleFontSize=14, Height=400, Width=400, DataFontSize=12, ThreeDee=true]}
```

The best way to determine how the additional named properties work is to experiment with them.

The PIE directive also supports these additional attributes that can be used to improve the presentation of the resulting pie chart:

- LabelPieOutside =true (moves slice labeling outside of the pie, which may make the chart less busy)
- AppendValuesToLabels=true (results in the numeric values being appended to the pie-slice labels, since otherwise they do not appear)
- LabelPieWithValues=true (directly labels slices with the numeric values, but this should be used in conjunction with ShowLegend=true since otherwise the text labels will not be visible)

“Expando” Report Sections

If a report has rows of data and aggregates where each row corresponds to a school, the previous techniques only allow you to configure the report if you know at the time of configuring the report exactly what each school will be, and in this case, you would have to “hard code” the schools into the report requiring you to go back and change the configuration any time the schools on the report need to change. Many typical

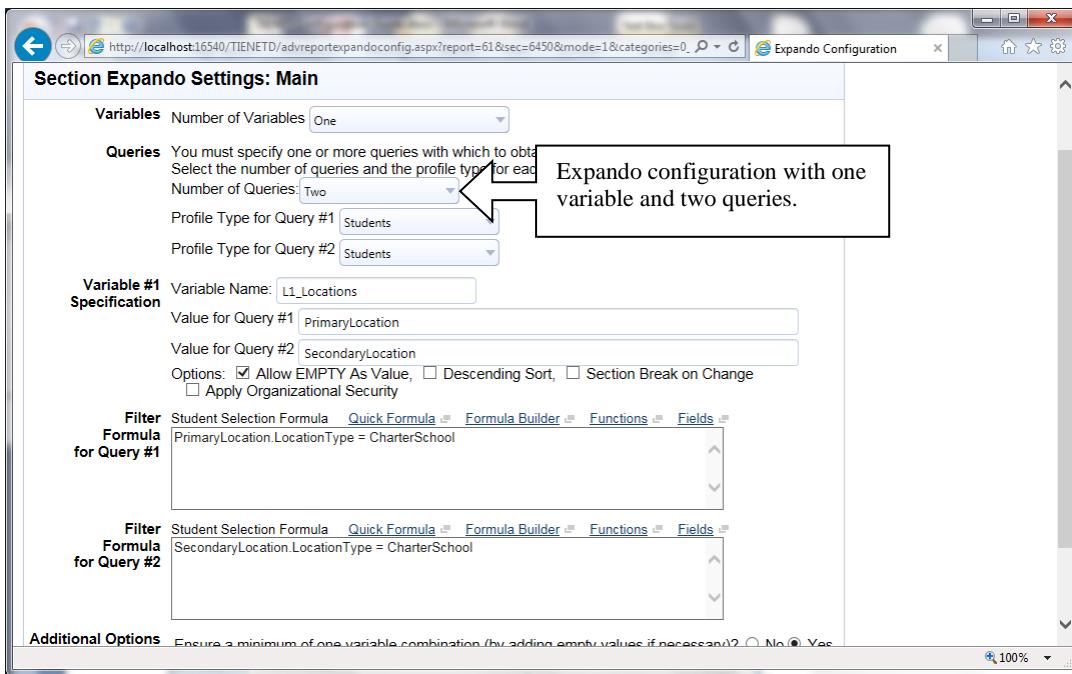
reports have qualities that are similar to this. To implement reports like this without “hard coding”, you configure an “expando” section.

To set up an “expando” section, create or go to a section that you would like to set up as “expando”. Then select “Section Exando Settings” from the “More dropdown”. The expando settings screen allows you to define one or more “expando variables” and how they get populated. Note that an “expando section” is simply a section that has one or more expando variables. When the advanced report is generated, PowerSchool Special Programs will take the expando section and repeat it for each unique combination of values that have been populated into the expando variables. As an example, if you would like to create an expando section that gets repeated for each charter school attended by at least one student in the database, then (as in the screen below), you could set up one expando variable named “L1_Location” that gets populated by querying the value of the PrimaryLocation field from the students profile type where the PrimaryLocation is a charter school (see below). You can name the expando variable whatever you like, but as a suggested convention, you can prefix it with L1_ and where there is more than one, L2_, L3_, etc. Prefixing expando variables this way makes them easy to identify in formulas later.

The screenshot shows a web-based configuration interface for 'Expando Configuration'. The title bar reads 'Expando Configuration' and 'Your School District Maximus - Generic Logout'. The main content area is titled 'Section Exando Settings: Main' with a sub-section 'Variables' showing 'Number of Variables One'. Below this, under 'Queries', it says 'You must specify one or more queries with which to obtain the values for the variables.' A dropdown 'Number of Queries' is set to 'One', and the 'Profile Type for Query #1' is 'Students'. Under 'Variable #1 Specification', the 'Variable Name' is 'L1_Locations', and the 'Value for Query #1' is 'PrimaryLocation'. Under 'Filter Formula for Query #1', the formula is 'PrimaryLocation.LocationType = CharterSchool'. At the bottom, 'Additional Options' includes a checkbox for 'Ensure a minimum of one variable combination (by adding empty values if necessary)?' with 'Yes' selected. There are 'Accept' and 'Cancel' buttons at the bottom right.

The above example with charter schools has only one expando variable. However, consider a report that will have a row of aggregate data for each unique combination of grade level, ethnic and gender found in the student population. This example would warrant three expando variables for grade level, ethnic and gender respectively.

In some cases, more than one query is needed to fully and accurately populate the expando variables. In the example below, two queries populate the L1_Location expando variable with all charter schools that are either the primary location or the secondary location. PowerSchool Special Programs will populate the expando variable with all the unique values found in both queries.



By default, PowerSchool Special Programs will repeat the entire section for each unique set of values populated into the expando variables. As an alternative, you can configure the section such that a portion of the section repeats, so that with more unique combinations, the one section expands instead of repeating in its entirety. To configure this, just enclose the repeating portion of the HTML format with {#EXPANDO} and {#ENDEXPANDO}.

You can use rules and measures anywhere within an expando section. However, to be useful, the expando section will need to have rules and/or measures that reference the populated values of the expando variables. Standard rules and measures cannot reference expando variables. However, once you have an expando section, you will see links that allow you to create “expando rules” and “expando measures”. The process is similar, except that only “expando rules/measures” can reference expando variables and “expando rules/measures” can only be used in an expando section.

When setting up expando variables in the expando configuration screen, there are some additional options that require explanation:

- **Allow EMPTY as Value:** If this option is set for an expando variable, then PowerSchool Special Programs allows the expando variable to be EMPTY. In this case, the values with which the expando variable is populated could include the EMPTY value if EMPTY results from the query. If this option is turned off, the EMPTY value is disallowed and filtered out.
- **Descending Sort:** When the expando section is repeated for each unique value(s) in the expando variable(s), the sections are repeated according to the natural ordering for that variable (i.e. alphabetical order, keyword order, numerical order, etc.) Enabling this option reverses the natural order.

- Section Break on Change: This option is only relevant if the {#EXPANDO}{#ENDEXPANDO} directives are used to repeat only a portion of a section. If enabled, PowerSchool Special Programs will generate an entirely new section wherever there is a change in any of the expando variable values.
- Apply Organizational Security: The option can only be enabled if the corresponding expando variable is a profile selection for an organizational location, and is actually only relevant if the repeating section approach is used. If enabled, PowerSchool Special Programs checks the user's security and only allows the user to see instances of the section corresponding to organizational locations that the user has profile view access to. To determine profile view access, PowerSchool Special Programs checks profile view privileges for the first profile type the report is aggregating data for (typically Students, or if not students, then typically Staff).
- Ensure a minimum of one variable combination (by adding empty values if necessary)? Yes/No: There is the possibility that zero values will be queried for the expando variables, and that would result in zero repetitions of the expando section causing it not to appear at all in the report output. If this option is turned on, PowerSchool Special Programs will check if zero values have been queried, and if so, will populate the expando variables with an empty value ensuring that at least one repetition is made.

Setting Drilldown Columns

By default, the system infers default columns when drilling down into each report cell based on cell rules and other heuristics, and an end user can customize the inferred default columns. However, one has the option of configuring a specific set of columns in a particular order, which then replaces the inferred columns as the default columns. To do this, select a configuration task, switch to the Layout tab, and select Setup > Set Drilldown Columns. The end user can still customize the default columns whether they are inferred or preset as part of the report definition. Since the preset columns are part of the report definition, they are synced by the CMT.

Optimizing Advanced Report Performance

The performance (and time to process) of an advanced report can often be improved by the following practices.

1. Be sure to utilize the report-wide filter rules and make them as restrictive as possible. This often does more than anything else to reduce the amount of processing.
2. If two or more rules are always applied to the same report cells or areas, you can often reduce processing by merging those rules into a single rule by combining the formulas with 'OR' logic.

Configuration Management Tool

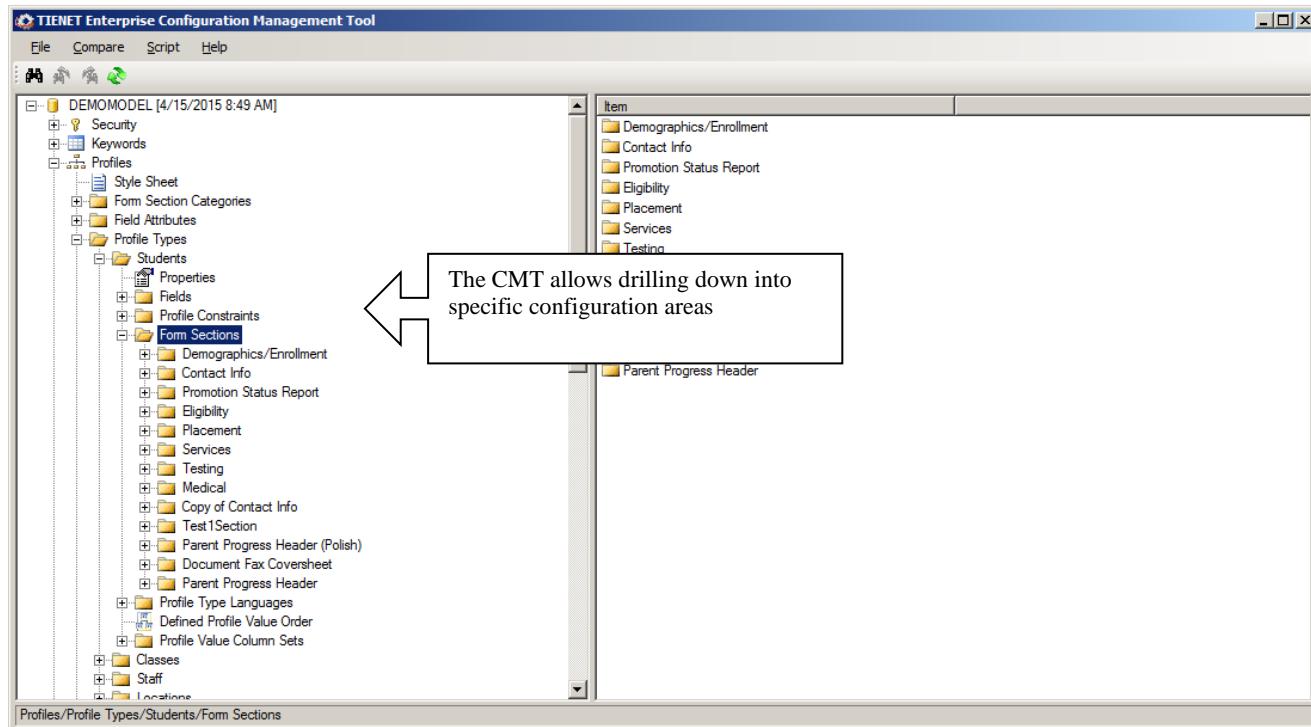
The Configuration Management Tool (CMT) is a server application used to extract, store, compare, and synchronize configurations of PowerSchool Special Programs databases.

Chapter

11

CMT Overview

When you open the configuration for a database in the CMT, the information is separated into two panes. The left pane allows drilling down into specific configuration areas while the right pane displays contained configuration elements.



The loaded configuration can be saved to a file on disk for later reference. These files can be used to track configuration changes to a database over time.

The configuration is organized into various repositories:

- **Security:** includes security settings (Admin, Consultant, Staff, Parents & Students, Audit Log) as well as Staff security groups with their granted and denied privileges
- **Keywords:** includes all keyword table configuration and data

- **Profiles:** includes overall profile configuration (global stylesheets, form section categories, field attributes), profile types configuration (fields, form sections, constraints, security, translation), update scripts, placement definitions, and custom UDF configuration
- **Documents:** includes overall document configuration (global stylesheets, document template categories), document template configuration (fields, sections, child templates, actions, security, translation), and custom event fields.
- **Reports:** includes all advanced reports, any standard (ad-hoc) reports that have been placed under configuration management, and security associated with the reports
- **Integration:** includes profile and document import layouts, and direct integrations (such as Active Directory)
- **Messaging:** includes messaging settings for the database
- **Help:** includes help settings and external help resources

One of the main uses of the CMT is to create and/or change configuration in a base database (such as a development or state model database), and when complete use the CMT to propagate those changes to one or more databases.

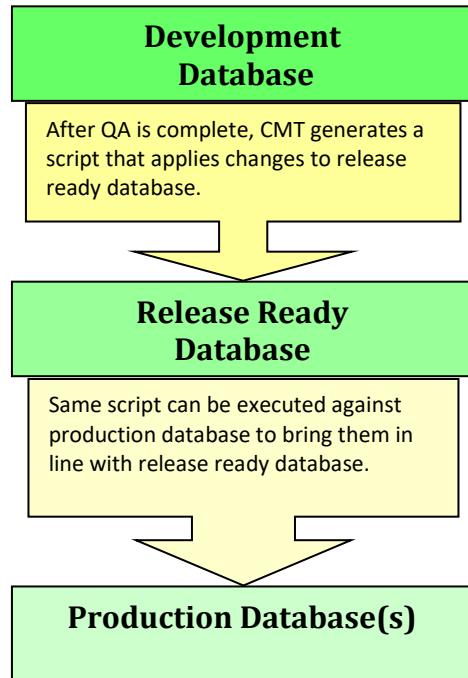


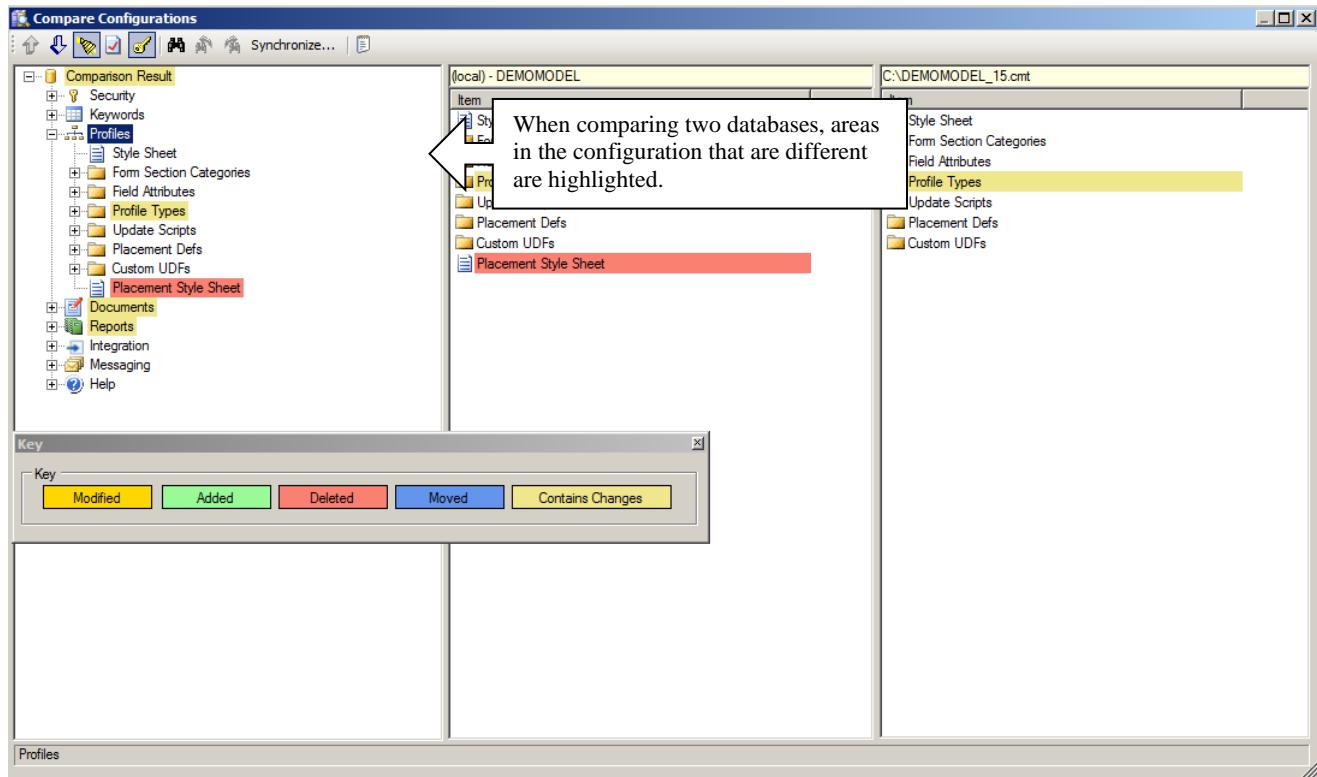
Figure 10.1 – Developing and Propagating Configurations

Model Versions

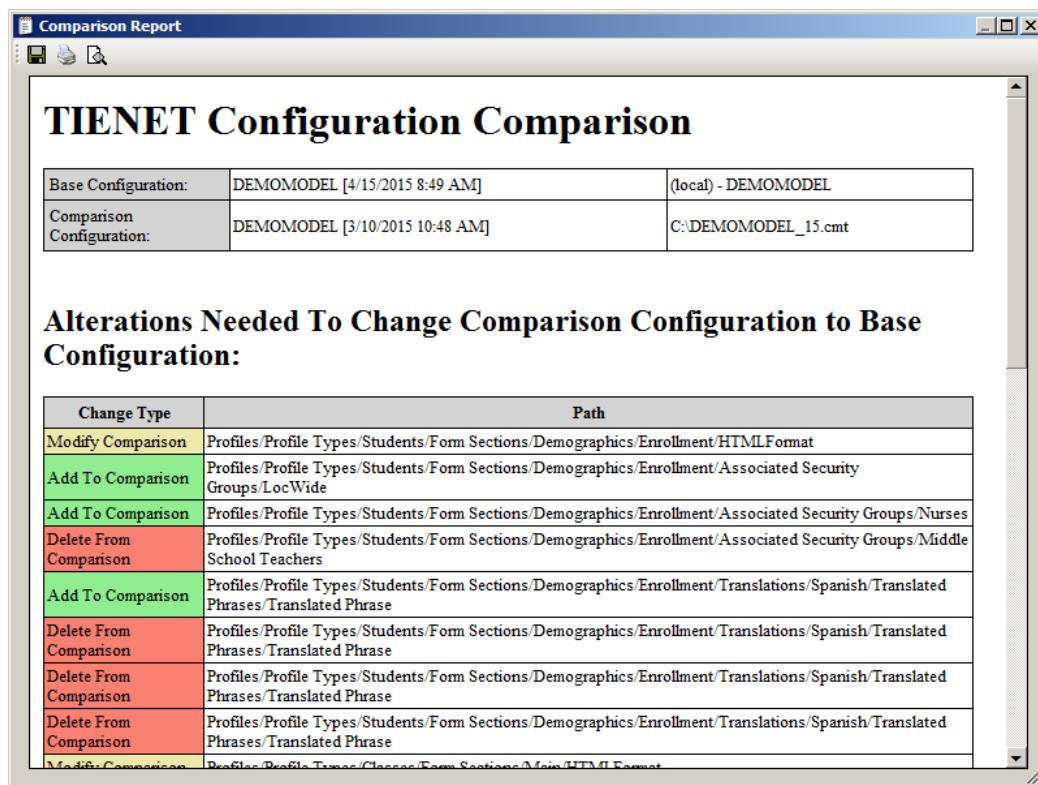
As of Version 17.1, model versions are explicitly supported in the configuration. To establish the model version, go to Configuration > Profile Types > More > Set Model Version (after selecting a configuration task). Model versions have the format <ALPHA>.<year:YYYY>.<month:1-12>.<iteration>, for example, NY.2017.11.0. The CMT recognizes the model version, and when generating a script that represents a model version upgrade (e.g. NY.2017.11.0 to NY.2017.11.1), the generated script will verify that the target database is on the old model version and then set the new model version on the target database after making all changes.

Comparing Databases

The first step to copying configuration changes from one database to another is comparing their configuration. This is accomplished by loading the “base” database into the CMT and then comparing it to a “comparison” database. .

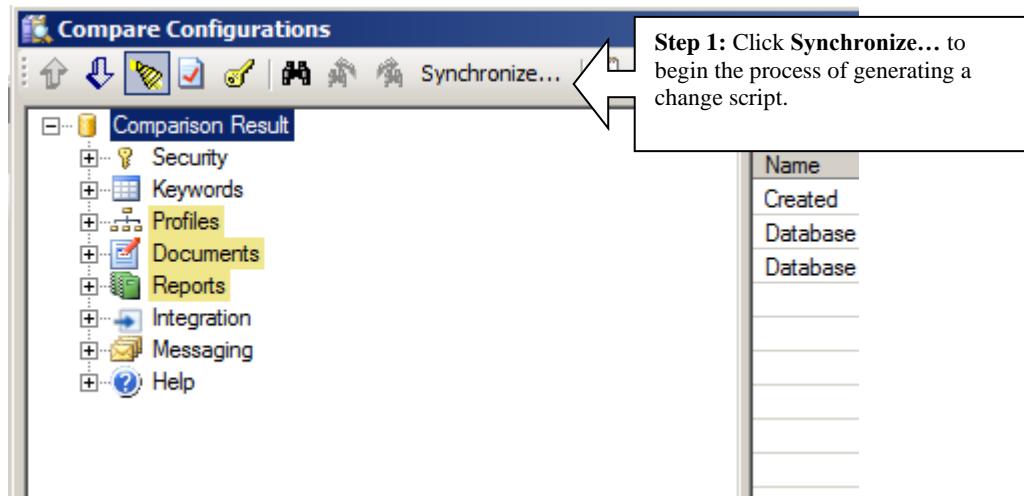


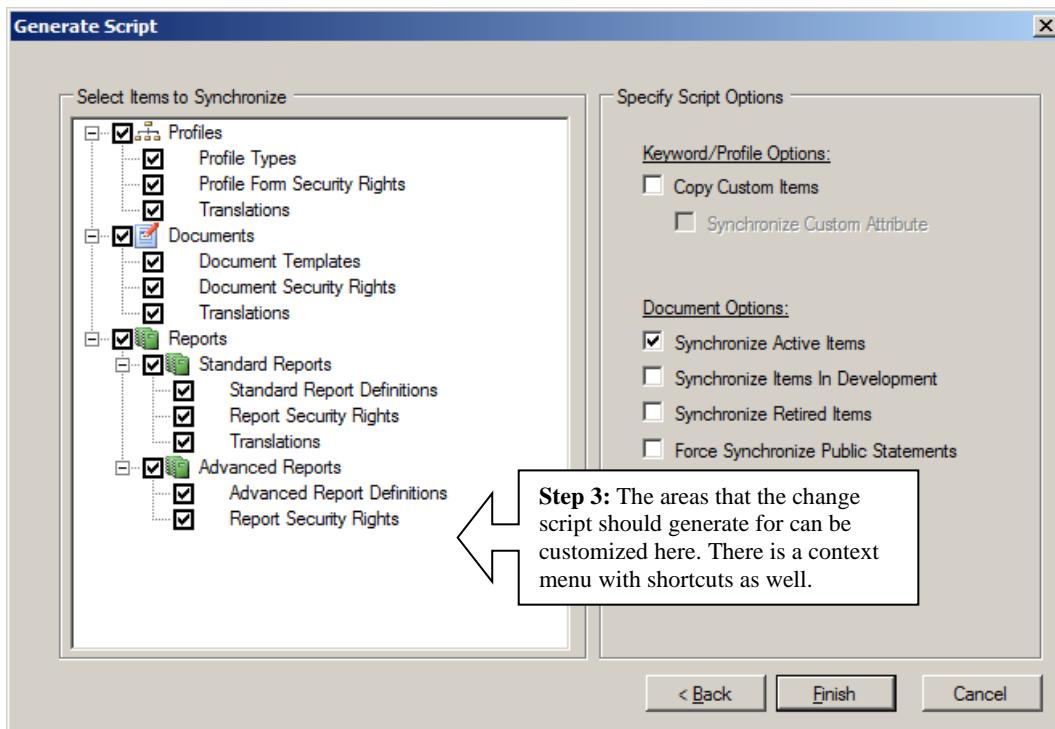
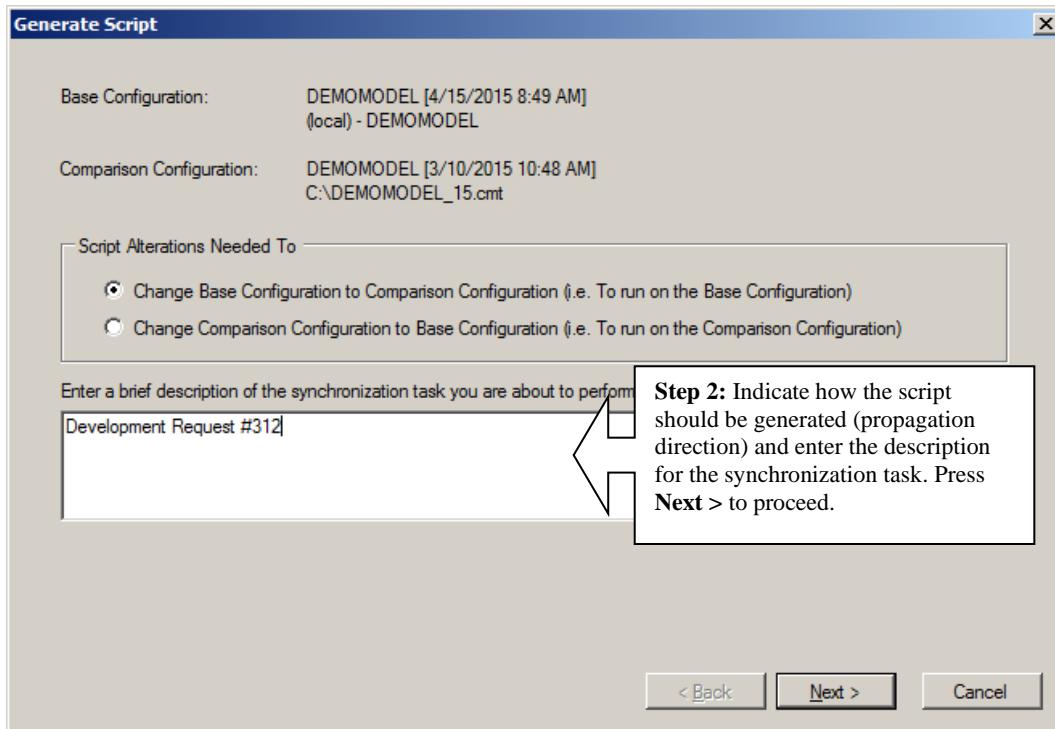
A basic report of changes items can be viewed by clicking on the **View Report...** button on the toolbar.

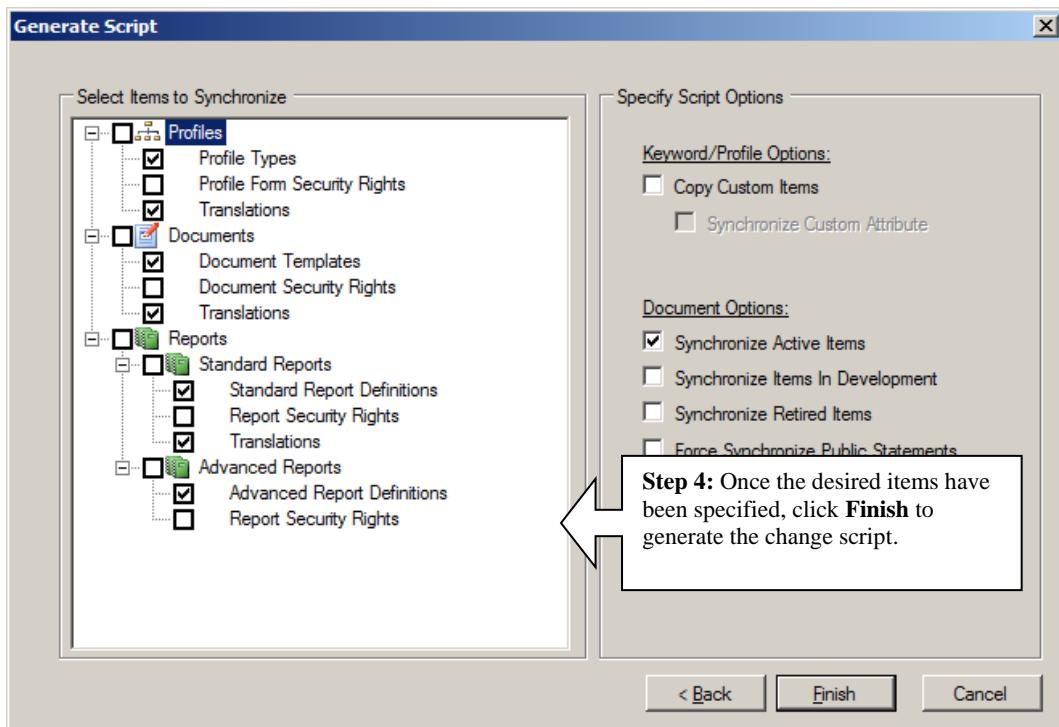
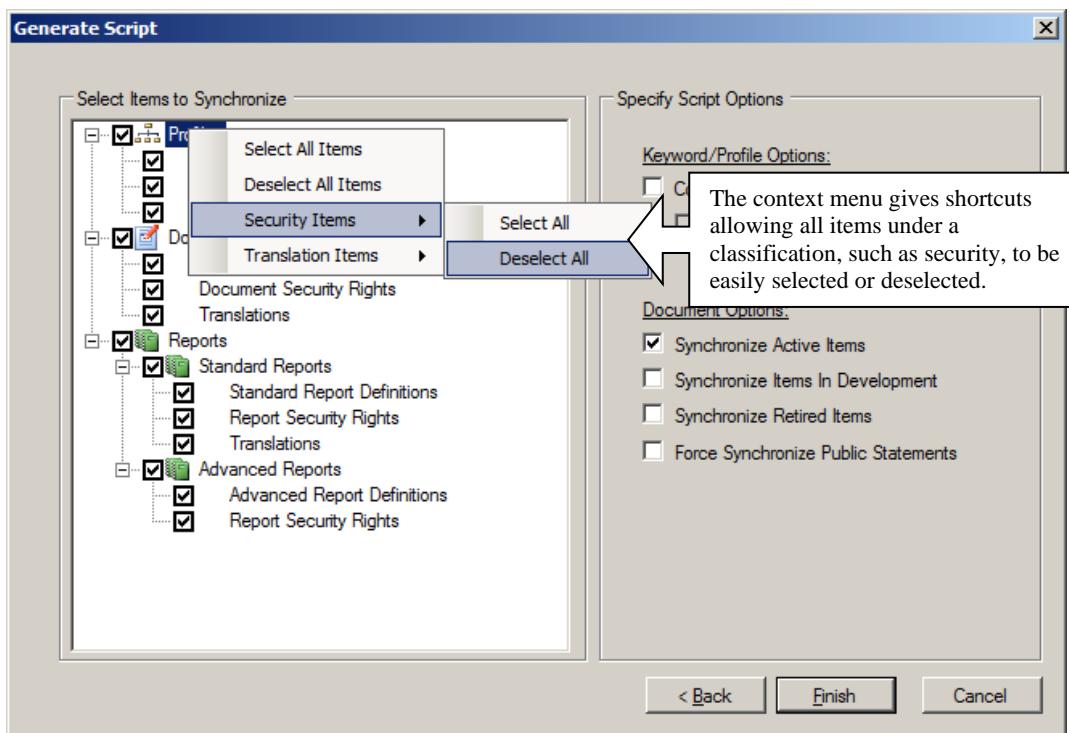


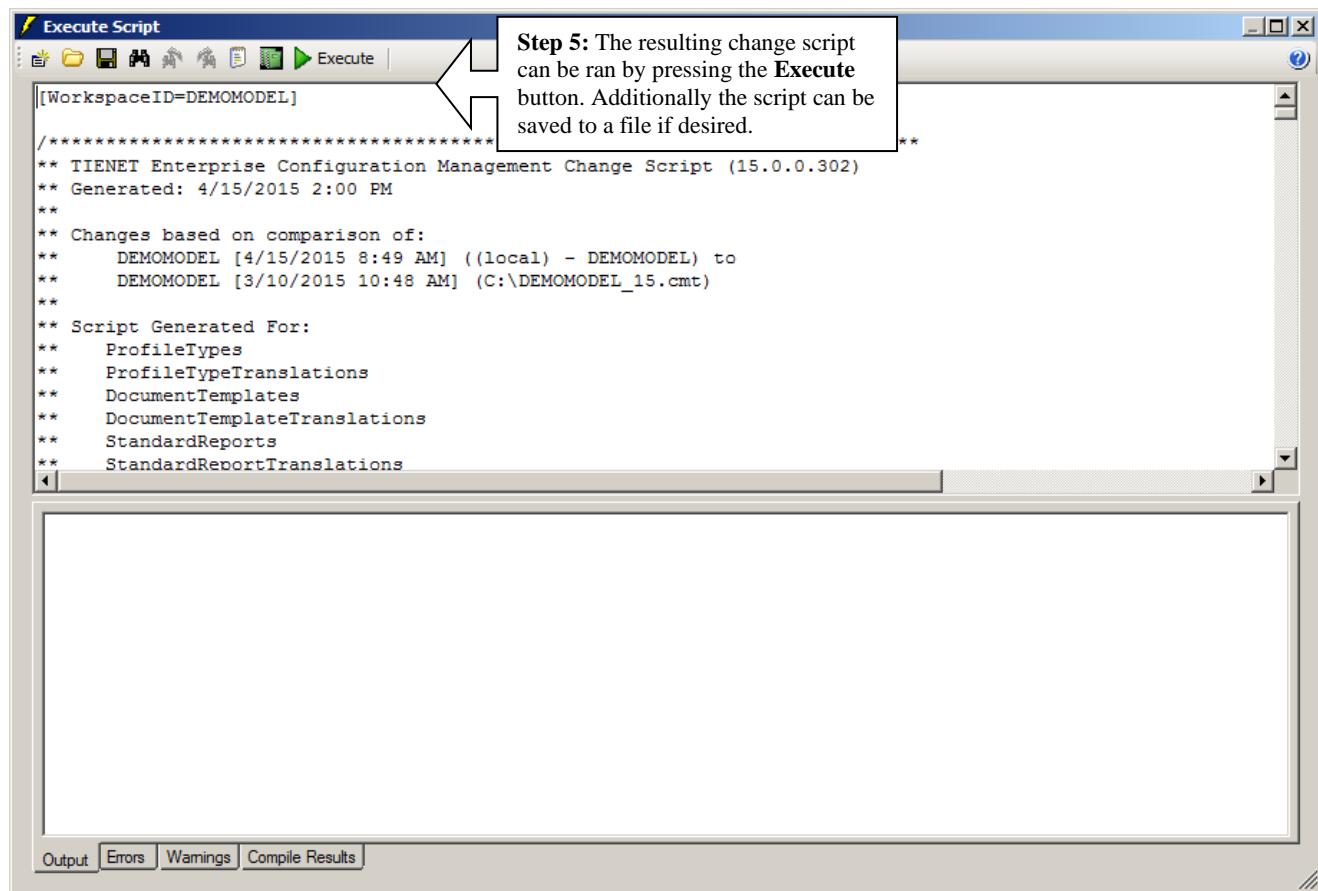
Synchronizing Configuration

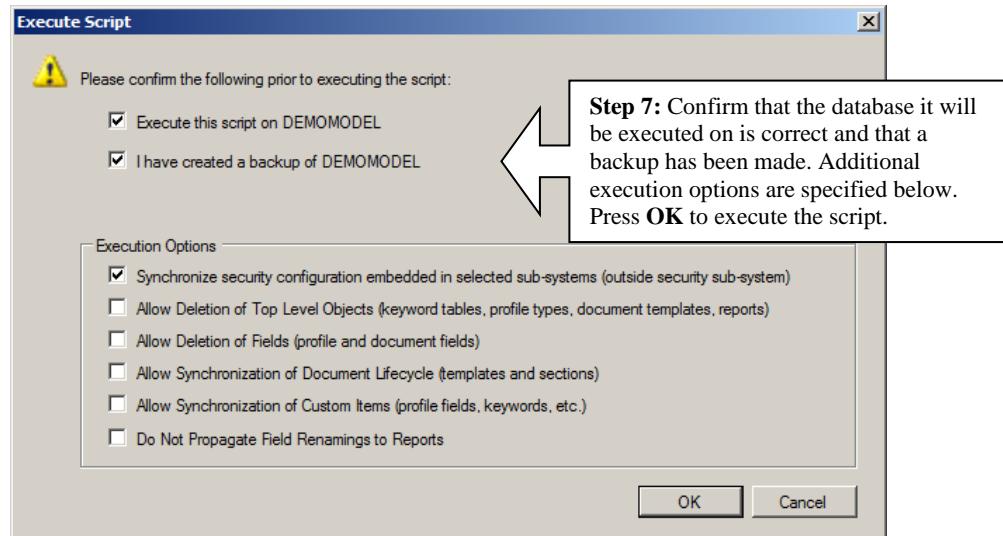
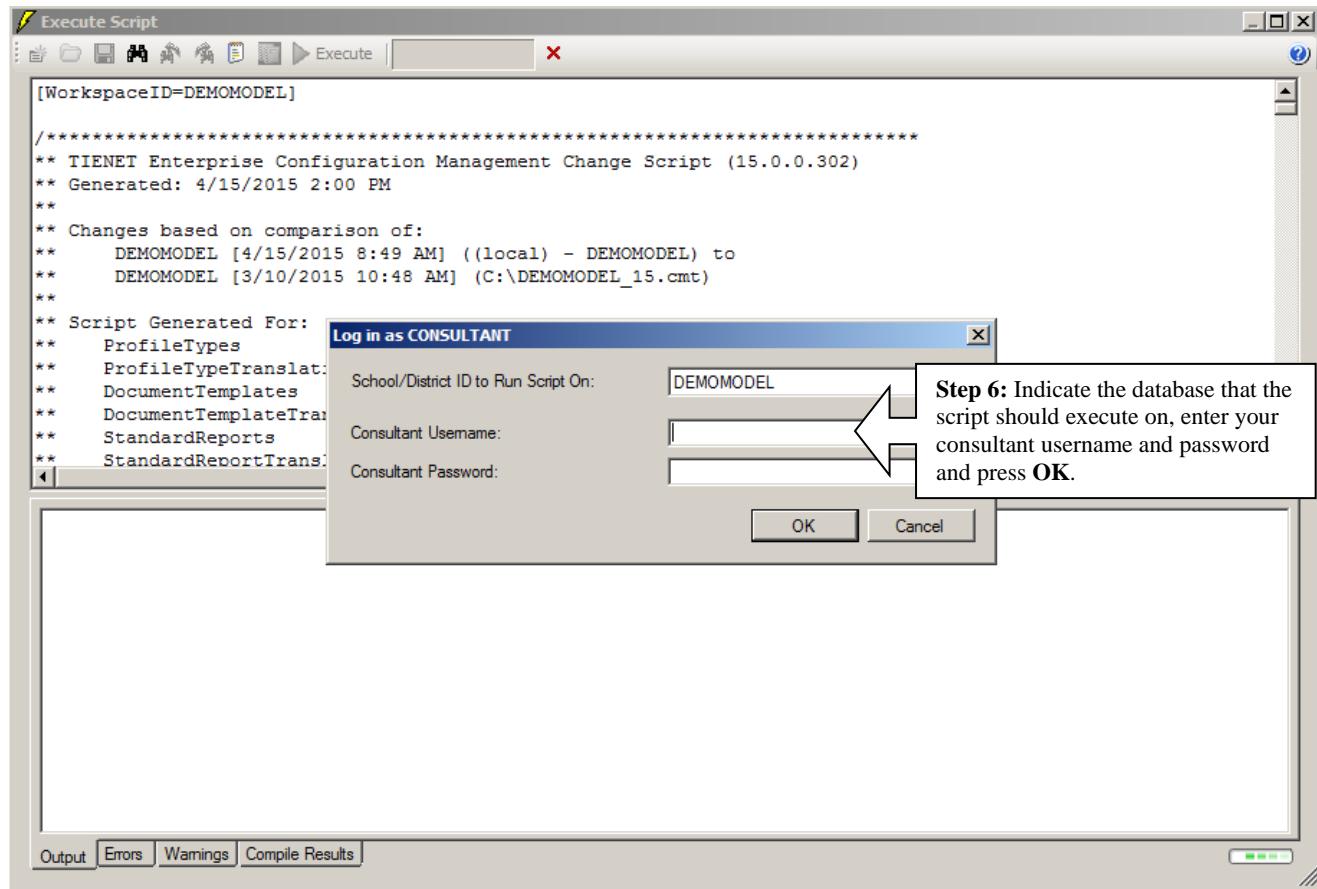
The configuration changes can be copied to a database by using the **Synchronize** functionality in the CMT. This will generate a script with commands that edit the target configuration to match the source database. It is possible to synchronize all changes or only subsets of changes.











```

Execute Script
[WorkspaceID=DEMOMODEL]

/*
** TIENET Enterprise Configuration Management Change Script (15.0.0.302)
** Generated: 4/15/2015 2:00 PM
**
** Changes based on comparison of:
**     DEMOMODEL [4/15/2015 8:49 AM] ((local) - DEMOMODEL) to
**     DEMOMODEL [3/10/2015 10:48 AM] (C:\DEMOMODEL_15.cmt)
**
** Script Generated For:
**     ProfileTypes
**     ProfileTypeTranslations
**     DocumentTemplates
**     DocumentTemplateTranslations
**     StandardReports
**     StandardReportTranslations

```

Set the placement style sheet [Line 38](#)
Modified HTML format for form section 'Students'/'Demographics/Enrollment'
Modified HTML format for form section 'Classes'/'Main' [Line 104](#)
Modified script format for form section 'Staff'/'Main' [Line 142](#)
Modified properties for profile field 'StudentGuardians'/'FirstName' [Line 111](#)
Modified properties for profile field 'StudentGuardians'/'LastName' [Line 112](#)
Modified properties for profile field 'StudentGuardians'/'MiddleName' [Line 113](#)
Modified properties for profile field 'StudentGuardians'/'Profile_Created_On' [Line 160](#)
Modified properties for profile field 'StudentGuardians'/'Profile_Modified_On' [Line 163](#)
Modified properties for profile field 'StudentGuardians'/'Student' [Line 166](#)
Modified associated fields for form section 'Locations'/'Main' [Line 168](#)
Set the translated phrase for 'Bool Val' for language 'es' for form section 'Demographics/Enrollment' in [Line 170](#)

Step 8: All results (messages, errors, warnings) of the script execution are displayed in the **Output** window. Additionally errors and warnings will also be written to their respective windows so they can be reviewed more easily.

Output Errors Warnings Compile Results

CMT Renaming Fields and Other Objects

Whenever renaming a configuration object, such as a field, there is a danger that when the change is synchronized to other databases with the CMT, that the CMT will actually delete the object under the old name and create it under the new name rather than renaming the object. This can and will result in data loss. For this reason, the properties screen for the following object types has a prior names field that allow you to identify one or more prior names:

- Profile Type (set when changing plural name)
- Profile Field (set when changing field name)
- Document Template or Child Template (set when changing Template ID)
- Document Template Sections (set when changing section name)
- Document fields (set when changing field name)
- Keyword Tables (set when changing keyword table name)

- Keyword Table Fields (set when renaming keyword table fields)

The CMT will use the old names to look for objects under the old name and rename them to the new name instead of deleting and adding.

CMT Data Transformations

Data transformation and migration capabilities are sometimes necessary when introducing new field structures that replace existing field structures. Such transformations can be inserted into a generated CMT script by manually inserting a line at the appropriate point in the script. The inserted line should have the format given below where <inlinescript> has exactly the same syntax as a profile update script.

```
+Profiles.ExecuteInlineUpdateScript( "<inlinescript>");
```

Note that inserting a new line into a script adds an additional step, and to account for that, the “steps” number in the following line near the beginning of the script should be manually incremented for each step manually inserted.

```
Initialize(steps: 36);
```

Data transformations may also be required for document data. Except where explicitly specified with a where clause, such transformations are applied to all documents including final documents (which may be unfinalized) and soft-deleted documents (which may be undeleted). However, operations that affect the section structure of existing documents are prohibited as explained in the table below.

The following operations on document data are supported:

Transformation	Explanation and example
Update top level document data	<p>Top level documents cannot be created or deleted in a script but top level field values can be updated using the syntax below:</p> <pre>UPDATE #<DocTemplateID> SET TargetField1=SrcExpression1, TargetField2=SrcExpression2... WHERE FilterCriteria</pre>
Insert (grand) child documents	<p>Child documents associated with repeating sections are prohibited from being inserted since that affects the section structure of existing documents, therefore this command is restricted to repeating row data. Source data expressions for inserting child document data will be in the context of the top level document. Source data expressions for inserting grand child document data will be in the context of parent child documents but may also reference top level field values.</p> <p>Child documents and grand child documents can be inserted using the syntax below.</p>

	<pre>INSERT #<DocTemplateID>#<ChildTemplateID> (TargetField1,TargetField2...) (SourceExpression1,SourceExpression2...) WHERE FilterCriteria</pre> <pre>INSERT #<DocTemplateID>#<ChildTemplateID>#<GrandTemplateID> (TargetField1,TargetField2...) (SourceExpression1,SourceExpression2...) WHERE FilterCriteria</pre> <p>Note that each insert statement variation above will insert at most one new child document row per document, or at most one new grand child document per parent child document. The DISTINCT keyword, which is supported for similar syntax used for profiles, is not supported for document data.</p> <p>New In 20.6.4.0: The following advanced insert syntax allows you to specify a specific source template to insert rows from (potentially multiple rows).</p> <p>To insert child documents from a specified child template source, use the following syntax:</p> <pre>INSERT #<DocTemplateID>#<ChildTemplateIDA> (TargetField1,TargetField2...) FROM #<DocTemplateID>#<ChildTemplateIDB> (SourceExpression1,SourceExpression2...) WHERE FilterCriteria</pre> <p>To insert grand child documents from a specified top-level child template that is not the parent of the grand child template, use the following syntax:</p> <pre>INSERT #<DocTemplateID>#<ChildTemplateIDA>#<GrandTemplateID> (TargetField1,TargetField2...) FROM #<DocTemplateID>#<ChildTemplateIDB> (SourceExpression1,SourceExpression2...) WHERE FilterCriteria</pre>
Update (grand) child documents	<p>The following syntax updates (grand)child document fields to expressions that can reference parent fields. The syntax is given below:</p> <pre>UPDATE #<DocTemplateID>#<ChildTemplateID> SET TargetField1=SourceExpression1, TargetField2=SourceExpression2... WHERE FilterCriteria</pre> <pre>UPDATE #<DocTemplateIDT>#<ChildTemplateID>#<GrandTemplateID> SET TargetField1=SourceExpression1, TargetField2=SourceExpression2... WHERE FilterCriteria</pre> <p>New In 20.6.4.0: More advanced update operations are also possible where a (grand)child template is updated from a specific source using a row-to-row join expression:</p>

	<p>To update child documents from a specified child template source for which row matches can be defined, use the following syntax:</p> <pre>UPDATE #<DocTemplateID>#<ChildTemplateIDA> FROM #<DocTemplateID>#<ChildTemplateIDB> SET TargetField1=SourceExpression1, TargetField2=SourceExpression2... WHERE TargetExpression1=SourceExpression1 and TargetExpression2=SourceExpression2....</pre> <p>To update grand child documents from a specified top-level child template that is not the parent of the grand child template, and for which row matches can be defined, use the following syntax:</p> <pre>UPDATE #<DocTemplateID>#<ChildTemplateIDA>#<GrandTemplateID> FROM #<DocTemplateID>#<ChildTemplateIDB> SET TargetField1=SourceExpression1, TargetField2=SourceExpression2... WHERE TargetExpression1=SourceExpression1 and TargetExpression2=SourceExpression2....</pre>
Delete (grand) child documents	<p>Child documents associated with repeating sections are prohibited from being deleted since that affects the section structure of existing documents. But child documents associated with repeating rows can be deleted using the syntax below (the filter criteria may reference both child document field values as well as top level field values):</p> <pre>DELETEFROM #<DocTemplateID>#<ChildTemplateID> WHERE FilterCriteria</pre> <pre>DELETEFROM #<DocTemplateIDT>#<ChildTemplateID>#<GrandTemplateID> WHERE FilterCriteria</pre>

CMT Conditional Logic and Other Special Cases

A CMT release script can now be manually modified to execute certain lines if a condition based on the globals profile is true. Just enclose the lines dependent on the global profile as follows:

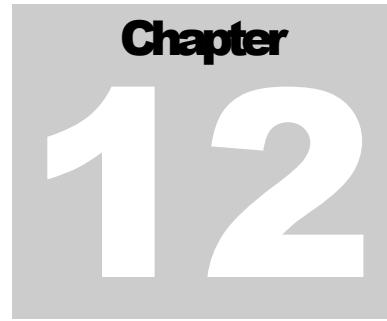
```
if (Versioning.IfGlobalsFormulaIsTrue("globals_conditional_formula")) {
    conditional changes go here
}
```

To preset the ADMIN override setting for a document behavior field to either true or false, insert a statement like this into the release script:

```
Documents.SetDocFieldBehaviorOverride("document_template_id", "behavior_field_name",
true/false);
```

PowerSchool SIS Integration

PowerSchool Special Programs has a rich integration with PowerSchool SIS. Some of the features are enabled through configuration within PSSP.



Overview

After you load the PSSP plugin into PS SIS, several integration features become available within the SIS:

1. Single Sign-On (SSO) into PSSP
2. Embedded content in PS SIS
3. Special Program alerts displayed in PS SIS

 A screenshot of the PowerSchool SIS interface. The top navigation bar includes links for 'Quick Lookup', 'Print A Report', 'Switch Student List (32)', 'Start Page', 'Student Selection', and 'Special Programs'. The user is logged in as Michael Abram. The school is Apple Grove High School, and the term is 18-19 Semester 2. On the left, a sidebar menu lists various student information categories like 'Access Accounts', 'Addresses', 'Attachments', etc., with 'Special Programs' highlighted. The main content area shows a 'Special Programs' section powered by PowerSchool Special Programs. It displays a table with columns for 'Status', 'Creation Date', 'Modification Date', 'Finalization Date', and 'Signatures'. The table contains entries for 'Written Notice/Consent' and '*Active* Individualized Education Program'. The bottom right of the table shows signatures for Michael Abram and Kenneth Kyser.

Documents for 2018/19	Status	Creation Date	Modification Date	Finalization Date	Signatures
Written Notice/Consent	Draft	05/21/2019 Tue, 01:57 PM	05/21/2019 Tue, 01:58 PM	--	--
Active Individualized Education Program	Final	05/21/2019 Tue, 04:29 PM	07/09/2019 Tue, 01:59 PM	07/09/2019 Tue, 01:59 PM	1 (Michael Abram) LA90 (Kenneth Kyser)

Figure 2 Embedded Special Programs Content

Alerts

Special Programs alerts can be configured to write values into PS SIS that indicate certain things for a student – these are called **Alerts**. In PS SIS alerts are displayed in student screens. This gives the ability for teachers to see that a particular student is currently in one or more special programs.

Start Page > Student Selection > Bell Schedule View

Bell Schedule View

Adams, Corby 10 4 AGHS1

	Monday 01/28/2019	Tuesday 01/29/2019	Wednesday 01/30/2019
09:00 AM		World History Wilson, Prescott X 400 08:30 AM - 10:00 AM	
10:00 AM			

Special Programs alerts shown directly in PS SIS.

Start Page > Student Selection > Bell Schedule View

Bell Schedule View

Adams, Corby 10 4 AGHS1

	Monday 01/28/2019	Tuesday 01/29/2019	Wednesday 01/30/2019	Thursday 01/31/2019
09:00 AM		World History Wilson, Prescott X 400 08:30 AM - 10:00 AM		
10:00 AM				

Clicking on an alert shows a window with additional details.

504

Adams, Corby
Is a 504 Student

- View Document List
- View Document List in Special Programs
- Open Document in Special Programs

Each alert has 3 links:

1. **View Document List** – view the embedded special programs page in PS SIS that shows the documents list for the student
2. **View Document List in Special Programs** – SSO into PSSP and go to the documents list page for the student

3. Open Document in Special Programs – if configured correctly, this will SSO into PSSP and take the user to the document relating to the special programs alert

In order to get the 3rd link to work, some configuration is needed to associate an alert with one or more document types. This is configured in the individual document templates as a property.

Step 1: Navigate to the document templates page and then click the desired template to view it

Template Name	Template ID	Status	# Sections
Parent Letter for Intervention (Entrance)	ParentLett	Active	2 Active, 2 Retired
Parent /Guardian Invite High School	PARINV	Active	1
Parent /Guardian Invite Elem & MS	ParInvELMS	Active	1
Student Interview (For Students 10 & Older)	STDINT	Active	1
Student Observation Form (RTI)	SOF	Active	1

Step 2: Click the Properties button for the template and then click Edit Properties.

Output Format	HTML Format	Testing Format	Section Properties	Fields	Section Actions
Edit HTML Format	Lock Section	Edit JavaScript	Add Field...	Section Security	Print

Code snippets in the HTML Format section:

```

{@Globals.DistrictLogo}
{@Globals.ThisDistrict}
{@Globals.Address}
{@Globals.City}, {@Globals.State} {@Globals.ZipCode}
Phone: {@Globals.Telephone}

```

SECTION 504 - PLAN

Student Name: {@FirstName} {@LastName}	ID: {@ID}	Date of Birth: {@BirthDate}
Serving School: {@ServingSchool}		Grade: {@Grade}
Parent/Guardian: {@Parent1FirstName} {@Parent1LastName}		Relationship: {@Parent1Relation}
Home Phone: {@Parent1HomePhone}		Email: {@Parent1Email}
Address: {@Parent1Address}, {@Parent1City}, {@Parent1State} {@Parent1ZipCode}		

Special Programs Curriculum Communication Administration

Template Properties

Template ID	Five04Plan <small>(1-10 characters)</small>	 <div style="border: 1px solid black; padding: 5px;">Scroll to the bottom of the edit template properties screen</div>
Template Name	Section 504 - Plan	
Prior Template ID(s)	<small>(used by CMT to rename from prior template ID or comma-separated IDs)</small>	
Status	Active	
Template Category	Section 504	
Preset Document Labels/Comments	<small>When creating a new document, the end-user enters a 50-character label/comment or picks it from those you specify below. Enter suggested labels/comments separated by carriage returns.</small>	
<hr/>		
Template Options	<input checked="" type="checkbox"/> Require Previous Document to be Finalized Before New One Created <input type="checkbox"/> Within the Same School Year Only	
<hr/>		
Integrated Meetings	<input type="radio"/> Yes <input checked="" type="radio"/> No	
Enabled?	Meeting Type Name: <input type="text"/>	
Language Options	Allow writing documents directly in other languages for which translations are available? <input type="radio"/> Yes <input checked="" type="radio"/> No Is this document template dedicated to another language? <input type="text" value="(No)"/>	
SIS/UC Alert	IS_504	
Data Entry Options	Date Fields - Maximum Days in Future: <input type="text"/> (0-999, optional)	
Custom?	<input type="radio"/> Yes <input checked="" type="radio"/> No <small>(Custom document templates are not synchronized by default)</small>	
<input type="button" value="Accept"/> <input type="button" value="Cancel"/>		

Note that an alert can be associated with more than one document type. The reason for this is that it is not uncommon to have several document types used for the same purpose. For example, there may be several IEP document templates that are used depending on the age of the student.

HTML Format Directives Reference

HTML format directives are essentially commands enclosed in {squiggly} brackets that are embedded in an HTML format. To generate the actual output, PowerSchool Special Programs parses all directives in an HTML format and replaces them with whatever the directives stipulate: a field value or edit box, a correct pronoun for the student's gender, the result of a calculation, etc. PowerSchool Special Programs recognized the directives below when embedded in an HTML format, and in many cases, when embedded in narrative public/private statements or default text for a long text field.

Appendix



Data Directive	Explanation
{FirstName} {LastName}	These directives make it easy to personalize text and use the student's name.
{.he,she} {.him,her} {.his, her} {.his, hers}	These directives are replaced with a pronoun appropriate for the student. In the directive, be sure you use the case that you want. For example, at the beginning of the sentence, use {.He, .She}, but in the middle of a sentence, instead use {.he,.she}.
{~snippetname}	Used to identify positions in the HTML format where the system administrator or CONSULTANT may insert "snippets", which are custom fragments of HTML formatting used to insert custom form content. The snippet HTML format may reference custom document fields. Each snippet is identified by the snippet name that you use in the directive, therefore snippet names must be unique within each section.

<p>{=formula}</p> <p>{=~formula}</p>	<p>Allows you to insert the result of any valid PowerSchool Special Programs formula calculation in the format. If the format is for a document template, the formula must be expressed in terms of the document fields. If the format is for a profile form section, the formula must be expressed in terms of the profile field.</p> <p>The second format with the ~ character suppresses the white space that would otherwise be rendered if the calculated value is empty. This is useful in the case below to prevent a blank line if the calculated value is empty.</p> <pre><div>{=@~formula}</div></pre>
	<p>{=^parent_profiletype_formula}</p> <p>{=^~parent_profiletype_formula}</p> <p>In a document template, this syntax allows embedding of a formula directly in the context of the profile instead of the document. This directive will render as expected when using the print blank document feature, but otherwise this should be used sparingly both for performance reasons and to preserve the historical nature of the document (since the profile contains current data). An example is that if {=Profile.Location} is changed to {=^Location}, it will render even when printing a blank document.</p> <p>This syntax can be used for a similar purpose in child profile forms. In a child profile form, a calculated field in the parent profile could be referenced through an embedded calculation such as {=Student.Age}. However, this calculation would not show when adding a new child profile because it is based on the current child profile that does not exist yet. The {^parent_formula} syntax allows the formula to be written directly in the context of the parent profile such that it will be shown when adding a new child profile. In the example, {=Student.Age} would become {=^Age}.</p> <p>The version of the directive with the ~ character works the same as {=~formula} to suppress white space. For details, see explanation for {=~formula}.</p>

<p><code>{=@globalsformula}</code></p> <p><code>{=@~globalsformula}</code></p>	<p>Allows you to insert the result of a PowerSchool Special Programs formula expressed in terms of fields in the “Globals” profile. Every database has a single globals profile that contains system-wide or district-wide fields such as “DistrictSuperintendent”. This is the most efficient way to create formulas based strictly on global fields and will work as expected even when printing blank documents.</p> <p>The version of the directive with the ~ character works the same as <code>{=~formula}</code> to suppress white space. For details, see explanation for <code>{=~formula}</code>.</p>
<p><code>={\$jsvarname=formula}</code></p> <p><code>{=@\$jsvarname=globalformula}</code></p>	<p>These variations of the PowerSchool Special Programs formula directive expose the value of the formula to JavaScript in a format consistent with the Data Form API. The value is accessed from JavaScript code via a reference to <code>window.form\$jsvarname</code>.</p>
<p><code>{fieldname}</code></p>	<p>When the format is being used to display a form for viewing data, this directive inserts the value of the field. When the format is being used to display a form for editing purposes, this directive inserts an editable field such as a text box, check box, or dropdown menu.</p>
<p><code>{fieldname:modifiers}</code></p>	<p>Same as above, but allows for inclusion of various modifiers that affect the look and feel of the field. A complete list of modifiers is presented later in this section. Keep in mind that multiple modifiers can be included, and there should be no separation between them, for example, you could have a directive such as <code>{StudentIsReady:D+"Ready"- "Not Ready"}</code> for a logical field named StudentIsReady. These modifiers specify that a dropdown menu be used to enter this field, and that “Ready” or “Not Ready” be used in place of “Yes” or “No”.</p>

{-fieldnames}	<p>This directive resets the specified fields (comma-separated list of field names) to their default values, but only if the fields are not visible (in editable form) on the respective form at the time the user submits the form.</p> <p>This directive is compatible with the #IF and #JSIF directives. In the example below, the {-OtherText} directive ensures that the OtherText field is reset to its default value if the Service field is on the form but the OtherText field is not:</p> <pre>#IF StudentNeedsService <label>Service:{TypeOfService}</label> #IF JSIF Service.Other,tag=span <label>Other: {OtherText}</label> #ENDIF {-OtherText} #ENDIF</pre>
{*constraintname}	<p>In a profile form, depending on the evaluation method of the constraint, this directive either anchors a constraint user message to a particular point on the form or generates a constraint button. Note that constraint user messages are displayed above the form by default, but can alternatively be anchored to a particular field or to a precise place on the form (using this directive). If the constraint has an evaluation type of ‘Explicit Button’, the directive instead generates a button on the form that, when clicked, evaluates the constraint for execution. By default, the button shows only when the form is in edit mode. By default, the label of the button is the constraint name and the CSS class “CONSTRAINTBUTTON”. This can be modified as described in the next row below.</p>

{*constraintname:modifiers}	<p>Modifiers only apply if the directive is for a button (see row above).</p> <p>The full set of possible modifiers for a constraint button is {*constraintname:L"label"C"css class"VE}. The L modifier specifies a button label other than the constraint name. The C modifier specifies a CSS class name other than “CONSTRAINTBUTTON”. V specifies that the button will be shown when the form is in view mode, and VE specifies that the button will be shown in both view and edit mode.</p>
{*sectionactionname}	<p>In a document template, this directive anchors the user message for a section action with a “Rollback Saving Section” or “Prevent Section Completion” trigger, or generates a button for a section action with a “Modify Data Upon Save” trigger. The button only shows in edit mode. By default, the button label is the name of the section action and the CSS class name is “SECTIONACTIONBUTTON”. This can be modified as described in the next row below.</p>
{*sectionactionname:modifiers}	<p>The full set of possible modifiers for a section action button is {*sectionactionname:L"label"C"css class"}. The L modifier specifies a button label other than the section action name. The C modifier specifies a CSS class name other than “SECTIONACTIONBUTTON”. It is possible to have a button not linked to a section action that acts like the “Save, Continue Button”. The format for this is {*:L"label"} (not the colon after the asterisk). The A modifier makes the button appear as a hyperlink instead of a button.</p>

{ profilesectionname:modifiers} { childprofiletypename:modifiers}	For document templates only. When a user is editing a section of a document, it may be useful to give the user quick view access to a particular section of the corresponding profile (e.g. student profile) at a particular point in the editing process. This directive produces a button (or alternatively a hyperlink) that opens a specified profile form section or list of child profiles in a popup window for quick access. The full syntax for a button is { profile_section_name:C"CSSClass" L"label"}. If the CSS Class or label is missing, defaults will be used. { profile_section_name:AC"CSSClass" L"label" } will produce a hyperlink (note the A modifier). Use the plural name of the child profile type in place of the section name to open a popup with a list of child profiles. Note that the button or hyperlink will only appear in edit mode.
{ !assessmentdirective }	Assessment directives are preceded with an exclamation point, are supported for student profile form sections only, and allow scores and information from the assessment repository and/or curriculum to be included in a profile form section. These are covered in detail in the next section of this appendix
{ §ionname }	Supported in HTML formats for document template sections only. At runtime, this directive is replaced with the HTML output for the named section from the same document. This is very useful for quickly building an IEP section that will work as a “Snapshot” or shortened IEP. Note that the section with this directive (not the referenced section) should have the “Always Regenerate When Not Final” property; otherwise, the included content may not be kept up to date.

{&templatecode::sectionname::filter}	<p>Supported in HTML formats for document template sections only. At runtime, this directive is replaced with the HTML output from the referenced other document and section.</p> <p>The filter is used to select a particular source document from the template since there may be more than one. The filter is a comma-separated list of one or more join equalities where the left side is an expression in the context of the current/target document, and the right side is an expression in the context of the source document. For example, the following takes a section from a source IEP document from the same school year:</p> <pre>{&IEP::Services::DocHistoryYear=DocuHistoryYear}.</pre> <p>If more than one potential source document exists in the filter, the one that was last created is selected. If no source document is found, or if it does not include the section, no content is included.</p> <p>Note that you can design a section in the source document designed exclusively for referencing from another document. Just enable the “Is Hidden” and “Other” options in this source section.</p> <p>Note that the section with this directive (not the source section) should have the “Always Regenerate When Not Final” property; otherwise, the included content may not be kept up to date.</p>
{%childprofilegrid{}}	<p>Is replaced by a formatted grid (HTML Table) listing rows from child profiles of the profile for which the form is being rendered. Also works with documents, although note that the information is not drawn from document fields, but rather child profiles of the current profile. This kind of directive is covered in detail later in this section.</p>

{^childtemplateid} {^childtemplateid:RH}	Used to position a repeating row grid in a repeating document section. This also supports several modifiers. {^childtemplateid:R} implements a requirement that the user must enter at least one row of data for the section to be complete. {^childtemplateid:H} hides the entire repeating row grid if it contains no rows of data. The grid is not hidden in edit mode so that rows can be entered.
{'tabid:tabtype'}	This directive is used to position a DocuSign e-signature tab. The tabid is either the name of a staff profile reference field marked as a DocuSign signer, or the tab ID for other signers configured for the document template. The tab variant is one of the following: FullName, SignHere, DateSigned, Checkbox.
{>}	Used to position a dynamic barcode in a document section. Note that the directive should be exactly {>}, otherwise, the platform will interpret as {>regionname}, described next.
{>namedblock}	<p>Integrates a read-only clone of a named block of content marked in another source section within the same document template. The named block should be marked in the source section as follows:</p> <pre>{#BLOCK-<name>} content {#ENDBLOCK}</pre> <p>Limitations: Blocks defined in the HTML format of a repeating section can only be used in another repeating section based on the same child template. Blocks cannot be nested. While a block is allowed to include a {^repeatingrow} directive, this directive will not work as expected in another section that uses the block unless the other section also has a repeating row layout defined for the same child template.</p>

Conditional Directives (available in profile form sections and document template sections)	
Conditional Directives	Explanation
{#BLOCK-<name>} content {#ENDBLOCK}	<p>Marks a named block of section content such that another section can include that block in view-only form via a {><i>namedblock</i>} directive.</p> <p>Limitations: Blocks defined in the HTML format of a repeating section can only be used in another repeating section based on the same child template. Blocks cannot be nested. While a block is allowed to include a {^repeatingrow} directive, this directive will not work as expected in another section that uses the block unless the other section also has a repeating row layout defined for the same child template.</p>
{#IFNEW} <i>HTML Formatting</i> {#ENDIF}	This is available for profiles only and includes the HTML formatting if and only if the user is manually adding a new profile. Once the profile is saved and created, this directive will not include the HTML formatting.
{#IFEDIT} <i>HTML Formatting</i> {#ENDIF}	The HTML formatting between the {#IFEDIT} and {#ENDIF} directives is only included when the form is used in edit mode.
{#IFVIEW} <i>HTML Formatting</i> {#ENDIF}	The HTML formatting between the {#IFEDIT} and {#ENDIF} directives is only included when the form is used in view mode.
{#IF formula} <i>HTML Formatting</i> {#ENDIF}	The HTML formatting between the {#IF <i>formula</i> } and {#ENDIF} directives is only included if the formula evaluates to true. If this is used in a profile form section, the formula is expressed in terms of the profile fields. If this is used in a document template section, the formula is expressed in terms of the document fields.

{#IFVIEWOR formula} HTML Formatting {#ENDIF}	The HTML formatting between the {#IFVIEWOR formula} and {#ENDIF} directives is included if the form is in view mode (formula ignored in this case), or if the form is in edit mode and the formula evaluates to true.
{#IFEDITOR formula} HTML Formatting {#ENDIF}	The HTML formatting between the {#IFEDITOR formula} and {#ENDIF} directives is included if the form is in edit mode (formula ignored in this case), or if the form is in view mode and the formula evaluates to true.
{#IF-RO formula} HTML Formatting {#ENDIF}	If the formula evaluates to true, then any fields in the content are forced to be read only.
{#IF_USER formula} HTML Formatting {#ENDIF}	If the current user meets the criteria defined in the staff-based formula (or if the user is the ADMIN or CONSULTANT), the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents).
{#IF_EDITANDUSER formula} HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user meets the criteria defined in the staff-based formula (or if the user is the ADMIN or CONSULTANT), the HTML formatting is included.
{#IF_USER_RO formula} HTML Formatting {#ENDIF}	If the current user meets the criteria expressed in the staff-based formula, the contents will be read-only.
{#IF_ADMINCONSULTANT} HTML Formatting {#ENDIF}	If the current user is an ADMIN user, CONSULTANT user or staff user with the 'Access Admin Form Content' privilege, the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents).
{#IF_ADMIN} HTML Formatting {#ENDIF}	If the current user is an ADMIN user or staff user with the 'Access Admin Form Content' privilege, the HTML formatting is included. This is only available

	for profile forms, not document templates (where it would cause problems with finalized documents).
{#IF_CONSULTANT} HTML Formatting {#ENDIF}	If the current user is the CONSULTANT, the HTML formatting is included. This is only available for profile forms, not document templates (where it would cause problems with finalized documents).
{#IF_EDITANDADMINCONSULTANT} HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user is the ADMIN or CONSULTANT, the HTML formatting is included.
{#IF_EDITANDADMIN} HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user is the ADMIN, the HTML formatting is included.
{#IF_EDITANDCONSULTANT} HTML Formatting {#ENDIF}	If the form is in edit mode and also the current user is the CONSULTANT, the HTML formatting is included.
{#IF_USERORFINAL <i>formula</i> } HTML Formatting {#ENDIF}	If the current user meets the criteria defined in the staff-based formula (or if the user is the ADMIN or CONSULTANT), the HTML formatting is included. This is only available for document templates and only when the current section has the “Always Regenerate When Not Final” property.
{#IFFIELD <i>fieldname</i> } HTML Formatting {#ENDIF}	This directive is available only in profile forms and is intended to provide support for field level security. If you wish to exclude part of the form section markup if a particular field is unavailable due to field level security, enclose that part of the markup with this directive. This will conditionally include the markup only if the specified field is available.
{#JSIF <i>fieldreference(s), options</i> } HTML Formatting {#ENDIF}	See the “ In the context of documents only, several functions are available that are not part of the API but which nonetheless can be called to invoke one of the save or cancel editing buttons. These functions are as follows:

	<ul style="list-style-type: none"> • <code>doSaveDoneEditing()</code>: This function is equivalent to clicking the “Save, Done Editing” button. • <code>doSaveContinueEditing()</code> or <code>doSaveContinueEditing(bSkipAttemptComplete)</code>: This function is equivalent to clicking the “Save, Continue Editing” button. Normally, clicking the “Save, Continue Editing” button also checks whether the section meets all completion requirements, and if so, marks it as complete. To avoid checking for section completion, the <code>doSaveContinueEditing</code> function takes an optional <code>bSkipAttemptComplete</code> parameter for which you can pass <code>true</code> to skip the behavior (e.g. <code>doSaveContinueEditing(true)</code>). • <code>doCancelEditing()</code>: This function is equivalent to clicking the “Cancel, Editing” button. <p>Conditional Form Logic: #JSIF Directive” section on page 333.</p>
<code>{#IF_SERVICECAPTURECONTEXT “modifiers”}HTML Formatting{#ENDIF}</code>	Supported in service capture forms only. This allows arbitrary content to be included or excluded based on the service capture context (individual service, group service, etc). The modifiers are exactly the same ones supported by the ‘M’ field modifier described in the next table.
<code>{#IFTRANS languageCode} HTML Formatting {#ELSEIFTRANS <i>languageCode</i>} HTML Formatting... {#ELSE} HTML Formatting {ENDIF}</code>	Used in document templates only. This allows the document template to show different content depending on the language version that is being viewed or edited. The content in the #ELSE block is shown for the default (non-translated) version of the document.
<code>{#TRANS_BLOCK_BEGIN} content {#TRANS_BLOCK_END}</code>	Used in document templates only. Used to ensure that the enclosed content is translated as a single block.

<pre>{#TABLE_HEADER} <table> <thead style="display:table-header-group"> <tr> <th>colheader1</th> <th>colheader2</th> </tr> </thead> {#END_TABLE_HEADER} <tr><td>cellcontent1</td> <td>cellcontent2</td></tr> {#TABLE_FOOTER} </table> {#END_TABLE_FOOTER}</pre>	<p>Support in profile form sections only. These directives can be used to construct a list style view that spans multiple forms, based on an HTML table construct. When multiple forms are printed, the content between {#TABLE_HEADER} and {#END_TABLE_HEADER} are included only for the first form printed, and the content between {#TABLE_FOOTER} and {#END_TABLE_FOOTER} are included only for the last form printed. The <thead> style used in the example to the left ensures that the column headings will print on each page if the list runs to multiple pages.</p> <p>Important: If you use this in a form section, be sure to enable the “No Page Break When Multiple Forms Printed” checkbox in section properties or strange results will occur!</p> <p>FYI – About Vertical Column Headers: To achieve vertically oriented column headers, add the style “.VERTICAL { width:1px; writing-mode:tb-rl; }” to the style sheet and use it as a class for your column headers, <td class=VERTICAL></p>
---	--

FYI-Important: About Conditional Directives: The word immediately after the pound sign (e.g. “IFEDIT” in {#IFEDIT}) must be in upper case. Any word(s) immediately after #ENDIF are ignored so that you can document what the #ENDIF corresponds to, for example {#ENDIF-IFVIEW}.

The follow modifiers are supported for the {fieldname:modifiers} directive described previously:

Field Directive Modifiers (available in profile form sections and document template sections)	
Modifier	Explanation
~	Suppress the white space that is by default inserted for an empty field.
!	In profile forms only, marks the field as required. The form will not be accepted until this field is filled in.
@	(ADVANCED) This is supported in document templates only and makes the field invisible to the user when in edit mode, but such that it is still an internally

	<p>“editable” field that is writable by JavaScript. The modifier has no effect in view mode.</p> <p>A variation of this modifier (currently supported for character and non-stylized long text fields only) allows the invisible field to still display its value as plain text while in edit mode, even when the field has been modified by JavaScript. Instead of a directive like {Field:@ } which only gets you invisibility, use {Field:@"V" } which prevents the editable field from showing, but automatically displays the value as plain text in the same position.</p> <p>It may also be useful to apply any of the above variations to a field that is marked as read only in the data dictionary. In this case, you can include an E modifier, such as {Field:@E}, to force it to be editable but the @ modifier still makes it invisible to the end user.</p>
^	(ADVANCED) This is supported for character, integer, numeric, date, date/time, keyword, shorttext, and long text fields (except stylized text fields) and makes the field writable by javascript but appears to the user as a grayed out read-only field. NOTE: In versions prior to 22.6.4.0, only the character, shorttext, and long text data types were supported.
A	This is supported in document templates only. It is used in conjunction with dropdown and checkbox fields only. If the user clicks a checkbox or changes a dropdown with this modifier, the “Save & Continue Editing” button will automatically be invoked.
C C"="	<u>Keyword field:</u> If used in a section linked to multiple curriculum outlines (e.g. IEP goal section), this modifier directs the system to use the keyword identified in this table as the root of the curriculum that should appear when the user clicks the button to select from a curriculum. If you use the C"=" variation, that further instructs the system to not allow the user to change the curriculum in the curriculum popup, and to furthermore prevent the user from bringing up any curriculum until one is selected in this field. Note that the keyword table must provide the curriculum root corresponding to each keyword. This can be done either by making the keywords identical to the curriculum root, or alternatively by adding a character column to the keyword table named “CurriculumRoot” that identifies the curriculum root for each keyword.
D	The behavior of this modifier depends on the data type: <ul style="list-style-type: none"> • If used with a logical (yes/no) field, the Yes/no options will be presented in a dropdown menu rather than using checkboxes.

	<ul style="list-style-type: none"> • If used with an integer field that has a minimum value and a maximum value, the integer field will be presented as a dropdown menu of all values between the minimum and maximum inclusive. You can also specify an interval. For example, if a field has a minimum value of 5 and a maximum value of 25, you can present {5, 10, 15, 20, 25} in the dropdown using the syntax (myinteger:D"5") where 5 is the interval. • If used in with a keyword field, the options will be presented as mutually exclusive checkboxes instead of a dropdown menu. Mutually exclusive checkboxes can also be configured as horizontal list of checkboxes or checkboxes in a grid layout. Simply specify the directive as {keywordfield:DW###} where ### should be replaced by the desired number of columns. If this number is equal to or greater than the number of keywords, this will result in a horizontal layout. If the W modifier is omitted, the default number of columns is one resulting in a default vertical layout. Any number between results in a grid layout. • If used in with a date/time field, the field will show only the date component if the “D” is upper case or only the time component if the “d” is lower case. This allows you to position the date and time components in different positions on the form (you have two directives – one with D and one with d). Note that you cannot have an editable time component without an editable date component, although you can make the date component invisible using the “@” modifier as long as date component will always be filled with a default value.
E	For document templates only. In certain circumstances, you might want to force a field that is otherwise marked as read-only in the data dictionary to be editable in a specific special situation. To do this, apply the E modifier.
F" <i>filterspec</i> "	If used in conjunction with a keyword selection field, the <i>filterspec</i> must be the name of the keyword table field/column to be used for filtering. The dropdown menu will only show keywords for which the specified keyword table field/column is neither EMPTY nor FALSE. You can also specify multiple fields/columns in the <i>filterspec</i> by separated them with commas. If the first keyword table field/column referenced in <i>filterspec</i> is itself a keyword selection field referencing a different keyword table, the dropdown menu will have a hierarchical organization that categorizes the keywords based on the values in that field/column. An example would be “diagnostic code” keywords that are categorized by the service type to which they apply. In this case, ‘service type’

	<p>would be a field/column in the diagnostic codes keyword table that references a services types keyword table. See how this looks below.</p>  <pre> (none) Physical Therapy PT Evaluation PT Individual Therapy PT Individual Therapy by Assistant PT Group Therapy PT Group Therapy by Assistant Occupational Therapy OT Evaluation OT Individual Therapy OT Individual Therapy by Assistant OT Group Therapy OT Group Therapy by Assistant Speech and Language services SLP Evaluation SLP Individual Therapy SLP Individual Therapy by Assistant SLP Group Therapy SLP Group Therapy by Assistant </pre>
	<p>If used in conjunction with a profile reference field, the <i>filterspec</i> will be a logical criteria formula for filtering the results displayed by the lookup popup window. The formula should be in the context of the profile type being looked up. Note that when a filter formula is applied using the F modifier this way, the lookup results immediately appear without the user needing or even being allowed to conduct a general search. But if you would like the user to be able to “escape” the filter and conduct a general search, you can add a hint in the form of a comment appended to the end of the formula, like this F"formula;AllowSearch". Note that the filter formula does not prevent the user from directly typing an ID in the textbox that is outside of the filter. If strict validation is required, a section action must be added to supplement the filter.</p> <p>In certain circumstances, it is possible to embed information from the current profile/document section into the filter formula for a profile reference field using a placeholder enclosed with \$ characters. The placeholder can be a name of a profile/document field linked to the same profile/document section. Documents (not profiles) also support the symbol “Profile” enclosed in \$ characters, which embeds a reference to the profile of the document. For example, in a student document, the directive below filters the lookup window to any teachers of any classes the student is in.</p> <pre>{Teacher:LF"EXISTS(ClassStaffRoster, EXISTS(ClassStudentRoster,Student=\$Profile\$))"}</pre> <p>If used in conjunction with a school year type data field, the <i>filterspec</i> will be a range of years relative to the current school year. For example F"-1,+1" will</p>

	<p>include last year, the current year, and the next school year. F"-10,0" will include the last ten years and the current year, but not future years.</p> <p>If used in conjunction with a long text field on a section linked to the curriculum, this can specify a comma-delimited list of curriculum outline level names (singular form) to which the long text field will be mapped. The end-user will only be able to insert statements at those levels into the box. If the level names are prefixed with a tilde character (e.g. F"~AnchorStandard"), the curriculum statement labels will be inserted into the long text field instead of the descriptions.</p>
F"field/column" O"masterfield"	<p>The previous description for the F modifier describes the capability to produce a “hierarchical” keyword dropdown menu if the F modifier references a keyword table field/column that provides categories (i.e. a keyword table field pointing to a second keyword table that delineates the categories). In this specified situation only, the O modifier can be introduced in conjunction with the F modifier to dynamically filter the keyword dropdown based on the user’s selection from a specified “master” keyword field dropdown (identified by <i>masterfield</i>). As an example, assume that there is a “ServiceType” field (keyword dropdown) and a “DiagnosticCode” field (keyword dropdown). The keyword table containing the dialog codes has a column named CodeServiceType that identifies the service type corresponding to each diagnostic code. Then a directive like {DiagnosticCode:LF"CodeServiceType"O"ServiceType"} can be used to link the DiagnosticCode field such that the DiagnosticCode field is dynamically filtered based on the ServiceType field.</p> <p>The F and O modifier combination will work in other situations as well. Assume that there is a “ServiceType” field (keyword dropdown) and a “DiagnosticCode” field (keyword dropdown). The keyword table containing the service types has a column named “Category” that identifies the service category corresponding to each service type. The keyword table containing the diagnostic codes also has a column named “ServiceCategory” that identifies the service category corresponding to each diagnostic code. Then directives like those shown below can be used to link the DiagnosticCode field such that the DiagnosticCode field is dynamically filtered based on the ServiceType field.</p> <pre>{Service:LF"Category"} {DiagnosticCode:LF"ServiceCategory"O"ServiceType"}</pre>

	Note: This type of filtering should also be backed up by profile constraints, since it is possible for a user to use the back button to “trick” this filtering system.
H##	Used in edit mode only for data fields with a data type of short-text or long-text. Gives the height (in lines) of the multi-line text box (replace ## with the actual height).
I##	Used only for long text fields in document templates only. Gives an alternative height in lines (replace ## with the actual height) used when the document is printed with extra space for handwriting (an option when printing a document). If used in conjunction with a logical field, please review the Digital Signature area for more information.
J	This modifier allows you to write a javascript notification function that automatically gets called when the value of the field changes on a form in edit mode. In the field directive, you specify the javascript function as in the example {Goal:W100H5J"api:onChangeGoal"} where onChangeGoal is the name of a function you define in the JavaScript tab. The javascript function will receive as a parameter the name of the field that has changed. If the field in question has an auto-postback, the javascript function will be called just before the auto-postback occurs. <pre>function OnChangeGoal(strFieldName) { alert('New value is: ' + DataFormAPI.getFieldValue(strFieldName)); }</pre>
K	In profile forms only, this identifies the field as a critical field with an asterisk. This is a visual effect only and entry of the field is not required by virtue of including this modifier.
L L"fieldlabel"	Automatically renders a field label for the field, both in viewing and editing mode. This assumes the fields are being laid out in a two column HTML table where the left column contains field names and the right column contains field values. Hence this directive is replaced with something like <tr><td>fieldlabel</td><td>fieldvalue</td></tr>.

	<p>PowerSchool Special Programs will render a default field label based on the field name, but you can override that by including a specific field label in quotes.</p> <p>If used in conjunction with the S or the I directive with a logical field, please review the Digital Signature area for more information.</p>
M"SDXI"	<p>Used in service capture (service record) forms only to indicate whether a field should be suppressed or not in various situations concerning whether the user has selected multiple students, multiple days, group service, individual service, etc. The M field attribute supports a number of options, as follows:</p> <p>S – Suppresses field if entering common information for multiple students.</p> <p>s – Suppresses field if not entering common information for multiple students.</p> <p>D – Suppresses field if entering common information for multiple days.</p> <p>d – Suppresses field if not entering common information for multiple days.</p> <p>X – Suppresses field if scheduling services in the future.</p> <p>x – Suppresses field if not scheduling services in the future, which implies recording services in the past.</p> <p>I – Suppresses field if entering individual service records after having entered common information for multiple students.</p> <p>G – Suppresses field if entering common information for group service (versus individual service). Note that prior to Version 11.0, selecting multiple students always implied group service, however, it is now possible to enable an option in the service record profile form section that supports a new scenario where the user selects multiple students and then creates individual (not group) service records at different times on the same day(s).</p> <p>g – Suppresses field if not entering common information for group service. See the notes for the G option above.</p> <p>M – Suppresses field if entering common information for multiple students, but for individual student service, <u>not</u> group service. Note that prior to Version 11.0, selecting multiple students always implied group service, however, it is now possible to enable an option in the service record profile form section that supports a new scenario where the user selects multiple students and then</p>

	<p>creates individual (not group) service records at different times on the same day(s).</p> <p>m – Suppresses field if not entering common information for multiple students, but for individual student service, not group service. See the notes for the M option.</p> <p>FYI – You can control the inclusion of arbitrary form content using the Use of {#IF_SERVICECAPTURECONTEXT “options”}content{#ENDIF} directive. The options here are the same options that are available for the M field modifier.</p>
N	<p>The behavior of this modifier varies according to the data type:</p> <p>Profile Reference: If the referenced profile type contains people (students, staff), this causes the referenced person's name to be displayed in first last-name format, rather than last-name, first format.</p> <p>Logical field: If the logical field allows EMPTY values, this causes the No option to be presented before the Yes option (by default, Yes is presented before No).</p> <p>Keyword Selection: Normally, when keyword descriptions appear in a dropdown, and the descriptions are greater than 25 characters, the description may be truncated at the first parenthesis to remove excessive detail. Including the N modifier prevents this truncation.</p>
O	<p>If used with a profile reference field, can be used to specify the name of a character field that will store a "Non-Lookup" text value where the user can type whatever the user wishes. For example, the following will allow the user to choose between a nurse staff lookup field and a non-lookup alternative character field.</p> <pre><label>Nurse: {Nurse:O"NurseNonLookup"}</label></pre> <p>The "O" modifier can be used with a logical field checkboxes to organize them into a "require one or more checks" group in a way that is consistent with them being a single field. The red/yellow colors are applied and change as expected. This is implemented as follows:</p> <ul style="list-style-type: none"> Arrange the logical fields on the form inside of a common parent element such as <code><div></code>, <code><p></code> or <code><fieldset></code>. Note that this common element will change color when the first checkbox is checked.

	<ul style="list-style-type: none"> In the directive for each of the logical fields in the group, use the “O” modifier to specify a “group name” that is unique to the group of checkboxes. For example, {logicalfield:L"label"O"groupname"}. <p>If used with a date or datetime field, allows you to specify a precise output format for the value. An example of an output format is “ddd, dd/mm/yyyy” which would output something like “Mon, 02/28/2012”. The following placeholders can be used in the format:</p> <ul style="list-style-type: none"> d (day, 1-31), dd (day, 01-31) ddd (day of week, Mon-Fri), dddd (day of week, Monday-Friday) h (hour, 1-12), hh (hour, 01-12), H (hour, 0-23), HH (hour, 00-23) m (minutes, 0-59), mm (00-59) M (month, 1-12), MM (month, 01-12) MMM (Jan-Dec), MMMM (January-December) s (seconds, 0-59), ss (seconds, 00-59) tt (AM/PM) yy (year with 2 digits), yyyy (year with 4 digits) Non-letter characters are copied to the output as is, except that \ is an escape character. For example, {MyDateTime:O"HH\h\mm"} would output a time that looks like 23h12. <p>If used with a numeric or integer field, allows you to take control over the output format for the value. Examples are below:</p> <ul style="list-style-type: none"> O"c" 14.1 => \$14.10, £14.10, ... (currency, server culture) O"X2" 15 => 0F (hexadecimal from integer only) O"x4" 254 => 00ffe (hexadecimal from integer only) O"n0" integer 9999001 => 9,999,001 O"n2" numeric 9999001.1 => 9,999,001.10 O"#.##" 0.3678 => .37, 0.3 => .3 O"0.00" 0.3678 => 0.37, 0.3 => 0.30 O"p%" .371 => 37.1% (percentage) O"p1%" .37 => 37.0% (percentage)
--	--

P	If used with a character field, the field value is treated as a fully qualified URL of an image, and the image itself is displayed instead of the URL text (when not in edit mode).
R	The field is read-only or view-only, even if shown in a form being used for editing.
R" <i>StaffRefField</i> "	<p>This specifies that the field will be conditionally readonly depending on who the current user is with respect to the staff reference field specified by <i>StaffRefField</i>. The field will be editable in either of the following two cases: 1) the current user is ADMIN or CONSULTANT, 2) the current user is a staff user and the <i>StaffRefField</i> field is either empty or points to the current staff user. In all other situations, the field will be read only. Note that <i>StaffRefField</i> must appear somewhere on the same form for this to work.</p> <p>Often used with the U"<i>triggerfield</i>" modifier to allow a user to take ownership of the values of a group of fields on a form. In the example below, the user who sets the value of indicator locks down not only its value but the value of the comment field so that other users (other than ADMIN or CONSULTANT) cannot edit them. The identity of the user and the date and time of this event is tracked as well.</p> <pre><p><label>Indicator:{Indicator:R"IndicatorUser"}</label></p> <p><label>Comment:{IndicatorComment:R"IndicatorUser"}</label></p> <p><label>User: {IndicatorUser:U"Indicator"}</label></p> <p><label>Date: {IndicatorDate:U"Indicator"}</label></p></pre>
S	<p>The behavior of this modifier depends on the context.</p> <ul style="list-style-type: none"> • If used in a <u>profile form</u> with a short-text field, a spell-check link will be included next to and for the field when in editing mode. • If used in conjunction with a long text field in a <u>document template</u>, an “Insert Statements” link will appear even if the field presently has no public statements. • If used in conjunction with a basic character field in a document template, the field is spell checked. By default, character fields are not spell checked. • If used in conjunction with a logical field, please review the Digital Signature area for more information.

S" <i>categoryfield</i> "	This can be used to potentially reduce the number of distinct long text fields needed in a document section. The main concept is that there is a keyword field on the form that works in conjunction with a long text field. The keyword field selects the category of "insert statements" that appears when the user clicks the "Insert Statements" link for the long text field. Furthermore the user cannot change the category that has been automatically selected (the category dropdown is not visible in the insert statements popup window). The S" <i>categoryfield</i> " attribute is applied to the long text field, and <i>categoryfield</i> identifies the keyword field which must be editable on the same form. They keyword table for the keyword field must be set up in a particular way. It must include a character column named "StatementsCategory" that specifies the category name corresponding to each keyword. If language localization is being used, there can be an additional column for each localized language named "StatementsCategory_xx" where xx is the ISO language code (e.g. es for Spanish). This would specify the localized category name for the language. Note also that it is important to place the insert statements for the long text field "under configuration management" so that they can only be edited in a controlled fashion (that will not break the relationship to the keyword table). Lastly, use of this feature will disable "private" insert statements because those would not be differentiated by category.
T T"fieldlabel"	Similar to the L directive above, but this does not assume that fields are being laid out in an HTML table, and therefore renders something like fieldlabel: fieldvalue.
U	As of Version 17.1, this modifier is supported for character fields (documents only) to support situations where the value is editable in the section but must be dynamically displayed and updated elsewhere in the same section. There are three specific scenarios where this is supported: 1) A top level character field is editable in a section but the value must be dynamically displayed in all repeating rows on the same section. In the repeating row, the outer field is referenced via { <i>FieldName</i> :U} where U sets up the dynamic updating. When the value of the editable field is changed, the values are automatically updated in each repeating row when the editable field loses focus.

	<p>2) Similarly, in a repeating section, a character field that is editable may be displayed in each repeating row nested within the repeating section.</p> <p>3) A field is editable in the section but the value must be mirrored in another place on the same section, but not in repeating rows. In this case, the mirrored field should be marked with both R (for read only) and U (for automatic updating).</p>
U" <i>triggerfield</i> "	<p>Instructs the system to automatically update or stamp the field when the value of another field specified by <i>triggerfield</i> changes in value. The field being stamped is automatically made readonly to the user since its value is being stamped. The value to stamp the field with is inferred from the data type of the field. At this time, only three data types are supported for the field being stamped. If it is a date or date/time field, it is stamped with today's date or now's date/time respectively assuming that the trigger field's new value is not empty, but if the trigger field's new value is empty, the date or date/time field is cleared. If the field being stamped is a staff reference field, it is stamped with a reference to the staff profile of the current user (or EMPTY if current user is not a staff profile user) assuming that the trigger field's new value is not empty, but if the trigger field's new value is empty, the staff reference field is cleared. Other data types are not supported at this time. IMPORTANT: the triggering field must appear as an editable field on the same form as the field being stamped, and the directive for the triggering field must appear in the layout before that of the stamped field. The stamped field must be on the form at the time form is submitted. If the stamped field is omitted via a conditional directive, it will not be stamped. If necessary, you can diagnose this by searching the page source for "SELFUPDATE" and see which fields to be stamped are on the form at any time.</p> <p>Often used with the R"<i>StaffRefField</i>" modifier to allow a user to take ownership of the values of a group of fields on a form. In the example below, the user who sets the value of indicator locks down not only its value but the value of the comment field so that other users (other than ADMIN or CONSULTANT) cannot edit them. The identity of the user and the date and time of this event is tracked as well.</p> <pre><p><label>Indicator: {Indicator:R"IndicatorUser"}</label></p> <p><label>Comment: {IndicatorComment:R"IndicatorUser"}</label></p> <p><label>User: {IndicatorUser:U"Indicator"}</label></p> <p><label>Date: {IndicatorDate:U"Indicator"}</label></p></pre>
V" <i>field</i> "	If used in conjunction with a multiple-value field, this modifier allows you to specify a field in the associated keyword table to use for filtering the values (determining a subset of values to be displayed). Only values for which the

	specified field is not EMPTY or is not FALSE in the keyword table will be displayed.
W## W##%	Used in edit mode for data fields that use a text box. Gives the width of the text box in number of characters based on an average character width (replace ## with the actual width). For long-text fields only, you can also express the width as a percentage of the available space (e.g. W50%).
Z	<p>The behavior of the Z modifier depends on the data type of the field:</p> <ul style="list-style-type: none"> Keyword Fields: In certain scenarios, it may be useful to have a keyword field dropdown next to a second editable field, but then upon saving, the value of the second field is displayed embedded in the middle of the keyword description. The “Z” modifier for the keyword field directive makes it possible to embed macros in keyword descriptions that get evaluated when the form is not in edit mode. For example, a keyword field named “Measure” has a keyword description of “Student will score between {MinScore} and {MaxScore}”, which includes macros for MinScore and MaxScore. The markup for Measure, MinScore, and Maxscore fields can be something like {Measure:TZ}{#IFEDIT}{MinScore:T},{MaxScore:T}{#ENDIF}. In this fashion, the MinScore and MaxScore fields only appear as separate fields in edit mode. In view mode and due to the “Z” modifier, the MinScore and MaxScore field values appear embedded in the keyword description. Profile Reference Fields: This is supported in profile forms only at this time. In profile forms, the selection for a profile reference field to a child profile can now be embedded inline in form rather than as a lookup. In a profile form for a top level profile, one can configure a profile reference field to a child profile of the current top level profile. Similarly, in a profile form for a child profile, one can configure a profile reference field to a different child profile type that shares the same top level profile. In either case, the profile reference field by default appears with a lookup link that opens a model dialog box allowing the user to select a particular child profile. In some cases, it may be desirable to present the selection embedded inline in the form itself so that the selection is always visible, and this is what is new in 19.4. An example would be a service record that has a field named “Mandate” that references a different “Mandates” child profile. In this case, the directive {Mandate:Z"inline"} presents the selection inline in the form. If you would like to have control over the size of the inline selection in the form, use a format like {Mandate:Z"inline,800-400") to

	<p>specify the width and height of the area. If the inline selection is dependent on other fields on the same form that may dynamically change, you can programmatically cause the inline selection to refresh using the <code>DataFormAPI.invokeLookup</code> method.</p>
<code>+ "wordforyes"</code> <code>- "wordforno"</code> <code>_ "wordfornone"</code>	<p>If used in conjunction with a logical field either configured to use a dropdown or to have separate checkboxes for “Yes” and “No”, these modifiers allow you specify text to be used as labels in place of the default “Yes” or “No”. Note that when a template is translated into another language, these labels extracted and translated within the directive. If the logical field allows empty values and is being displayed as a dropdown menu (with the D modifier), you can also customize the label for the EMPTY value using the underscore modifier.</p> <p>When there are separate checkboxes, the plus and minus modifiers can be used (with or without the label strings) to have separate directives for the Yes and No checkboxes allowing for additional layout flexibility. In the directive for the Yes checkbox, include the plus modifier but not the minus modifier. In the directive for the No checkbox, include the minus modifier but not the plus modifier.</p> <p>These modifiers are also useful for calculated logical fields to control how they are presented. By default, such fields are presented a read-only checkbox. You can present it instead as dual checkboxes using <code>{LogicalField:+ "On"- "Off"}</code> or as a simple text value using <code>{LogicalField:D+ "On"- "Off"}</code>.</p>
<code>- "wordfornone"</code>	<p>Can also be used in conjunction with a keyword field dropdown menu to set the word used for the EMPTY value which is by default “none”. Note that when a template is translated into another language, this word/string is extracted and translated within the directive.</p>

Index

- accessibility**, 316
- active documents**, 237
- Advanced Reports**, 351
 - charting**, 366
 - creating**, 357
 - directives**, 354
 - expando sectopm**, 367
 - measures**, 356
 - optimizing**, 370
- Alerts**, 384
- analysis fields**
 - setup**, 150
 - synchronization**, 152
- assessment directives**, 343
- attributes**, 103
- automatic profile form section**, 96
- button directives**, 93, 234
- calculated fields**, 99
- child profile type**, 106
- child template**, 252
- class general ed roster**, 122
- class profiles**, 49
- CMT**. *See* Configuration
- Configuration**, 371
 - CMT**, 371
 - Compare**, 374
 - Configuration Management Tool**, 371
 - Synchronize**, 375
- Configuration Management Tool**. *See* Configuration
- configuration task**
 - closing**, 54
 - creating**, 54
- configuration task**, 54
- Configuring Packages**, 349
- constraints**, 87
- Data Fields**, 50
- data flow**, 186, 190
- data gathering**, 247
- district profiles**, 48
- document action**, 53
- document actions**, 215
- document field**, 52
- document file attachments**, 273
- document form section**, 52
- document owner security**, 243
- document Repository**, 51
- document template**, 52
 - creating**, 172
 - properties**, 200
 - translations**, 280
- Document template security**, 194
- DocuSign integration**, 297
- end-of-year rollover**, 129
- fast add**, 259
- File Attachments Only Document Template**, 273
- File Fields**, 274
- file-based documents**, 272
- general ed students**, 121
- IEP snapshot section**, 250
- integrated meetings**, 209
- JavaScript**, 323
- language translations**, 280
- letter section**, 177, 210
- localization**, 296
- location profiles**, 48
- many-to-many students/locations security**
 - model**, 125
- master/detail documents**, 294
- meetings**, 209
- Model Versions**, 373
- narrative statements**, 191
- OnFormGetCurRoot**, 323
- OnFormLoad**, 323
- OnFormSubmit**, 323
- optimization**, 311
- PowerSchool SIS**, 384
- profile constraints**, 87

profile field attributes, 103
profile form sections
 customizing, 68
profile grid directives, 345
profile repository, 48
profile tags, 120
profiles, 48
progress monitoring groups
 configuration, 158
PS SIS, 384
public statements, 191
repeating rows, 252, 254
 fast add, 259
repeating rows within repeating section, 260
repeating section, 252, 253
reporting snapshot approval process, 140
Revision documents, 208
screening groups
 configuration, 155
section action, 53
section actions, 215
section extension child template, 270
Security Profiles, 50
service capture configuration, 130
signing documents, 184
SQL timeout errors, 312
staff profiles, 48
student data transfer package, 127
student profiles, 48
stylized text, 183
Synchronize. *See Configuration template*
 creating, 172
 properties, 200
 translations, 280
Template security, 194
timeout errors, 312
translations, 280
user defined functions, 144
workflow, 244